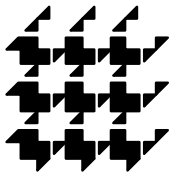
DIMACS
Center for Discrete Mathematics &
Theoretical Computer Science



DIMACS EDUCATIONAL MODULE SERIES

MODULE 03-1
Introduction to Clustering Algorithms:
Hierarchical Clustering
Date prepared: March 17, 2003

Alexander Kheyfits [1]
Bronx Community College of the City University of New York
University Avenue at 181st Street, Bronx, NY 10453
email: alexander.kheyfits@bcc.cuny.edu

DIMACS Center, CoRE Bldg., Rutgers University, 96 Frelinghuysen Road, Piscataway, NJ 08854-8018
TEL: 732-445-5928 • FAX: 732-445-5932 • EMAIL: center@dimacs.rutgers.edu
Web: http://dimacs.rutgers.edu/

Founded as a National Science Foundation Science and Technology Center and a Joint Project of Rutgers University, Princeton University, AT&T Labs - Research, Bell Labs, NEC Laboratories America and Telcordia Technologies with affiliated members Avaya Labs, HP Labs, IBM Research, Microsoft Research.

[1]Supported by the National Science Foundation (Grant DUE 9752776)

DIMACS EDUCATIONAL MODULES SERIES
Description of Module 03-1

Title:

Introduction to Clustering Algorithms: Hierarchical Clustering

Author:

Alexander Kheyfits

Abstract: Clustering algorithms separate (partition) discrete data sets into disjoint groups-clusters, such that every cluster contains only the elements close to each other in a precisely defined sense. These procedures are widely used in mathematical taxonomy, management, and many other applications of mathematics. In this module we discuss the simplest and commonly used hierarchical algorithms for clustering — Hubert's single-link and complete-link algorithms. These algorithms are called hierarchical because they build up an hierarchy of larger and larger clusters. The algorithms are based on the properties of a graph describing an initial collection of objects, and the terms *single-link*, *complete-link* refer to the methods of combining two subgraphs in one larger subgraph.

The module is aimed at freshman and sophomore students studying finite mathematics, introductory discrete mathematics, or statistics. So, only a minimal, high-school background in mathematics is assumed. In particular, we do not expect any knowledge of graph theory or probability theory. All relevant graph-theoretical definitions (like spanning trees, etc.) are discussed and illustrated by examples. The algorithms considered are simple and can be examined in an introductory computer science course. We do not perform any formal analysis of the algorithms, however we present them in a pseudocode form. The module can be used in the classroom and for the students' projects.

It is supposed that after active studying the module and working the included exercises, the reader

- will learn some basic concepts of graph theory together with their simple applications,
- will learn basic concepts of cluster theory and clustering algorithms,
- will be able to apply these algorithms for clustering small (since we do not discuss any software issues) arrays of data,
- will have enough background to learn and apply computer software for clustering real data,
- will be able to study more advanced literature on classification and clustering.

Informal Description:

In this module we discuss how to split a set of different objects into several groups of more or less similar entities. For example, when a manager of a big supermarket decides where to place various goods, she has to solve a problem of cluster analysis. Cluster analysis is used in any field concerned with classification of data. Examples are numerical taxonomy, design of the Internet, classification of natural languages, or image processing, to name just a few. The readers of the module will learn how to form clusters step by step, first in an informal

way, considering a model example, and then using some simple graph theory. These necessary concepts of the graph theory are also introduced in the module. In the last section a real case study is considered.

Target Audience:

The module is aimed at freshman and sophomore students studying finite mathematics, introductory discrete mathematics, or statistics. It may be used by well-motivated high-school students.

Prerequisites:

Only a minimal, high school background in mathematics is assumed. In particular, we do not expect any knowledge of graph theory or probability theory. All relevant graph-theoretical concepts, such as spanning trees, are discussed and illustrated by examples. The algorithms considered are simple and can be examined in an introductory computer science course. We do no formal analysis of the algorithms, however we present them in a standard pseudocode form.

Mathematical Field:

Graph theory; Classification theory; Cluster analysis.

Applications Areas:

Classification theory; Mathematical and biological taxonomy.

Mathematics Subject Classification:

MSC (2000) Primary 91C20; Secondary 05C90, 90B80, 92B10.

Contact Information:

Alexander Kheyfits
Dept. of Math. and Computer Science
Bronx Community College/CUNY
University Avenue at W. 181st Street
Bronx, NY 10453

Tel.: (718) 289-5616
E-mail: alexander.kheyfits@bcc.cuny.edu

Other DIMACS modules related to this module:

Module03-7

TABLE OF CONTENTS

ABSTRACT

Clustering algorithms separate (partition) discrete data sets into disjoint groups-clusters, such that every cluster contains only the elements close to each other in a precisely defined sense. These procedures are widely used in mathematical taxonomy, management, and many other applications of mathematics. In this module we discuss the simplest and commonly used hierarchical algorithms for clustering — Hubert's single-link and complete-link algorithms. These algorithms are called hierarchical because they build up an hierarchy of larger and larger clusters. The algorithms are based on the properties of a graph describing an initial collection of objects, and the terms *single-link*, *complete-link* refer to the methods of combining two subgraphs in one larger subgraph.

The module is aimed at freshman and sophomore students studying finite mathematics, introductory discrete mathematics, or statistics. So, only a minimal, high-school background in mathematics is assumed. In particular, we do not expect any knowledge of graph theory or probability theory. All relevant graph-theoretical definitions (like spanning trees, etc.) are discussed and illustrated by examples. The algorithms considered are simple and can be examined in an introductory computer science course. We do not perform any formal analysis of the algorithms, however we present them in a standard pseudocode form. The module can be used in the classroom and for the students' projects.

It is supposed that after actively studying the module and working the included exercises, the reader

- will learn some basic concepts of graph theory together with their simple applications,
- will learn basic concepts of cluster theory and clustering algorithms,
- will be able to apply these algorithms for clustering small (since we do not discuss any software issues) arrays of data,
- will have enough background to learn and apply computer software for clustering real data,
- will be able to study more advanced literature on classification and clustering.

1 INTRODUCTION

Suppose a student wants to put some money into a mutual fund. To make the right choice, she may consider many different funds on the basis of their long-term and short-term performance, the manager's philosophy of investing, administrative costs, and so forth. Comparing various funds, she can pick up the few that are more suitable for her goals. The things under consideration, like mutual funds, are called *objects* or *entities*. The properties of objects, like performance or attitude to risk, are called *features*, or *variables*, or *attributes*. If every object is characterized by several variables, it is difficult to compare different objects, and we want to get a kind of a "common denominator" to measure *similarity* of the objects. We can separate all available funds into several groups containing similar funds.

Such classification is useful in many occasions. For instance, if the investor hears about a new fund within a short time after its inception, it is hard, without any information, to make a prediction of the fund's future performance based on its own history. If our student can include the fund into a group of several similar funds, she can apply the information on the whole group to each member and make a more reliable prediction. Furthermore, if we have a lot of similar objects, it is often just impossible to study every one of them separately, but we can study a representative of each group of similar objects and apply the information derived to every item[2].

To perform such analysis, we first separate the objects into smaller groups, called *clusters* (overlapping groups are sometimes called *clumps*). This process, called *clustering*, is an essential part of *cluster analysis*. In this module we discuss initial concepts and algorithms of this subject. The list of references at the end of the module contains more extended and advanced expositions of the subject.

Obviously, the objects combined in a group should have some common features and properties. The more two objects have in common, the less is their *dissimilarity*. Ultimately, the similarity of identical objects is infinite and their dissimilarity is zero. In cluster analysis, it is often more convenient to measure dissimilarity rather than the similarity of various objects. We do not discuss here how to assign the dissimilarity values to two multivariate objects, because it essentially depends upon particularities of specific problems — see, for example, [4], Chap. 2; [10]. We assume that the dissimilarities are assigned in advance — given a set of objects to be explored, we are provided with a *table* (also called a *matrix*) of their dissimilarities.

Example 1.1 Table 1.1 contains the average altitudes above the sea level of fifteen southern states in the U.S. (see [5], p. 59).

State	AL	DE	FL	GA	KY	LA	MD	MI	MO	NC	SC	TN	TX	VA	WV
Average Altitude	50	6	10	60	75	10	35	30	80	70	35	90	170	95	150

Table 1.1: The average altitudes above the sea level of fifteen southern states in the USA (in tens of feet).

[2]From the mathematical point of view, this approach is not new. We often use the *equivalence relations* to replace a set under consideration with a smaller *factor-set* to simplify our study. Here this idea is applied to clustering the given objects.

If we are interested in the altitudes only, we can consider the difference (or the absolute value of the difference) between the altitudes as a kind of distance, or as a measure of the dissimilarity between two states. Even though this difference is not the real geographical distance, similar quantities, under some specific conditions, are also called distances. In this sense, the dissimilarity between Alabama and Delaware is $50 - 6 = 44$, the dissimilarity between Florida and Georgia is $|10 - 60| = 50$, and the dissimilarity between Florida and Louisiana is 0 — unlike the mathematical distance, the dissimilarity of two different objects can be 0. Table 1.1 can be transformed into a dissimilarity table (Table 1.2 below), where the main diagonal contains only zeros since each object is absolutely similar to itself. We have completed only the upper triangle, because the table is symmetrical with respect to the main diagonal.

	AL	DE	FL	GA	KY	LA	MD	MI	MO	NC	SC	TN	TX	VA	WV
AL	0	44	40	10	25	40	15	20	30	20	15	40	120	45	100
DE		0	4	54	69	4	29	24	74	64	29	84	164	89	144
FL			0	50	65	0	25	20	70	60	25	80	169	85	140
GA				0	15	50	25	30	20	10	25	30	110	35	90
KY					0	65	40	45	5	5	40	15	95	20	75
LA						0	25	20	70	60	25	80	160	85	140
MD							0	5	45	35	0	55	135	60	115
MI								0	50	40	5	60	140	65	120
MO									0	10	45	10	90	15	70
NC										0	35	20	100	25	80
SC											0	55	135	60	115
TN												0	80	5	60
TX													0	75	20
VA														0	55
WV															0

Table 1.2: Dissimilarity table for the average altitudes.

Depending upon the level of dissimilarity we are willing to accept — this level is called a *threshold value* or just a *threshold* — we can form different clusterings. That is, we can split the fifteen states into different clusters. For instance, the following is a partition of these states into eight clusters with a threshold value of 10. That is, the maximum distance between *any* two objects in *each* cluster does not exceed 10:

{DE, FL, LA}, {MD, MI, SC}, {AL, GA}, {NC, KY, MO}, {TN}, {VA}, {TX}, {WV}.

If we select the threshold level of 5, then the corresponding clustering may be the following one:

{DE, FL, LA}, {MD, MI, SC}, {AL}, {GA}, {NC, KY}, {MO}, {TN, VA}, {TX}, {WV}.

We can also set up another clustering with the same dissimilarity level of 5:

{DE, FL, LA}, {MD, MI, SC}, {AL}, {GA}, {NC}, {KY, MO}, {TN}, {VA}, {TX}, {WV}.

We see that this procedure is not generally unique. It is clear also that if we decrease the threshold, some clusters may decompose into smaller ones. Thus, the second and the third clusterings contain more clusters than the first one.

Exercise 1 Construct another clustering of these fifteen states with the same dissimilarity level of 10.

Exercise 2 Construct clusterings of these fifteen states consisting of four or five clusters. Find corresponding threshold values.

Exercise 3 Find a dissimilarity value that generates a unique clustering.

Compare our clusterings once more. While building the second clustering, we relocated some objects. Now, the group {TN, VA} of the second clustering does not belong completely to any cluster in the first clustering. On the other hand, every cluster of the third clustering is contained completely in a cluster of the first one. A process that forms a series of consecutive clusters such that every cluster of the preceding level is a subset of a cluster in the next level, is called *hierarchical clustering*. We begin with a completely disjoint clustering, where every object forms its own single-element cluster. Then step by step we join (*amalgamate*) two or more clusters with the smallest dissimilarity into larger ones, until we reach a threshold value. Such algorithms are called *agglomerative*.

We can also proceed in the opposite direction, as in the example above. Namely, we can depart from a *conjoint clustering*, when one cluster contains all the objects under consideration, and split it repeatedly into smaller groups, until we reach either the threshold value or the completely disjoint clustering. Such algorithms are called *divisive*.

Any problem involving classification of real data can not be reduced to applying a clustering algorithm alone. Before that, the data must be collected and consistently presented, and dissimilarity values must be assigned. After building the clusters, these groups are to be validated and assessed. The results have to be properly interpreted. All these are crucial issues, because any algorithm generates some clustering, but without further considerations we can not conclude whether these clusters reflect real structure of data or this is just an artifact of the algorithm. We leave out all these issues along with the problem of computer implementation and in this module consider only clustering algorithms.

In the module we discuss hierarchical algorithms for clustering discrete sets of data. These algorithms are based on the properties of a graph describing an initial collection of objects. Section 2 contains relevant graph-theoretical definitions — a reader familiar with graph theory can skip this section. In Section 3, using a simple model example, we develop a single-link hierarchical clustering algorithm. Section 4 is devoted to Hubert's single-link algorithm. In Section 5 we discuss a connection of the single-link hierarchical clustering with minimum spanning trees. Section 6 is devoted to another hierarchical clustering algorithm — Hubert's complete-link algorithm. In Section 7 we apply the single-link algorithm to a more realistic problem.

The *partition clustering* algorithms, based on the nearness of different objects, will be discussed in another module in this series. In the literature (see the list in the end of module) one can find other approaches to clustering. This module is aimed at undergraduate students studying discrete mathematics and/or statistics. The algorithms presented are simple and can be used in an introductory computer science course.

Acknowledgment. The author is extremely grateful to the DIMACS center at Rutgers University for their hospitality during the Reconnect '98 and Reconnect '99 conferences and to Professor Catherine McGeoch for her kindness and generous help.

2 DEFINITIONS

Our exposition uses basic concepts of graph theory, and here we remind our readers of some basic definitions. For a detailed exposition of graph theory see, for example, [11], Chap. 7 and 8. Intuitively, a graph G is a collection V of points called *vertices*. Some or all of these points are connected by arcs called *edges* — we denote the set of all edges by E. We call such drawings *undirected geometrical graphs* or just *graphs*[3] and denote them by $G = (V, E)$. An actual shape of the edges makes no difference in our considerations. There is only one essential issue — which pairs of vertices are connected and which pairs are not.

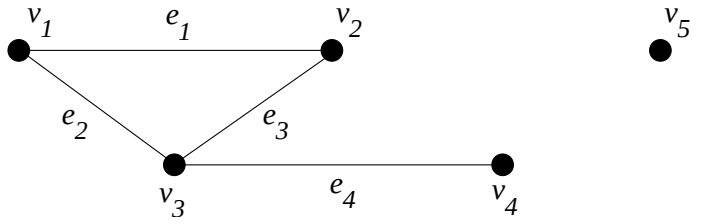


Figure 2.1: Graph G_1.

The graph G_1 (Figure 2.1) has five vertices labelled $v_1, \ldots, v_5$ and four edges labelled $e_1, \ldots, e_4$. The vertex v_5 is called *isolated*, because it is connected with *no* other vertex. Every edge has two corresponding vertices called its *endpoints*. Thus, the vertices v_1 and v_2 are the endpoints of the edge e_1. An edge connecting the vertices v_i and v_j is often denoted by $\{v_i, v_j\}$ or even by $e_{i,j}$. The edge is *incident* to each of its endpoints and vice versa. Two vertices are called *adjacent* if they are the endpoints of the same edge. Thus, v_1 and v_2 are adjacent vertices but v_1 and v_4 are not.

Definition 2.1 Given a graph $G = (V, E)$, any alternating sequence of its vertices and edges $P = (v_{i_1}, e_{i_1}, v_{i_2}, e_{i_2}, v_{i_3}, \ldots, e_{i_k}, v_{i_{k+1}})$, where $v_i \in V$ and $e_i \in E$, is called a *path* (of length k) in G, provided that in every consecutive triple $(v_{i_j}, e_{i_j}, v_{i_{j+1}})$ the vertices v_{i_j} and $v_{i_{j+1}}$ are the endpoints of the edge e_{i_j}. The path P *connects* vertices v_{i_1} and $v_{i_{k+1}}$. If $v_{i_1} = v_{i_{k+1}}$, the path is called a *closed path* or a *cycle*.

Exercise 4 Find two paths having length 3 in the graph G_1 (Figure 2.1).

Definition 2.2 A graph G is called *connected* if, for each pair of its vertices v_i and v_j, there exists a path connecting them.

Exercise 5 Are the graphs G_1 (above) and G_2 (see Figure 2.3 below) connected?

Definition 2.3 If every pair of vertices of a graph is adjacent, the graph is called *complete*. A complete graph with n vertices is commonly denoted by K_n.

Exercise 6 Draw complete graphs $K_1, \ldots, K_4$.

Exercise 7 Give an example of a connected but not complete graph. What is a minimal number of vertices in such graph?

[3]We consider only *simple* graphs, that is, graphs without loops and multiple edges.

Definition 2.4 A graph $G' = (V', E')$ is called a *subgraph* of a graph $G = (V, E)$, if $V' \subseteq V$ and $E' \subseteq E$. That is, if $e \in E'$ and v_i, v_j are its endpoints in G, then these vertices v_i, v_j must belong to G'.

Definition 2.5 A connected subgraph $G' = (V', E')$ of a graph $G = (V, E)$ is called a *connected component* of G if either $G' = G$ or no vertex in the set-difference $V \setminus V'$ (that is, outside V') is connected with any vertex in V'.

Exercise 8 Find all connected components in the graphs G_1 (Figure 2.1) and G_2 (Figure 2.3).

Definition 2.6 An edge e in a graph G is called a *cut-edge* or a *bridge* if its removal increases the number of connected components in the graph. For example, in the graph G_1 only the edge e_4 is a cut-edge.

In many problems, it is useful to assign a piece of additional information, numerical or otherwise, to some edges of a graph. This information may represent the length of a trip, or a potential flow through the pipe, or directions like "One-Way" signs in streets. A numerical or literal label is called the *weight* of an edge. If every edge of a graph has a weight, the graph is said to be a *weighted graph*. The total sum of the weights of all edges is called the *weight of the graph*.

Definition 2.7 A connected graph without cycles is called a *tree*.

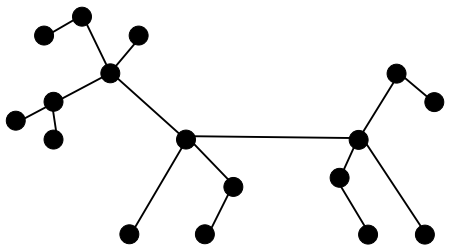


Figure 2.2: An example of a tree.

Exercise 9 Prove that every edge in a tree is a cut-edge.

Exercise 10 Prove that a tree with n vertices has $n - 1$ edges.

The graph G_2 (Figure 2.3) is actually a subgraph of G_1 above — we just removed the isolated vertex v_5. Compare the graph G_2 (Figure 2.3) and a graph $T = (\{v_1, v_2, v_3, v_4\}, \{e_1, e_3, e_4\})$ (Figure 2.4):

The graph T (Figure 2.4), in turn, is a subgraph of G_2 — it contains all vertices of G_2 and some of its edges. Moreover, T is a tree containing all vertices of G_2. Such a subgraph is called a spanning tree of G_2. We give a formal definition.

Definition 2.8 A subgraph $G' = (V', E')$ of a graph $G = (V, E)$ is called a *spanning tree* of G if G' is a tree and $V' = V$.

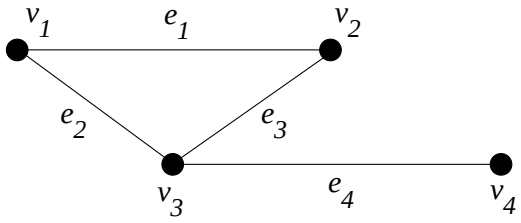


Figure 2.3: Graph G_2.

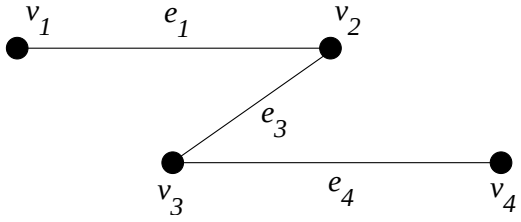


Figure 2.4: Graph T.

A graph may have several spanning trees. In a weighted graph, spanning trees may have different weights; a spanning tree with minimal total edge weight over all spanning trees is called a *minimum spanning tree*. Again, a graph might have several minimum spanning trees.

Exercise 11 Draw a graph having only one spanning tree.

Exercise 12 Find all spanning trees of the graph G_2.

Exercise 13 Find all minimum spanning trees in the graph G_3 (Figure 2.5):

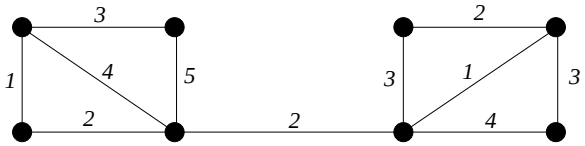


Figure 2.5: Graph G_3.

We also need a few definitions from set theory.

Definition 2.9 A family of sets $U = \{X_a, X_b, X_c, \ldots\}$ is called a *partition of a set* X if all sets $X_a, X_b, X_c, \ldots$ (called *terms of the partition*) are non-empty mutually disjoint subsets of X, whose union is equal to X; that is, if

1. $\emptyset \neq X_a \subseteq X$, $\emptyset \neq X_b \subseteq X$, ...
2. All pairwise intersections are empty: $X_a \cap X_b = \emptyset$, etc., and
3. $X_a \cup X_b \cup \ldots = X$.

Definition 2.10 A partition V of a set X is called *nested* in a partition U of X if every term of V is a subset of a term of U.

For example, a family of sets $U = \{X_1, X_2, X_3\}$, where $X_1 = \{a, b\}$, $X_2 = \{c\}$, and $X_3 = \{d, e, f\}$, forms a partition of the set $X = \{a, b, c, d, e, f\}$. A family $V = \{Y_1, Y_2, Y_3, Y_4, Y_5\}$, where $Y_1 = \{a\}$, $Y_2 = \{b\}$, $Y_3 = \{c\}$, $Y_4 = \{d\}$, and $Y_5 = \{e, f\}$, forms another partition of the same set $X = \{a, b, c, d, e, f\}$, which is nested into the partition U.

Exercise 14 (**a**) Prove that the set of whole numbers and the set of negative integer numbers form a partition of the set Z of all integers.

(**b**) Prove that the set of natural numbers, the set of negative integer numbers, and the one-element set $\{0\}$ form another partition of Z.

(**c**) Which one (if any) of the partitions in parts **a)** and **b)** is nested in another partition?

3 MODEL EXAMPLE

In this section, we consider the following model problem. A certain state has eight cities. Hereafter, we denote them by $c_1, c_2, \ldots, c_8$. The Government wants to connect all of them by highways. It is possible to build a highway connecting every pair of cities. In graph theory terms, such a road network can be described as the complete graph K_8. However, that project is very expensive. On the other hand, it is possible to link every city with only one or two other cities, thus having a minimal number of roads built. Even though it is less expensive to construct, this project (which can be modeled by a minimum spanning tree) is not good for future commuters, who will have to waste their time and fuel, because some pairs of cities do not have direct routs. Thus, a mathematician has offered an intermediate approach - he has suggested to split all the cities into several groups — clusters. The cities within each cluster are to be connected completely, but any two different clusters should be connected by only one road.

A cluster should, obviously, include the cities that are close to each other. The closeness here can be measured in various ways. The Government provides information about the average number of commuters in both directions between the cities. Let us, say, these amounts of commuters, in thousands of people per day, are 24 between the cities c_1 and c_2, 2 between c_1 and c_6 and 6 between c_2 and c_6. Thus, there is a large flow of commuters between c_1 and c_2 — in this sense these two cities are near to each other, even though they may be located far away from each other. So, they are similar and should be in one cluster. Yet, c_6 is distant from them. However, if we use these quantities — 24, 2, 6, etc., as a measure of closeness (a generalized distance), then the distance between nearby cities is greater than the distance between the distant ones.

In this problem and, as we have already mentioned, in clustering theory generally, it is more suitable to use the *dissimilarities* of objects rather than their similarities. We can convert the commuter data into dissimilarity values in some convenient way, perhaps by taking inverses or subtracting from some maximum value – see Table 3.1 below.

Definition 3.1 A square symmetric[4] matrix (table) with non-negative elements, whose main diagonal contains only zeros, is called a *dissimilarity matrix (table)*.

Table 3.1 below is a dissimilarity table for our model example. We consider the total amount of commuters in both directions, so the table is symmetrical with respect to the main diagonal and we fill in only its upper triangle. Moreover, since we want to start with a simple example, all the entries are different (the table contains no ties) and they are all natural numbers from 1 to $\binom{8}{2} = 28$.

The mathematician must now solve the problem of combining the cities into clusters according to this dissimilarity matrix. A procedure for building clusters is called a *clustering algorithm*. At the initial step, the algorithm treats each object under consideration as a single-element cluster. The set of these clusters is called the clustering of *level zero*. If we use graph-theory language, we can depict this clustering as a graph having only isolated vertices, without any edges.

Then, at every step, the algorithm uses only one edge to combine two closest (that is, with the smallest dissimilarity) clusters into a new one. Such an edge connecting two clusters of the same level into a cluster of the next level is called a *link*. That is why this and similar procedures are called *single-link clustering*, or *single linkage*. In this section we begin with a descriptive version of an agglomerative single-link clustering algorithm, apply it to the model example, and then give a formal treatment of the algorithm. In Section 4 we present a version of the algorithm known as Hubert's single-link algorithm [6, 7].

[4] In this module we do not consider non-symmetrical case.

	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8
c_1	0	5	10	7	22	27	25	13
c_2		0	8	12	28	23	17	6
c_3			0	1	9	19	3	26
c_4				0	4	14	2	21
c_5					0	11	16	18
c_6						0	15	20
c_7							0	24
c_8								0

Table 3.1: The dissimilarity table for the model example.

We denote the consecutive clusterings by boldface capital letters with one index $\mathbf{C}_0, \mathbf{C}_1, \mathbf{C}_2$, and so forth. The italic capital letters with double indices, C_{kl}, denote clusters — the first index, k, means the level of clustering and the second index, l, stands for the number of this particular cluster in the clustering of the k^{th} level. Thus, C_{34} denotes the fourth cluster in the third-level clustering $\mathbf{C}_3$.

Now we build clusterings for the model example. In our notations, $\{c_i, c_j\}$ is a pair (two-element set) comprising the i^{th} city c_i and j^{th} city c_j, and a number $d(c_i, c_j)$, or $d(i, j)$ for short, at the crossing of the i^{th} row and j^{th} column stands for the dissimilarity of these two cities; due to the symmetry, $d(i, j) = d(j, i)$. First, we rearrange all pairs of the cities in ascending order of their dissimilarities (Table 3.2).

At the initial step, we form a disjoint clustering, such that each city forms its own cluster containing only one element. This is $\mathbf{C}_0$ — the clustering of level zero:

$$\mathbf{C}_0 = \{C_{01}, C_{02}, C_{03}, C_{04}, C_{05}, C_{06}, C_{07}, C_{08}\} \text{ where } C_{0i} = \{c_i\}, i = 1, \ldots, 8.$$

The dissimilarity $diss(C_{0i}, C_{0j})$ between two clusters of level zero is defined to be the dissimilarity between the corresponding cities, that is, $diss(C_{0i}, C_{0j}) = d(c_i, c_j)$.

It is helpful to visualize the process of clustering by drawing graphs of special kind, called threshold graphs.

Definition 3.2 Given a dissimilarity matrix of size n and a nonnegative number v, the *threshold graph* $G(v)$ is a (simple) weighted graph with n vertices corresponding to n entities under consideration, such that two vertices v_i and v_j are adjacent if and only if $d(c_i, c_j) \leq v$. The weight of an edge $e_{i,j}$ connecting two vertices v_i and v_j is the dissimilarity $d(c_i, c_j)$.

Pair $\{c_i, c_j\}$	Dissimilarity $d(i,j)$
$\{c_3, c_4\}$	1
$\{c_4, c_7\}$	2
$\{c_3, c_7\}$	3
$\{c_4, c_5\}$	4
$\{c_1, c_2\}$	5
$\{c_2, c_8\}$	6
$\{c_1, c_4\}$	7
$\{c_2, c_3\}$	8
$\{c_3, c_5\}$	9
$\{c_1, c_3\}$	10
$\{c_5, c_6\}$	11
$\{c_2, c_4\}$	12
$\{c_1, c_8\}$	13
$\{c_4, c_6\}$	14
$\{c_6, c_7\}$	15
$\{c_5, c_7\}$	16
$\{c_2, c_7\}$	17
$\{c_5, c_8\}$	18
$\{c_3, c_6\}$	19
$\{c_6, c_8\}$	20
$\{c_4, c_8\}$	21
$\{c_1, c_5\}$	22
$\{c_2, c_6\}$	23
$\{c_7, c_8\}$	24
$\{c_1, c_7\}$	25
$\{c_3, c_8\}$	26
$\{c_1, c_6\}$	27
$\{c_2, c_5\}$	28

Table 3.2: The same dissimilarity table (Table 3.1) rearranged in ascending level of the dissimilarities.

So, two vertices are adjacent in $G(0)$ if and only if their dissimilarity is zero; if there is no such a pair of vertices, $G(0)$ contains only n isolated vertices and no edge. If a threshold value v is greater than or equal to the largest entry of the dissimilarity matrix, we get a complete threshold graph and denote it by $G(\infty)$.

The smallest dissimilarity in the problem is $d(3,4) = 1$. So that, if the threshold value (an acceptable level of dissimilarity) is less than 1, we can not combine any two cities in one cluster and have to stop here. In terms of our model example, that means that no cluster has an infrastructure, and we have to build a road between all pairs of cities (Figure 3.2).

Suppose, the threshold is at least 1. Then we have to consider all 28 pairwise unions $C_{01} \cup C_{02}, C_{01} \cup C_{03}, \ldots, C_{01} \cup C_{08}, C_{02} \cup C_{03}, \ldots, C_{02} \cup C_{08}, \ldots., C_{07} \cup C_{08}$. In a corresponding graph (the same Figure 3.2) its 28 edges $\{1,2\}$, $\{1,3\}$,..., $\{1,8\}$, $\{2,3\}$, ..., $\{7,8\}$ with weights $d(1,2), \ldots, d(7,8)$ correspond to these 28 unions.

Since the lowest weight is $d(c_3, c_4) = 1$, the clusters C_{03} and C_{04} have the smallest dissimilarity

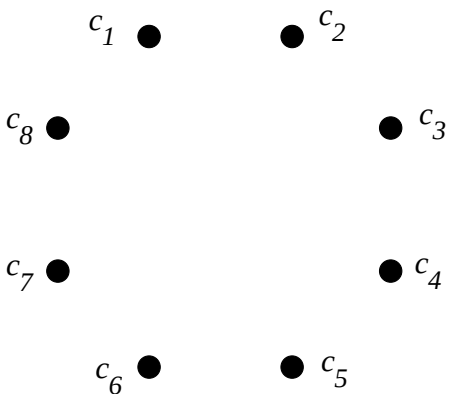


Figure 3.1: The threshold graph $G(0)$ for the model example. It corresponds to the $\mathbf{C}_0$-clustering.

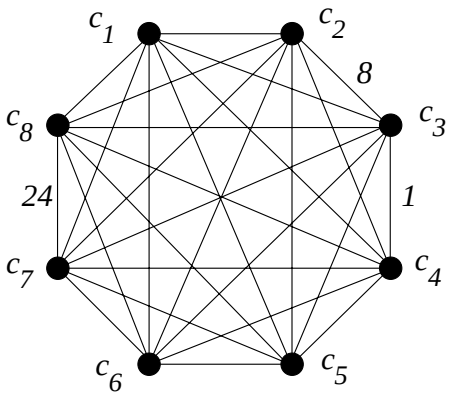


Figure 3.2: The complete graph $G(\infty)$ for the model example; only a few weights (dissimilarities) are shown.

$diss(C_{03}, C_{04}) = d(c_3, c_4) = 1$. In terms of our problem they have the largest flow of commuters between them. Thus, we have to connect them first, and we amalgamate these two clusters of level zero in one cluster of the first level. All the other clusters of level zero automatically become clusters of the first level. This way, we get the first-level clustering $\mathbf{C}_1$:

$$\mathbf{C}_1 = \{C_{11}, C_{12}, C_{13}, C_{14}, C_{15}, C_{16}, C_{17}\},$$

where $C_{11} = C_{03} \cup C_{04} = \{c_3, c_4\}, C_{1,i} = C_{0,i-1} = \{c_{i-1}\}$ for $i = 2, 3$, and $C_{1i} = C_{0,i+1} = \{c_{i+1}\}$ for $i = 4, \ldots, 7$. This clustering is shown in Figure 3.3 where the connected component with the vertices $\{c_3, c_4\}$ corresponds to the cluster C_{11}. The dissimilarities between the clusters $C_{12}, \ldots, C_{17}$ are the same as those between the corresponding "old" clusters of level zero. The dissimilarity between C_{11} and any cluster $\{c_i\}, i = 1, 2, 5, \ldots, 8$, is the smaller of $d(c_3, c_i)$ and $d(c_4, c_i)$. For instance, $diss(C_{11}, C_{14}) = \min\{d(c_3, c_5); d(c_4, c_5)\} = \min\{9; 4\} = 4$.

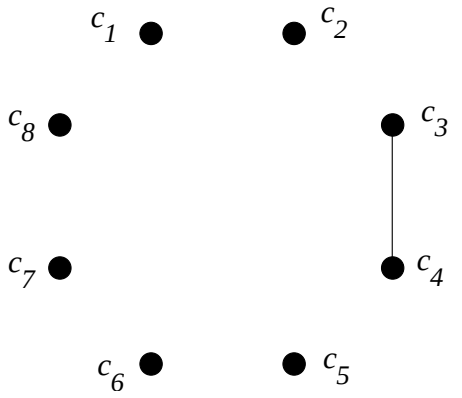


Figure 3.3: The threshold graph $G(1)$ corresponds to the first-level clustering $\mathbf{C}_1$ — only two vertices are connected.

The clustering in Figure 3.3 contains one two-element cluster $\{c_3, c_4\}$ and 6 one-element clusters $\{c_1\}, \{c_2\}, \{c_5\}, \{c_6\}, \{c_7\}, \{c_8\}$.

It should be repeated that while building $\mathbf{C}_1$ from $\mathbf{C}_0$, we have used only one link — in the sense that the threshold graph corresponding to $\mathbf{C}_1$ contains just one new link, one more edge than the graph corresponding to $\mathbf{C}_0$. All the 28 pairs of vertices $\{C_{01}, C_{02}\}, \ldots, \{C_{07}, C_{08}\}$, each pair taken together with the incident edge, represent connected two-vertex subgraphs of the graph in Figure 3.2 — we have selected among them a subgraph with the minimal weight and linked two vertices of this chosen subgraph into a cluster. Again in terms of our model, we have to build a road between c_3 and c_4. This "in-cluster" road is shown in bold in Figure 3.4. Then we have to connect each other city with either c_3 or c_4, but not with both, using "between-clusters" roads.

Given two clusters, $C_{11} = \{c_3, c_4\}$ and $C_{1i} = \{c_i\}, i = 1, 2, 5, \ldots, 8$, the decision on what city (c_3 or c_4) to connect with c_i, is based on the dissimilarity between the clusters C_{11} and C_{1i} . For example, since $diss(C_{11}, C_{14}) = \min\{d(c_3, c_5); d(c_4, c_5)\} = d(c_4, c_5) = 4$, the cluster $\{c_5\}$ is to be connected with c_4. A corresponding road map may look like one in Figure 3.4.

If the threshold is 1, we should stop here. However, if we can accept a larger threshold, we are to continue. To build a second-level clustering, we proceed in the same way. Namely, we consider all pairs of the first-level clusters and look for a connecting link with the smallest dissimilarity. The edge $\{c_3, c_4\}$, which had been utilized before, may not be used again. Among the unused edges, the smallest dissimilarity is $d(c_4, c_7) = 2$, and we form the second-level cluster C_{21} as a set — union of

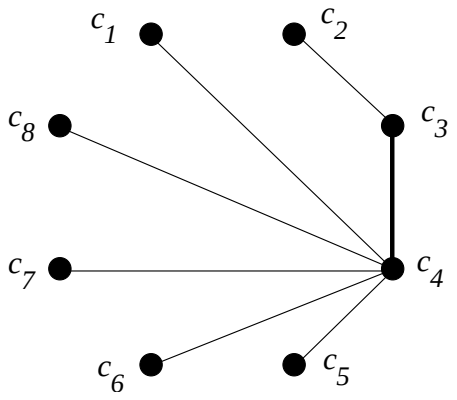


Figure 3.4: A road map corresponding to the clustering $\mathbf{C}_1 = \{C_{11}, C_{12}, C_{13}, C_{14}, C_{15}, C_{16}, C_{17}\}$. Keep in mind that the "between-clusters" roads reflect the dissimilarities, not the real distances between the cities.

two first-level clusters containing the cities c_4 and c_7. To form this cluster, we have used a single link — the edge $\{c_4, c_7\}$. All the other first-level clusters move into the second-level clustering $\mathbf{C}_2$ unchanged, after just renumbering (Figure 3.5):

$$\mathbf{C}_2 = \{C_{21}, C_{22}, C_{23}, C_{24}, C_{25}, C_{26}\}$$

where $C_{21} = C_{11} \cup C_{16} = \{c_3, c_4, c_7\}, C_{2i} = C_{1i}, i = 2, \ldots, 5,$ and $C_{26} = C_{17}$.

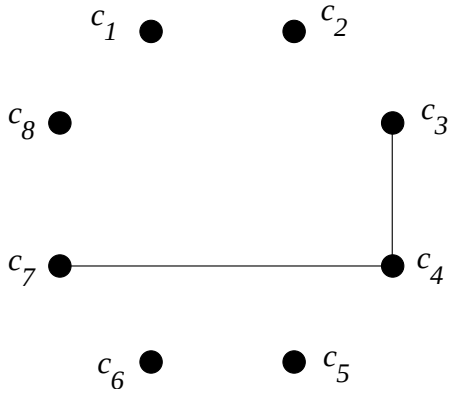


Figure 3.5: The threshold graph $G(2)$ corresponds to $\mathbf{C}_2$ — one more edge is added; $\mathbf{C}_2$ consists of one three-element cluster $\{c_3, c_4, c_7\}$ and five one-element ones $\{c_1\}$, $\{c_2\}$, $\{c_5\}$, $\{c_6\}$, $\{c_8\}$.

This clustering corresponds to the threshold value of 2. It is worth noting that the dissimilarity $d(3, 7)$ between the objects c_3 and c_7 in the cluster C_{23}, is greater than 2, but there are edges in the cluster, connecting these vertices, namely, $\{c_3, c_4\}$ and $\{c_4, c_7\}$, such that their weights do not exceed the threshold value. This is an important feature of single-link methods: *for any two objects x and y in a cluster there exists a sequence of objects in this cluster connecting x and y, such that the dissimilarity of any two neighbors in this sequence does not exceed the threshold value, even though the dissimilarity of x and y may be greater than the threshold.*

We continue building a hierarchical clustering for our model example. Suppose we can accept a value of the threshold greater than 2. The next unused dissimilarity $d(c_3, c_7)$ gives nothing new,

because the cities c_3 and c_7 have been already connected in a cluster. Therefore, $d(c_3, c_7)$ *does not generate* a next clustering (Figure 3.6).

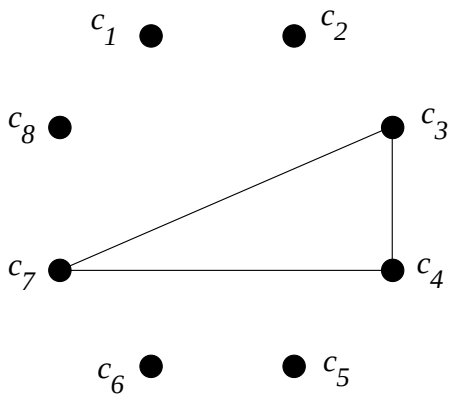


Figure 3.6: The threshold graph $G(3)$ does not generate a new clustering.

Thus, we skip $d(c_3, c_7)$ and use the next dissimilarity $d(c_4, c_5) = 4$. Therefore, the next clustering, corresponding to the threshold value of 4, is $\mathbf{C}_3 = \{\{c_3, c_4, c_5, c_7\}, \{c_1\}, \{c_2\}, \{c_6\}, \{c_8\}\}$. Five sets in $\mathbf{C}_3$ represent all five clusters of the third level (Figure 3.7). Again, the dissimilarity between some vertices in the first cluster C_{31} is greater than 4, but for any two vertices there exists a connecting path such that every edge in the path has a weight (dissimilarity) of not more than 4. In formal terms, we consider all the unions $C_{2a} \cup C_{2b}$ formed by a single edge and look for the link with the smallest weight, which generates a new cluster.

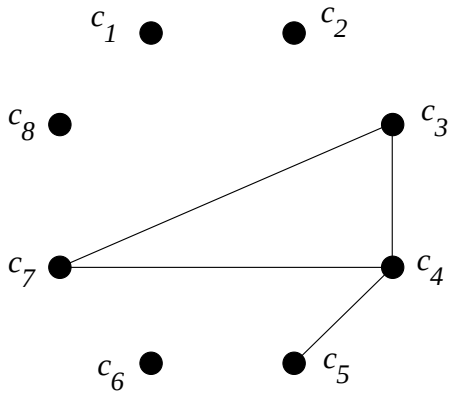


Figure 3.7: The threshold graph $G(4)$ contains one new edge. It corresponds to $\mathbf{C}_3$-clustering containing one four-element cluster $\{c_3, c_4, c_5, c_7\}$ and four one-element ones $\{c_1\}$, $\{c_2\}$, $\{c_6\}$, $\{c_8\}$.

The next smallest weight to use is $d(1, 2) = 5$, and if we are willing to continue and use this value of the threshold, we have to combine $\{c_1\}$ and $\{c_2\}$, that is, $\mathbf{C}_4 = \{\{c_1, c_2\}, \{c_3, c_4, c_5, c_7\}, \{c_6\}, \{c_8\}\}$.

A road map corresponding to the fourth-level clustering $\mathbf{C}_4$ is shown in Figure 3.9.

This way, we construct the hierarchy of consecutive clusterings, corresponding to increasing values of the threshold. It is now the turn of $d(2, 8) = 6$, and the fifth-level clustering is $\mathbf{C}_5 =$

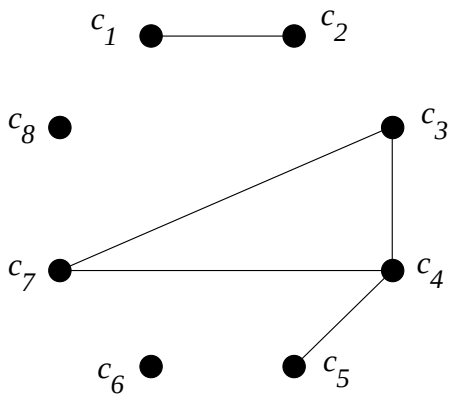


Figure 3.8: The threshold graph $G(5)$ generates the fourth-level clustering C_4 consisting of one four-element cluster $\{c_3, c_4, c_5, c_7\}$, one two-element $\{c_1, c_2\}$, and two one-element ones $\{c_6\}$, $\{c_8\}$.

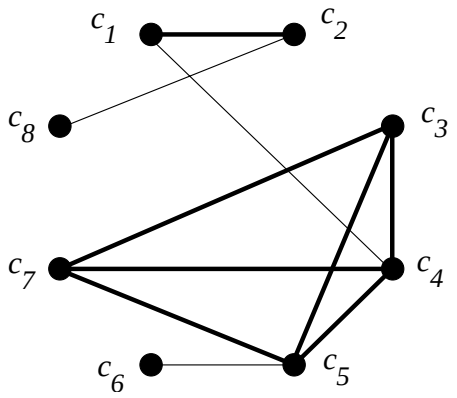


Figure 3.9: A road map corresponding to the clustering $\mathbf{C}_4 = \{\{c_1, c_2\}, \{c_3, c_4, c_5, c_7\}, \{c_6\}, \{c_8\}\}$. Two first clusters are connected by the edge $\{c_1, c_4\}$, because this edge has the smallest dissimilarity among the edges connecting the two clusters.

$\{\{c_1, c_2, c_8\}, \{c_3, c_4, c_5, c_7\}, \{c_6\}\}$.

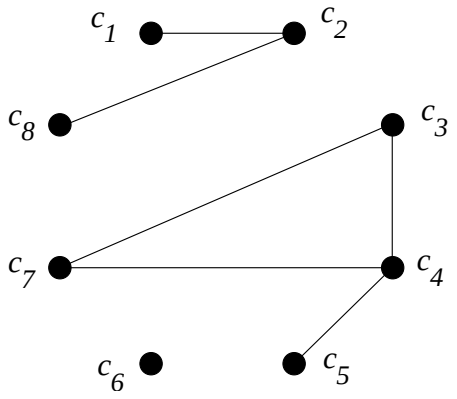


Figure 3.10: The threshold graph $G(6)$ generates the fifth-level clustering $\mathbf{C}_5$, which contains one four-element cluster $\{c_3, c_4, c_5, c_7\}$, one three-element cluster $\{c_1, c_2, c_8\}$, and a one-element cluster $\{c_6\}$.

The next unused edge with the lowest weight is $\{c_1, c_4\}$ with $d(1, 4) = 7$ and we come up with the next clustering $\mathbf{C}_6 = \{\{c_1, c_2, c_3, c_4, c_5, c_7, c_8\}, \{c_6\}\}$ (Figure 3.11).

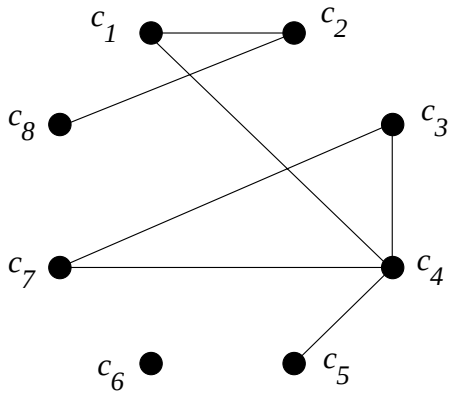


Figure 3.11: The threshold graph $G(7)$ generates the sixth-level clustering $\mathbf{C}_6$ containing one seven-element and one one-element clusters $\{c_1, c_2, c_3, c_4, c_5, c_7, c_8\}$ and $\{c_6\}$.

The edges with weights 8, 9, and 10 generate no new clusters. Finally, using $d(5, 6) = 11$ we get the *one-cluster* clustering (Figure 3.12), which is called *conjoint*: $\mathbf{C}_7 = \{C_{7,1}\}$, where $C_{7,1} = \{c_1, c_2, c_3, c_4, c_5, c_6, c_7, c_8\}$.

It is worth noting that in terms of our model, both $\mathbf{C}_0$ and $\mathbf{C}_7$ result in the same road network as shown in Figure 3.2.

Having done the job, the mathematician made a presentation, showing possible road networks corresponding to all consecutive levels of clustering. In particular, he showed Figures 3.2, 3.4, and 3.9. A road map corresponding to the clustering $\mathbf{C}_4$ (Figure 3.9) was actually accepted as a plan for a future construction.

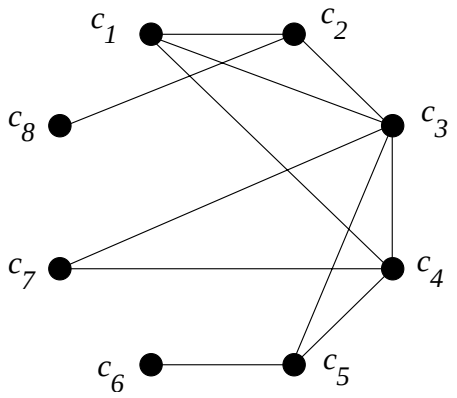


Figure 3.12: The threshold graph $G(11)$; it generates the conjoint clustering $\mathbf{C}_7 = \{C_{7,1}\}$.

Exercise 15 Draw road maps corresponding all other levels of clustering $(\mathbf{C}_2, \mathbf{C}_3, \mathbf{C}_5, \mathbf{C}_6)$ in the model example.

We see that the algorithm above can be stated in the following formal way suitable for computer realization, which is called a pseudocode form.

AGGLOMERATIVE SINGLE-LINK ALGORITHM

Given a set of n objects $X = \{x_1, x_2, \ldots, x_n\}$, their dissimilarity table, and a threshold value v.

1. Rearrange the dissimilarity table in ascending order.
2. Set $m = 0$ and form a completely disjoint clustering of zero level $\mathbf{C}_0 = \{C_{01}, C_{02}, \ldots, C_{0n}\}$ where $C_{0i} = \{x_i\}, i = 1, \ldots, n$.
3. Set $m := m + 1$ and consider the first unused entry in the dissimilarity table. Let it be, say, $d(x_k, x_l)$. If $d(x_k, x_l) > v$, stop. Otherwise, there are two possibilities.

A) The set $\{x_k, x_l\}$ is a subset of an existing cluster. Then skip $d(x_k, x_l)$ and return to step 3 (increase m).

B) The objects x_k and x_l belong to different existing clusters, say $x_k \in C_{m-1,a}$ and $x_l \in C_{m-1,b}$. Form a cluster of the m^{th} level as a union $c_{m,1} = C_{m-1,a} \cup C_{m-1,b}$, renumerate all the other clusters of the $(m-1)^{\text{th}}$ level to the m^{th} level unchanged, and return to step 3 (increase m).

$\diamondsuit$

Remark 3.3 We can get the conjoint clustering before we achieve the threshold level.

Remark 3.4 Given n objects, there are n levels of clustering — $\mathbf{C}_0, \ldots, \mathbf{C}_{n-1}$, where the last one is the conjoint clustering. Moreover, as far as the dissimilarity table contains $n(n-1)/2$ entries, there are no more than $n(n-1)/2+1$ threshold graphs (exactly $n(n-1)/2+1$ if there are no ties).

Remark 3.5 Since we look only for disjoint clusters, a clustering of any level is just a certain partition of the initial set of objects. Therefore, our algorithm generates a family of nested partitions of the given set. Moreover, we know (see, for example, [11], p. 412) that every partition of a set generates an *equivalence relation* on this set and vice versa. This relationship is dealt with in the next exercise.

Exercise 16 Describe explicitly the equivalence relations corresponding to the partitions of the set $C = \{c_1, c_2, c_3, c_4, c_5, c_6, c_7, c_8\}$ generated by the clusterings $\mathbf{C}_0, \ldots, \mathbf{C}_7$.

Exercise 17 Construct dissimilarity tables for a set with n elements such that there are exactly $2, 3, \ldots, n(n-1)/2 + 1$ threshold graphs.

Exercise 18 Change the $\{c_3, c_7\}$ entry in Table 3.2 to 2 and $\{c_2, c_8\}$ entry to 7, respectively, so that a new table contains ties. Apply the algorithm of this section to this new table and compare the resulting clusterings.

Exercise 19 Using the algorithm above, build all consecutive clusterings of the set $X = \{x_1, x_2, \ldots, x_6\}$, given the dissimilarity table 3.3 below. What level of clustering corresponds to the threshold level of 3? Of 2?

	x_1	x_2	x_3	x_4	x_5	x_6
x_1	0	6	8	3	4	8
x_2		0	2	4	1	5
x_3			0	6	2	3
x_4				0	9	2
x_5					0	4
x_6						0

Table 3.3: The dissimilarity table for Exercise 19.

4 HUBERT'S SINGLE-LINK ALGORITHM

In the preceding section we considered an agglomerative single-link algorithm and applied it to the model example. Each cluster is a set of vertices, so that a cluster generates a subgraph of the complete graph $G(\infty)$ corresponding to the initial set of entities. Vice versa, each connected subgraph of this complete graph can be viewed as a cluster consisting of the vertices of this subgraph. Therefore, we will freely interchange, on one hand, the language of objects and their collections — clusters — and, on the other hand, the language of vertices, graphs, and subgraphs. Hubert [6] gave versions of a single-link algorithm and a complete-link algorithm based on the concept of a threshold graph. In this section we give a formal treatment of Hubert's single-link algorithm [6, 7]. This algorithm leads to the same clustering as the agglomerative algorithm of the preceding section. However, we present it here to introduce some important notations used in discussing more difficult complete-link algorithm in the next section. First of all, some more designations are in order.

As always, we denote the clustering of the m^{th} level by $\mathbf{C}_m = \{C_{m1}, C_{m2}, \ldots, C_{m,n(m)}\}$, $m = 0, 1, 2, \ldots$, where $n(m)$ stands for the number of clusters contained in $\mathbf{C}_m$. In particular, $n(0) = n$. After $\mathbf{C}_m$ has been generated, we consider all pairwise unions $C_{ma} \cup C_{mb}$, $a, b = 1, \ldots, n(m), a \neq b$; there are $n(m)(n(m)-1)/2$ of these unions. A union $C_{ma} \cup C_{mb}$ contains some objects, say, $x_i, \ldots, x_j$. Given a union $C_{ma} \cup C_{mb}$, we can form several *connected* subgraphs of the threshold graph $G(v)$ spanned on these vertices $x_i, \ldots, x_j$. Namely, to make up such a subgraph from two clusters C_{ma} and C_{mb} , we consider all possible connections of a vertex from C_{ma} with a vertex from C_{mb} using only one edge. Let $S_m(a, b) = \min\{d(x_i, x_j) | x_i \in C_{ma}, x_j \in C_{mb}\}$ stand for the smallest dissimilarity between a vertex in C_{ma} and a vertex in C_{mb}. If the initial dissimilarity matrix contains ties, there may be several edges with the minimal weight — we can select any one of them. At every step we decrease the number of clusters by 1.

The function $S_m = S_m(a, b)$ is defined on all pairs of clusters $\{C_{ma}, C_{mb}\}$ of the m^{th} level. The letter "S" in $S_m(a, b)$ stands for "Single-linkage". Since we only consider finite sets, this function attains its minimum value on a certain pair of clusters, C_{mp} and C_{mq}. Let us denote this minimum value over all pairs of indices $\{a, b\}$ by $\min_{a,b}\{S_m(a, b)\} = S_m(p, q)$ and also denote the dissimilarity of the edge that links these clusters C_{mp} and C_{mq} by $d_{\min}(p, q)$.

Next we present Hubert's single-link algorithm in standard pseudo-code notation.

HUBERT'S SINGLE-LINK ALGORITHM

Given a set of n objects $X = \{x_1, x_2, \ldots, x_n\}$, the dissimilarity table, and a threshold value v.

1. Set $m = 0$ and form the disjoint clustering of zero level:

   $$\mathbf{C}_0 = \{C_{01}, C_{02}, \ldots, C_{0n}\}$$

   consisting of n one-element clusters $C_{0k} = \{x_k\}$, $k = 1, \ldots, n$. Define *the dissimilarities between the clusters* of level zero as $diss(C_{0i}, C_{0j}) = d(x_i, x_j)$.

2. Set $m := m + 1$, calculate the values $S_{m-1}(a, b) = \min\{d(x_i, x_j) | x_i \in C_{m-1,a}, x_j \in C_{m-1,b}\}$ for all pairs of indices $\{a, b\}, a \neq b$, and also find their minimum value $\min_{a,b}\{S_{m-1}(a, b)\} = S_{m-1}(p, q)$. To form the next clustering $\mathbf{C}_m$, we merge those two clusters $C_{m-1,p}$ and $C_{m-1,q}$, whose second indices are p and q, into a cluster $C_{m,1} = C_{m-1,p} \cup C_{m-1,q}$ by making use of an edge with the weight $d_{\min}(p, q)$. If there are ties, that is, there exist several edges with the

same weight $d_{\min}(p,q)$, we can use any one of them. All the other clusters of the $(m-1)^{\text{th}}$ level become the clusters of level m without changes but just renumbering.

3. Update the dissimilarity table as follows: The dissimilarity between every two "old" clusters (promoted from the preceding level) remains the same. The dissimilarity between $C_{m,1}$ and any cluster $C_{m-1,r}$, $r \neq p, r \neq q$, is the smaller of $diss(C_{m-1,p}, C_{m-1,r})$ and $diss(C_{m-1,q}, C_{m-1,r})$.

4. Continue until we reach the threshold value v or all the objects are merged into one conjoint cluster, whichever occurs first.

$\diamondsuit$

Remark 4.1 As we see in the example, not every threshold graph generates a new clustering.

Remark 4.2 It should be noted that finding a next clustering involves calculating the double minimum value

$$S_m(p,q) = \min_{a,b}\{S_m(a,b)\} = \min_{a,b}\{min\{d(x_i,x_j)|x_i \in C_{ma}, x_j \in C_{mb}\}\}.$$

We illustrate this algorithm on the same model example from preceding section. So that, in the rest of this section we denote the objects by c_i. The algorithm starts with single-element clusters corresponding to each city $c_1, \ldots, c_8$. That is, we set $m = 0$, and form the disjoint clustering $\mathbf{C}_0 = \{C_{01}, C_{02}, \ldots, C_{0n}\}$, where $C_{0k} = \{c_k\}, k = 1, \ldots, n$. This clustering corresponds to a subgraph of the graph $G(\infty)$ with no edges — every vertex is an isolated one.

Next, set $m = 1$. To every union $C_{0a} \cup C_{0b}$, there corresponds a unique connected subgraph of $G(\infty)$; this subgraph contains two vertices and their incident edge. Therefore, at this step $d_{\min} = d(c_3, c_4)$, $p = 3, q = 4$, and $S_0(p,q) = 1$. Thus, we have to combine the clusters C_{03} and C_{04} in a cluster C_{11} of the first level, promote all other zero-level clusters to the first level, and update the dissimilarity table. For example, since $C_{12} = \{c_1\}$, we get $diss(C_{11}, C_{12}) = \min\{diss(C_{0,1}, C_{0,3}); diss(C_{0,1}C_{0,4})\} = \min\{d(c_1,c_3); d(c_1,c_4)\} = \min\{10; 7\} = 7$.

Now, set $m = 2$. In addition to the same two-vertex subgraphs with vertices other than c_3 and c_4 that were considered before, we have to look for connected subgraphs with three vertices. Namely, we consider the subgraphs, which contain c_3 and c_4, their common edge, one other vertex, and an edge connecting the latter with either c_3 or c_4. The minimal dissimilarity is now $S_1(4,7) = 2$ and we have to connect clusters $\{c_7\}$ and $\{c_3, c_4\}$ in a cluster of the second level.

At the next step set $m = 3$. In the threshold graph $G(2)$, there are one three-element and five one-element clusters. The smallest unused dissimilarity is $d(3,7) = 3$, but adding the corresponding edge to $G(2)$ does not create a new cluster. Therefore, we have to leave out $d(3,7)$ and proceed to $d(4,5) = 4$. This way, we build on the clusterings of all higher levels, up to $\mathbf{C}_7$.

When amalgamating the clusters, step by step, we are increasing the threshold value and respectively, we are generating the threshold graphs. The latter were drawn in section 3 (Figures 3.1 .. 3.3, 3.5 .. 3.8, 3.10 .. 3.12).

To visualize the process of clustering, a special kind of tree-like graph is also used. These graphs are called *dendrograms*. Below we build the single-link dendrogram corresponding to our

model problem. It is clear from this example how to build a dendrogram for any problem. It should be noted that different horizontal levels of the dendrogram, going down, correspond to consecutive clusterings in the problem.

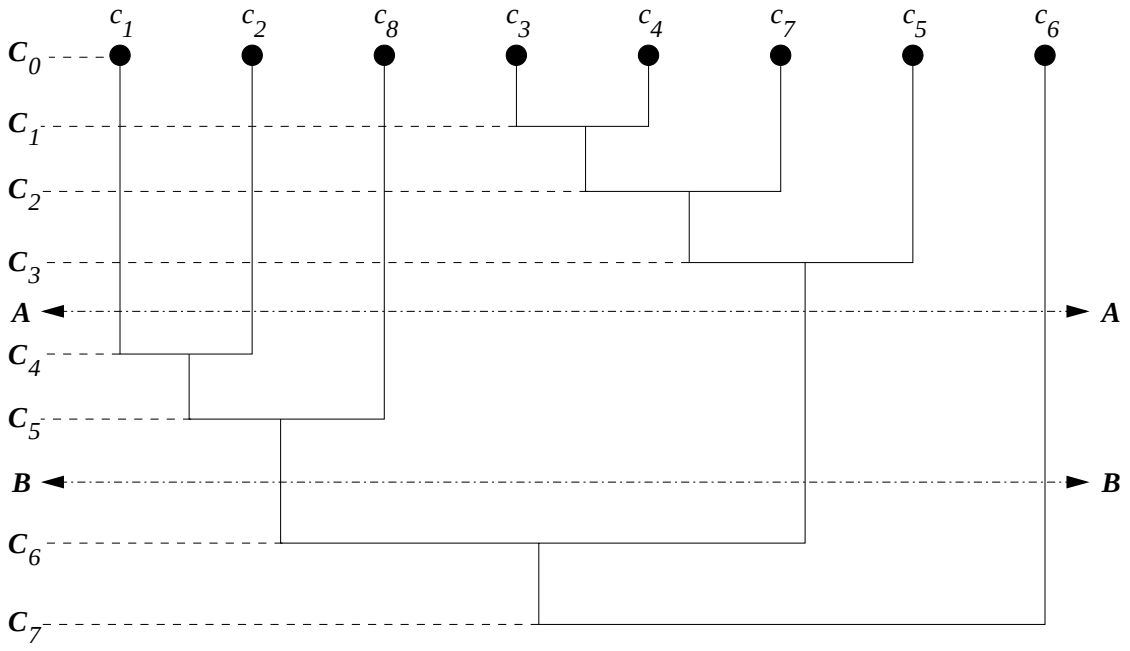


Figure 4.1: Dendrogram corresponding to the model example.

Any cutting of a dendrogram generates a clustering. Thus, cutting at a level $\mathbf{A} \longleftrightarrow \mathbf{A}$ gives the clustering $\mathbf{C}_3 = \{\{c_1\}, \{c_2\}, \{c_8\}, \{c_3, c_4, c_7, c_5\}, \{c_6\}\}$, sectioning at a level $\mathbf{B} \longleftrightarrow \mathbf{B}$ generates the clustering $\mathbf{C}_5 = \{\{c_1, c_2, c_8\}, \{c_3, c_4, c_5, c_7\}, \{c_6\}\}$.

Exercise 20 Using Hubert's single-link algorithm, build all consecutive threshold graphs and clusterings of the set $X = \{x_1, x_2, \ldots, x_6\}$, given the dissimilarity Table 3.3. What levels of clustering correspond to the threshold levels of 2? Of 3? Of 4?

Exercise 21 Apply Hubert's single-link algorithm to the dissimilarity table of Exercise 18 and compare the results with the clusterings obtained in that exercise.

5 SPANNING TREES

Consider a connected weighted graph G_4 (Figure 5.1), where $w_1 = 2$, $w_2 = 5$, $w_3 = 1$, $w_4 = 3$:

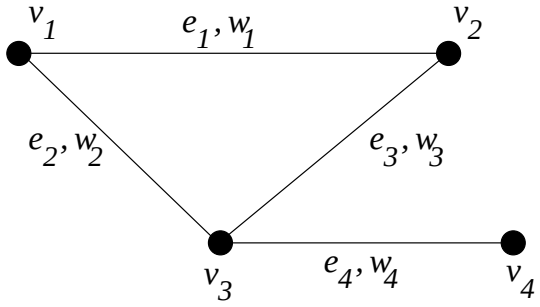


Figure 5.1: Graph G_4.

The graph in Figure 5.1 has three weighted spanning trees:

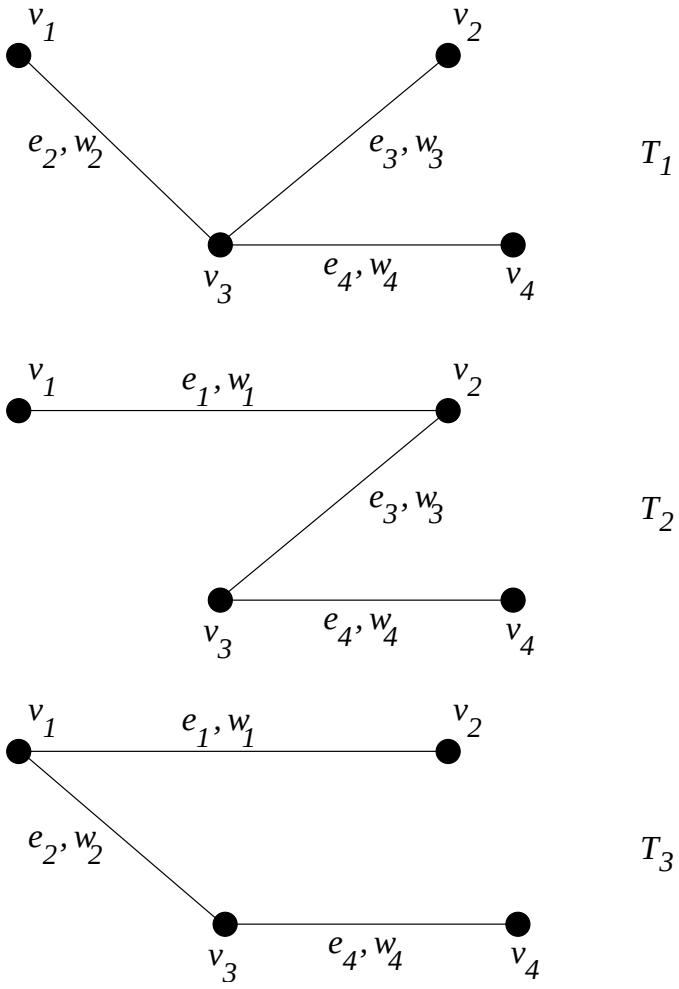


Figure 5.2: Spanning trees T_1, T_2, T_3.

These trees have different weights: $W(T_1) = w_2 + w_3 + w_4 = 9$, $W(T_2) = w_1 + w_3 + w_4 = 6$,

$W(T_3) = w_1 + w_2 + w_4 = 10$, with T_2 having the smallest weight — it is the minimum spanning tree of the graph G_4.

Clustering algorithms can be developed in terms of spanning trees. To discuss this connection, we first present here Kruskal's algorithm for finding a minimum spanning tree in a connected graph. Connectedness is, obviously, a necessary condition for a graph to have a spanning tree.

KRUSKAL'S ALGORITHM

Given a connected weighted graph $G(V, E)$ with n vertices. We assume that all weights are nonnegative numbers.

1. Select an edge e with a minimum weight. If a graph has several edges with equal weights, we can choose any of them. The edge e and its end-points form an initial subgraph T_1 of G.

2. For $m = 1, 2, \ldots, n-2$, select an unused edge with the smallest weight such that this edge does not form a cycle with the edges selected earlier. In particular, we can use an edge with the same weight as the one in the previous step. Add the edge chosen and its end-points, if necessary, to the subgraph T_m generated at the previous step, to generate the next subgraph T_{m+1}.

3. The subtree T_{n+1} is a minimum spanning tree in G.

♢

Remark 5.1 The condition of not forming cycles in Step 2 is in a sense equivalent to the restriction in a single-link algorithm, which forbids using an edge connecting two vertices that already belong to the same cluster.

Remark 5.2 Not every graph among $T_2, \ldots, T_{n-2}$ must be a tree, but each has a property that every connected component is a tree; such graphs are (obviously) called *forests*.

Exercise 22 Prove that the algorithm generates a minimum spanning tree in any connected graph.

Now we build a minimum spanning tree for the basic graph (Figure 3.2) in the model example. The next figures exhibit all steps of this process.

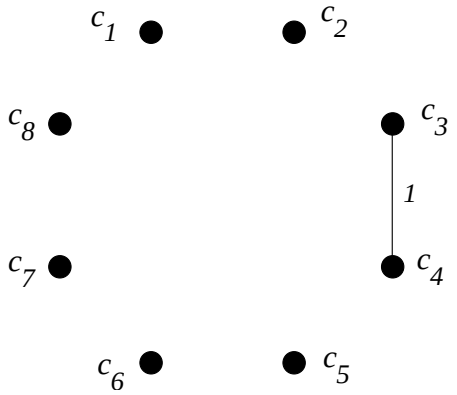


Figure 5.3: Step One. The first subtree T_1 with only one edge is formed. Its two vertices $\{c_3, c_4\}$ together with all other isolated vertices precisely correspond to the first level clustering $\mathbf{C}_1$.

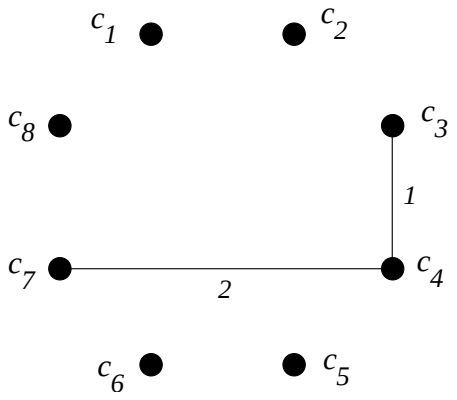


Figure 5.4: Step Two. The second edge is added. This drawing corresponds to the subtree T_1 and to the clustering $\mathbf{C}_2$.

The next smallest weight is $d(3,7) = 3$. However, we can not choose the edge $\{c_3, c_7\}$ because it forms a cycle with the previously selected edges $\{c_3, c_4\}$ and $\{c_4, c_7\}$. It should be noted that when we worked out Hubert's single-link algorithm in Section 4, we also could not choose the edge $\{c_3, c_7\}$ since it had not set up a new cluster. Thus, the next subgraphs are:

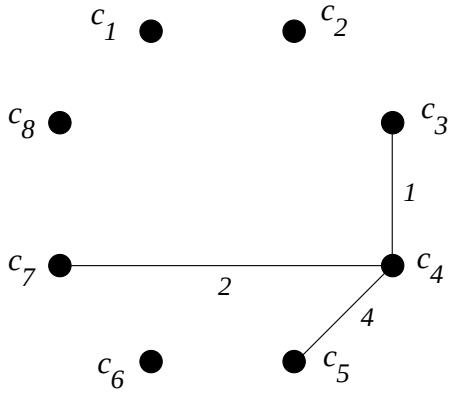


Figure 5.5: Step Three. This subtree with three edges corresponds to the clustering $\mathbf{C}_3$.

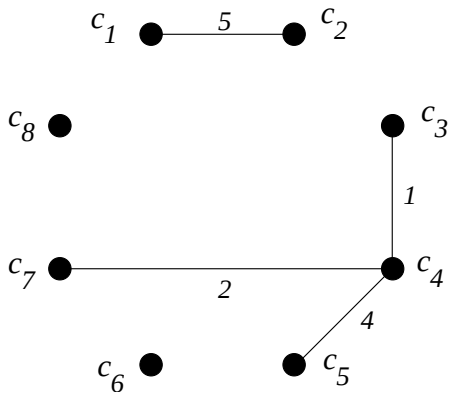


Figure 5.6: Step Four. The subgraph T_4 is not a tree, since it is not connected. It consists of two subtrees — with three edges and with one edge. This drawing corresponds to the clustering $\mathbf{C}_4$.

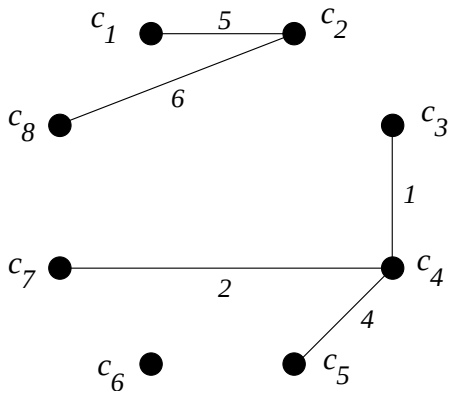


Figure 5.7: Step Five. This drawing corresponds to the subgraph T_5 and to the clustering $\mathbf{C}_5$.

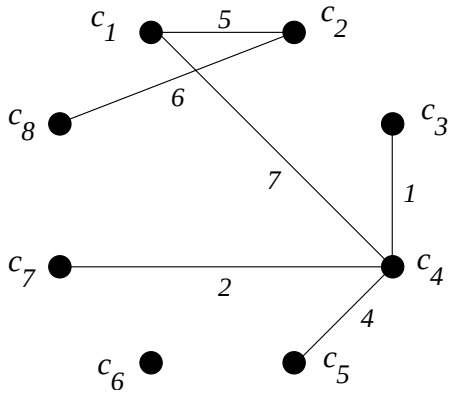


Figure 5.8: Step Six. Two subtrees merged into the subtree T_6. This drawing corresponds to the clustering $\mathbf{C}_6$.

And finally, at the last step we obtain a minimum spanning tree T_7 of the initial graph:

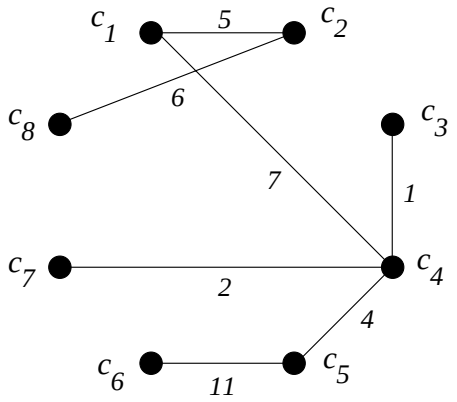


Figure 5.9: The minimum spanning tree of the graph $G(\infty)$; its weight is $w(T) = 36$.

Note that we have exactly repeated all the steps in the construction of consecutive clusterings (Figures 3.1..3.12) in the model example. So that, while building a minimum spanning tree in a graph, we simultaneously generate single-link hierarchical clusterings of the set of vertices of the graph. Conversely, the agglomerative clustering algorithm generates a minimum spanning tree. Thus, the problem of constructing a single-link clustering and that of constructing a minimum spanning tree are, in this sense, equivalent.

Moreover, a minimum spanning tree generates straightforwardly a divisive single-link clustering algorithm: given a minimum spanning tree, we can go backward and remove the edges, one at a time, starting from the heaviest one. In the model example, the minimum spanning tree corresponds to the conjoint clustering $\mathbf{C}_7$. Removing the edge with $d(c_5, c_6) = 11$ off the spanning tree, we get the $\mathbf{C}_6$-clustering. After that, removing the edge with $d(c_1, c_4) = 7$, we generate the $\mathbf{C}_5$-clustering, and so forth.

Exercise 23 Find a minimum spanning tree in graph G_5 (Figure 5.10).

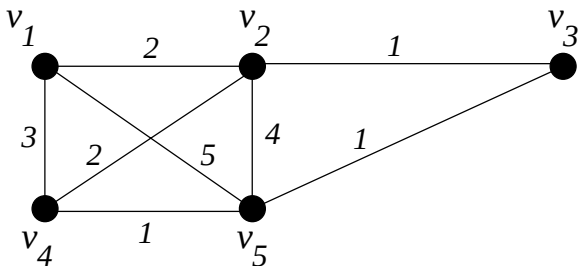


Figure 5.10: Graph G_5.

Exercise 24 Considering the weights of the edges of graph G_5 as the dissimilarities of the objects $V_1, \ldots, V_5$, use the minimum spanning tree from Exercise 23 to construct the single-link clusterings of these objects.

6 HUBERT'S COMPLETE-LINK ALGORITHM

In this section we consider a complete-link clustering algorithm. An essential distinction between single-link and complete-link algorithms is the rule of merging two clusters into one of a higher level. Instead of *connected* subgraphs of the threshold graph $G(\infty)$ used in the single linkage, now we consider the *maximum complete subgraphs* of $G(\infty)$. As we shall see in examples, single linkage and complete linkage may result in different clusterings. In this section we use the same notations as in Section 4 but consider only dissimilarity matrices without ties[5].

Again, we start with an informal description of the algorithm. Given the clustering $\mathbf{C}_m = \{C_{m1}, C_{m2}, \ldots, C_{m,n(m)}\}$ of the m^{th} level, $m = 0, 1, 2, \ldots$, we consider all pairwise unions $C_{ma} \cup C_{mb}$, $a, b = 1, \ldots, n(m), a \neq b$. Suppose, the union $C_{ma} \cup C_{mb}$ contains objects $x_i, \ldots, x_j$. While building a single-linkage, we looked for a single link with the smallest dissimilarity. Now, however, we are adding, one at a time, edges connecting a vertex in C_{ma} with a vertex in C_{mb}, in increasing order of their dissimilarities, until we form a *complete subgraph* (in the sense of Definition 3) of the threshold graph $G(\infty)$ spanned on all these vertices $x_i, \ldots, x_j$. Let $\mathfrak{c}_m(a, b)$ stand for the *maximal* dissimilarity over all edges used in this construction, that is,

$$\mathfrak{c}_m = \mathfrak{c}_m(a, b) = \max\{d(x_i, x_j) | x_i \in C_{ma}, x_j \in C_{mb}\}.$$

Similarly to S_m, $\mathfrak{c}_m$ is a function on pairs of clusters of the m^{th} level. Let $\{C_{mp}, C_{mq}\}$ denote a pair of clusters where this function attains its minimum value and denote the minimum value by $\mathfrak{c}_m(p, q) = \min_{a,b}\{\mathfrak{c}_m(a, b)\}$. To form the clustering $\mathbf{C}_{m+1}$, we merge these two clusters C_{mp} and C_{mq}.

HUBERT'S COMPLETE-LINK ALGORITHM

Given a set $X = \{x_1, x_2, \ldots, x_n\}$, the dissimilarity table, and a threshold value v.

1. Set $m = 0$, and form the disjoint clustering of level zero:

   $$\mathbf{C}_0 = \{C_{01}, C_{02}, \ldots, C_{0n}\}$$

   consisting of n one-element clusters $C_{0k} = \{x_k\}$, $k = 1, \ldots, n$. Define the dissimilarities between the clusters of level zero as $diss(C_{0I}, C_{0j}) = d(x_i, x_j)$.

2. Set $m := m+1$ and calculate the values $C_{m-1}(a, b) = \max\{d(x_i, x_j) | x_i \in C_{m-1,a}, x_j \in C_{m-1,b}\}$ for all pairs of indices $\{a, b\}$ and their minimum value $\min_{a,b} C_{m-1}(a, b) = C_{m-1}(p, q)$. To form the next clustering $\mathbf{C}_m$, we look for two clusters $C_{m-1,p}$ and $C_{m-1,q}$, whose second indices are p and q, and define $C_{m,1} = C_{m-1,p} \cup C_{m-1,p}$. All the other clusters of the $(m-1)^{\text{th}}$ level become, after renumbering, the clusters of level m without changes.

3. Update the dissimilarity table as follows: The dissimilarity between every two "old" clusters (promoted from the preceding level) remains the same. The dissimilarity between the "new" cluster $C_{m,1}$ and any "old" cluster $C_{m-1,r}$ with $r \neq p$ and $r \neq q$ is the smaller of the two dissimilarities $diss(C_{m-1,p}, C_{m-1,r})$ and $diss(C_{m-1,q}, C_{m-1,r})$.

[5]Clustering in the presence of ties is discussed, for example, in [7], p. 76.

4. Continue until we reach the threshold value v or all the objects are merged into one conjoint cluster, whichever occurs first.

$\diamondsuit$

Remark 6.1 In step 2 we combine two clusters into a new one only when we reach an edge with the maximal dissimilarity between the entities in the two clusters; so to say, we link them *completely*. In terms of graphs, we merge two complete subgraphs G_p and G_q by using all edges with one end in G_p and another end in G_q.

Remark 6.2 Unlike the single linkage where we calculate a double minimum of the dissimilarities (first over a fixed pair of clusters and then over all pairs of clusters — see Remark 4.2 after Hubert's single-link algorithm), in the complete linkage we calculate a minimum of the maximal values: first we calculate the maximal dissimilarity of the objects over a fixed pair of clusters and then the minimum of these maximums over all pairs of clusters.

We apply this algorithm to our model example with the dissimilarity table reproduced in the next page. A graph generated by two vertices c_3 and c_4 is a complete subgraph (it is isomorphic to K_2) of the threshold graph $G(1)$:

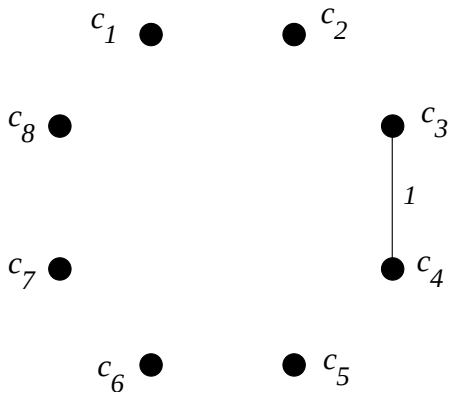


Figure 6.1: The threshold graph $G(1)$.

Therefore, first two clusterings are the same as in the single-link case:

$$\mathbf{C}_0 = \{C_{01}, C_{02}, C_{03}, C_{04}, C_{05}, C_{06}, C_{07}, C_{08}\},$$

where $C_{0i} = \{c_i\}$, $i = 1, \ldots, 8$, and $\mathbf{C}_1 = \{C_{11}, C_{12}, C_{13}, C_{14}, C_{15}, C_{16}, C_{17}\}$, where $C_{11} = C_{03} \cup C_{04} = \{c_3, c_4\}, C_{1,i} = C_{0,i-1} = \{c_{i-1}\}$ for $i = 2, 3$, and $C_{1i} = C_{0,i+1} = \{c_{i+1}\}$ for $i = 4, \ldots, 7$.

However, the threshold graph $G(2)$ does not contain a complete subgraph — its subgraph, spanned by the vertices c_3, c_4, and c_7, is not a complete graph. Thus, even though $G(2)$ generates a single-link clustering (see Section 4), it does not generate a complete-link clustering (see Figure 6.2).

Pair $\{c_i, c_j\}$	Dissimilarity $d(i,j)$
$\{c_3, c_4\}$	1
$\{c_4, c_7\}$	2
$\{c_3, c_7\}$	3
$\{c_4, c_5\}$	4
$\{c_1, c_2\}$	5
$\{c_2, c_8\}$	6
$\{c_1, c_4\}$	7
$\{c_2, c_3\}$	8
$\{c_3, c_5\}$	9
$\{c_1, c_3\}$	10
$\{c_5, c_6\}$	11
$\{c_2, c_4\}$	12
$\{c_1, c_8\}$	13
$\{c_4, c_6\}$	14
$\{c_6, c_7\}$	15
$\{c_5, c_7\}$	16
$\{c_2, c_7\}$	17
$\{c_5, c_8\}$	18
$\{c_3, c_6\}$	19
$\{c_6, c_8\}$	20
$\{c_4, c_8\}$	21
$\{c_1, c_5\}$	22
$\{c_2, c_6\}$	23
$\{c_7, c_8\}$	24
$\{c_1, c_7\}$	25
$\{c_3, c_8\}$	26
$\{c_1, c_6\}$	27
$\{c_2, c_5\}$	28

Table 6.1: Dissimilarity table (Table 3.2) for the model example.

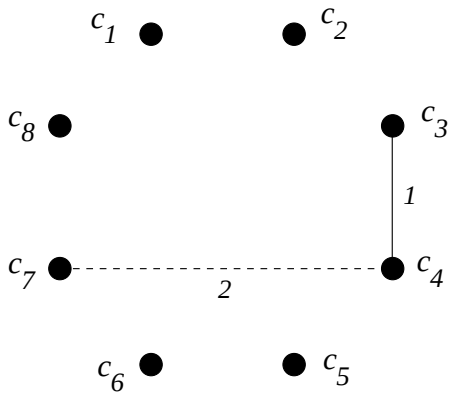


Figure 6.2: The threshold graph $G(2)$. The edge $\{c_4, c_7\}$ is dotted because it does not generate the next level complete-link clustering.

The threshold graph $G(3)$ contains a complete subgraph, isomorphic to K_3, spanned by the vertices $\{c_3, c_4, c_7\}$:

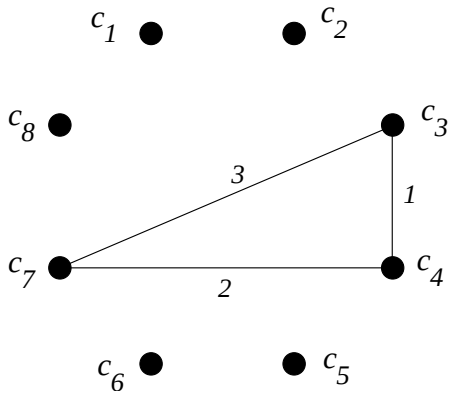


Figure 6.3: The threshold graph $G(3)$.

So that, we merge these three vertices into a cluster and the next complete-link clustering is $\mathbf{C}_2 = \{C_{21}, C_{22}, C_{23}, C_{24}, C_{25}, C_{26}\}$, where $C_{21} = C_{11} \cup C_{16} = \{c_3, c_4, c_7\}$. Five other clusters contain only one vertex each. It is worth noting that we have used here three edges — three links. At this step the single-link and the complete-link clusterings still coincide. The next threshold graph $G(4)$ is generated by the edge $\{c_4, c_5\}$. However, this edge does not generate a new complete subgraph, that is why it is dotted in the next figure. Thus, the threshold graph $G(4)$ does not generate the next level of complete clustering.

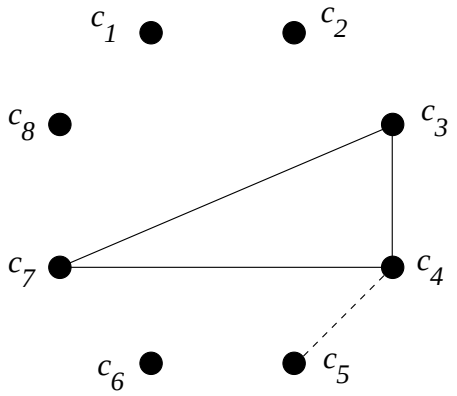


Figure 6.4: The threshold graph $G(4)$. The subgraph spanned on the vertices $\{c_3, c_4, c_5, c_7\}$ is not complete.

The threshold graph $G(5)$ contains a K_2-isomorphic subgraph with the vertices c_1 and c_2. Thus, it generates a new complete-link clustering $\mathbf{C}_3 = \{C_{31}, C_{32}, C_{35}, C_{36}, C_{33}\}$, where $C_{31} = \{c_1, c_2\}$, $C_{32} = \{c_3, c_4, c_7\}$, $C_{33} = \{c_5\}$, $C_{34} = \{c_6\}$, and $C_{35} = \{c_8\}$. Starting from this step, Hubert's complete-link algorithm generates clusterings different from the single linkage.

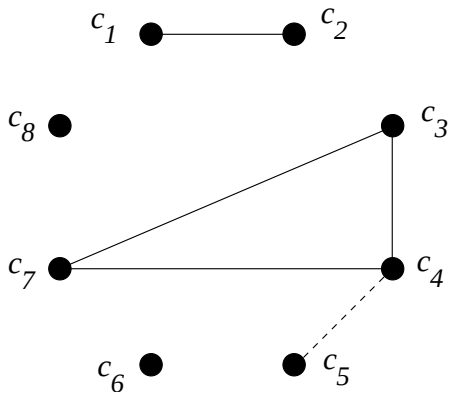


Figure 6.5: The threshold graph $G(5)$.

$G(6)$ does not contain a new complete subgraph either:

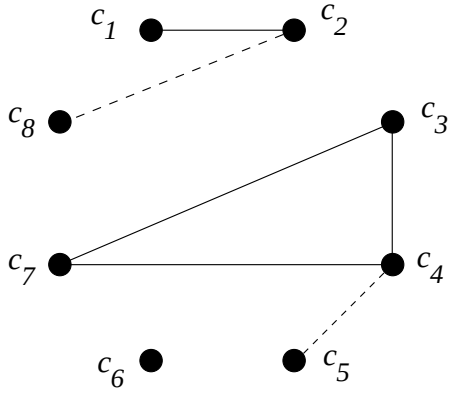


Figure 6.6: The threshold graph $G(6)$.

The next threshold graphs, $G(7)$ – $G(12)$ (Figure 6.7) also contain no new complete subgraphs and generate no new clusterings.

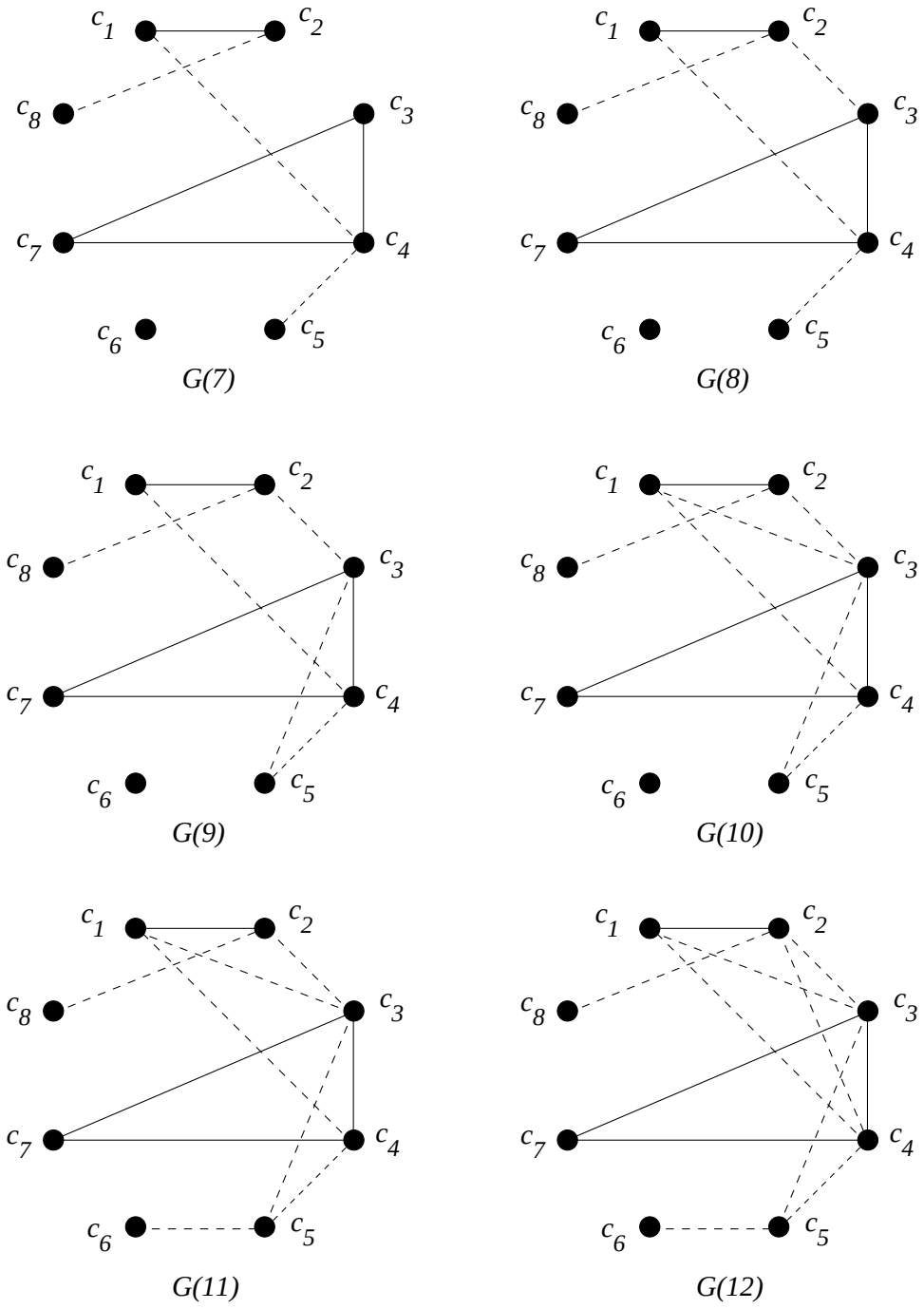


Figure 6.7: The threshold graphs $G(7)$ – $G(12)$.

Only in the threshold graph $G(13)$ a vertex, namely, c_8, is completely connected with both vertices of a previous cluster (and the edge $\{c_2, c_8\}$ becomes now continuous).

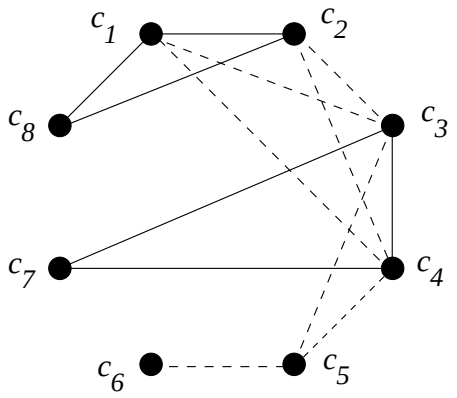


Figure 6.8: The threshold graph $G(13)$.

Thus, we obtain the next clustering $\mathbf{C}_4 = \{C_{41}, C_{43}, C_{45}, C_{46}\}$, where, $C_{41} = \{c_1, c_2, c_8\}$, $C_{42} = \{c_3, c_4, c_7\}$, $C_{43} = \{c_5\}$, $C_{44} = \{c_6\}$.

The threshold graphs $G(14)$ and $G(15)$ do not generate a new clustering as well:

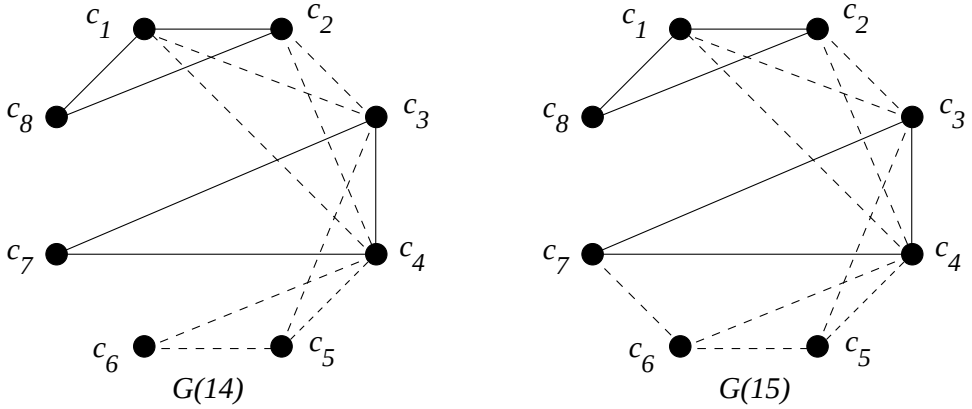


Figure 6.9: The threshold graphs $G(14)$, $G(15)$.

However, in the next threshold graph $G(16)$ the vertex c_5 is completely connected with all the vertices of an old cluster C_{42} and we get the fifth-level complete-link clustering, $\mathbf{C}_5 = \{C_{51}, C_{52}, C_{53}\}$, consisting of clusters $C_{51} = \{c_3, c_4, c_5, c_7\}$, $C_{52} = \{c_1, c_2, c_8\}$, and $C_{53} = \{c_6\}$ (Figure 6.10).

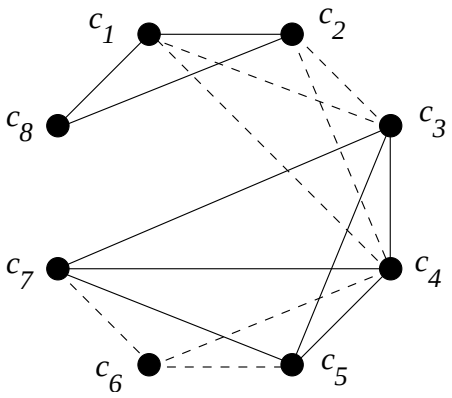


Figure 6.10: The threshold graph $G(16)$.

We continue building the threshold graphs:

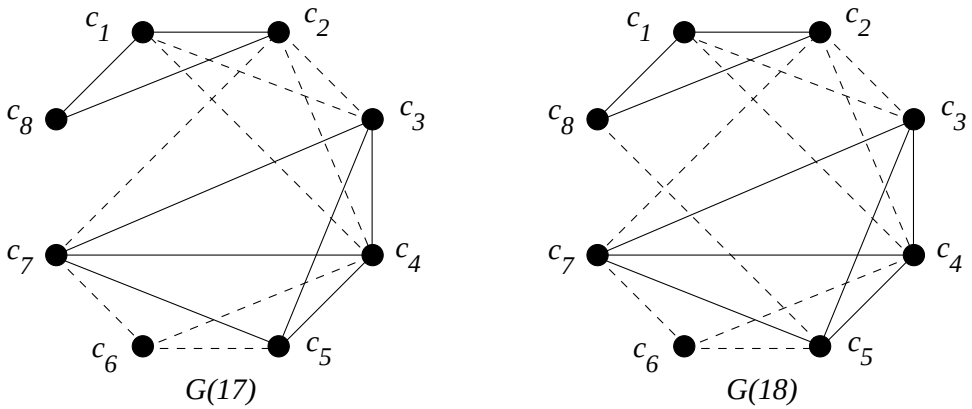


Figure 6.11: The threshold graphs $G(17)$, $G(18)$.

The threshold graph $G(19)$ generates the next complete-link clustering $\mathbf{C}_6 = \{C_{61}, C_{62}\}$, with $C_{61} = \{c_3, c_4, c_5, c_6, c_7\}$ and $C_{62} = C_{52}$:

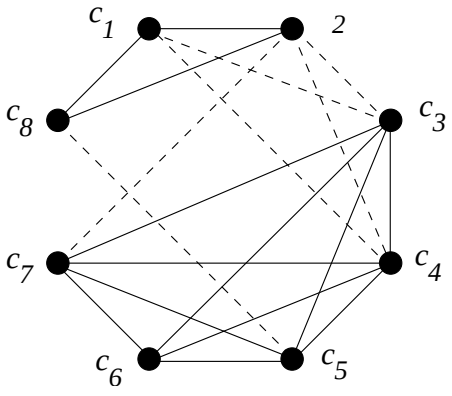


Figure 6.12: The threshold graph $G(19)$.

Obviously, only the last threshold graph $G(28)$ generates the final conjoint complete-link clustering $\mathbf{C}_7$.

Remark 6.3 We repeat that in this example only the first three levels in the single-link and complete-link clusterings coincide. From the fourth level on, the clusters are different.

Finally, we draw a dendrogram for the complete-link clustering in this example, which is different from the one in Section 4 (Figure 4.1):

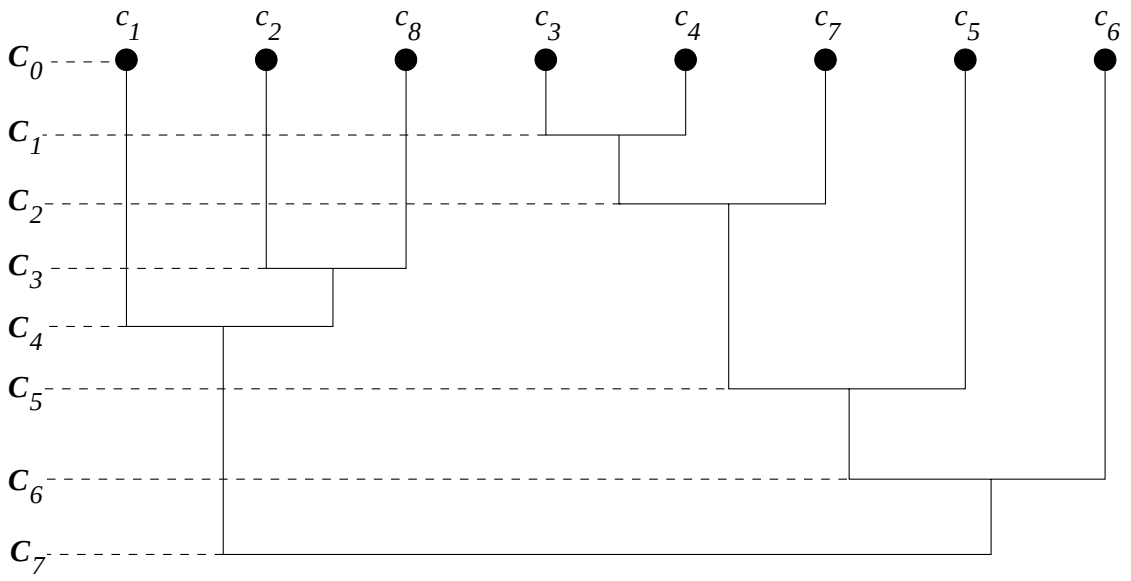


Figure 6.13: A dendrogram for the complete-link clustering in the model example.

Exercise 25 Using Hubert's complete-link algorithm, build conjoint clustering of the set X, whose dissimilarity table is given below:

	x_1	x_2	x_3	x_4	x_5
x_1	0	4	1	3	8
x_2		0	2	5	10
x_3			0	6	7
x_4				0	9
x_5					0

Exercise 26 Give an example of a 4-element set with different single-link and complete-link clusterings.

7 CASE STUDY

Table 7.1 contains the actual final grades and GPA scores of 15 students in a Statistics class. The students are listed in alphabetical order. The GPA scores were calculated earlier, so that they do not reflect the grades in the Statistics class. Using the final grades, we build single-link clusterings[6] of this 15-element set and compare the results with the students' GPA scores. Our goal in doing this comparison is to assess the reliability and validity of the presented clustering algorithm.

Name	St1	St2	St3	St4	St5	St6	St7
Final Grade	62	54	71	60	36	81	84
GPA	1.808	2.369	3.058	2.825	2.460	3.681	3.508

Name	St8	St9	St10	St11	St12	St13	St14	St15
Final Grade	69	55	70	58	61	60	40	75
GPA	2.793	2.738	3.123	3.100	2.197	2.285	2.113	2.703

Table 7.1: The final grades and GPA scores.

As a measure of dissimilarity, we chose the absolute value of the difference between the final grades and completed the dissimilarity table:

	St1	St2	St3	St4	St5	St6	St7	St8	St9	St10	St11	St12	St13	St14	St15
St1	0	8	9	2	26	19	22	7	7	8	4	1	2	22	13
St2		0	17	6	18	27	30	15	1	16	4	7	6	14	21
St3			0	11	35	10	13	2	16	1	13	10	11	31	4
St4				0	24	21	24	9	5	10	2	1	0	20	15
St5					0	45	48	33	19	34	22	25	24	4	39
St6						0	3	12	26	11	23	20	21	41	6
St7							0	15	29	14	26	23	24	44	9
St8								0	14	1	11	8	9	29	6
St9									0	15	3	6	5	15	20
St10										0	12	9	10	30	5
St11											0	3	2	18	17
St12												0	1	21	14
St13													0	20	15
St14														0	35

Table 7.2: The dissimilarity table for the Statistics class.

In this problem we have many ties. Thus, some intermediate steps are not unique. However, application of the agglomerative single-link algorithm (Section 3) gives the following results. Since there are two elements whose dissimilarity is zero, the first-level clustering $\mathbf{C}_1$ contains one two-element cluster $\{St4, St13\}$ and thirteen one-element ones. If the threshold level does not exceed 1, we get nine clusters:

$$\mathbf{C}_2 = \{\{S_1, S_4, S_{12}, S_{13}\}, \{S_3, S_8, S_{10}\}, \{S_2, S_9\}, \{S_5\}, \{S_6\}, \{S_7\}, \{S_{11}\}, \{S_{14}\}, \{S_{15}\}\}.$$

[6]We do not apply here complete-link clustering because the problem involves ties.

At the level 2 we have eight clusters:

$$\mathbf{C}_3 = \{\{St1, St4, St11, St12, St13\}, \{St3, St8, St10\}, \{St2, St9\}, \{St5\}, \{St6\}, \{St7\}, \{St14\}, \{St15\}\}.$$

At the level 3 we have six clusters:

$$\mathbf{C}_4 = \{\{St1, St2, St4, St9, St11, St12, St13\}, \{St3, St8, St10\}, \{St6, St7\}, \{St5\}, \{St14\}, \{St15\}\}.$$

At the level 4 we have only four clusters:

$$\mathbf{C}_5 = \{\{St1, St2, St4, St9, St11, St12, St13\}, \{St3, St8, St10, St15\}, \{St6, St7\}, \{St5, St14\}\}.$$

There is no merger at the level 5, but two of these clusters link at the level of 6:

$$\mathbf{C}_6 = \{\{St1, St2, St4, St9, St11, St12, St13\}, \{St3, St6, St7, St8, St10, St15\}, \{St5, St14\}\}.$$

At the level 7 only two clusters left:

$$\mathbf{C}_7 = \{\{St1, St2, St3, St4, St6, St7, St8, St9, St10, St11, St12, St13, St15\}, \{St5, St14\}\}.$$

Ultimately, these two clusters amalgamate to the conjoint clustering at the fourteenth level.

Now we want to assess the clustering derived. The following three charts represent three clusters in $\mathbf{C}_6$. Every chart contains the GPA-scores of the students in the corresponding cluster (See Table 7.4).

C_{61}	Name	St1	St2	St4	St9	St11	St12	St13
	GPA	1.808	2.369	2.825	2.738	3.100	2.197	2.285

C_{62}	Name	St3	St6	St7	St8	St10	St15
	GPA	3.058	3.681	3.508	2.793	3.123	2.703

C_{63}	Name	St5	St14
	GPA	2.460	2.113

Table 7.3: The $\mathbf{C}_6$-clustering.

Naturally, as is always the case while dealing with real data, there is no perfect match. However, we see that at this threshold level the clusters C_{62} and C_{63} demonstrate good uniformity of the GPA scores contained, while C_{61} comprises more variety of scores.

Next, let us consider the clustering $\mathbf{C}_5$. The following four charts represent its four clusters. Every chart contains the GPA-scores of the students in the corresponding cluster.

Again, we see that there is a sensible closeness of the GPA scores within the clusters C_{52}, C_{53}, and C_{54}. In particular, the cluster C_{53} contains two highest GPA scores.

C_{51}	Name	St1	St2	St4	St9	St11	St12	St13
	GPA	1.808	2.369	2.825	2.738	3.100	2.197	2.285

C_{52}	Name	St3	St8	St10	St15
	GPA	3.058	2.793	3.123	2.703

C_{53}	Name	St6	St7
	GPA	3.681	3.508

C_{54}	Name	St5	St14
	GPA	2.460	2.113

Table 7.4: The $\mathbf{C}_5$-clustering.

To get more quantifiable assessment of the clusterings derived, we make some calculations. The following charts contain the averaged statistics grades and the averaged GPA scores for each cluster in all levels:

Clustering C_0	Averaged statistics grade	Averaged GPA score
Cluster C_{01}	62.4	2.717

Table 7.5: The averaged statistics grades and GPA-scores over the whole class — the $\mathbf{C}_0$-clustering.

Clustering $\mathbf{C}_1$	Averaged statistics grade	Averaged GPA score
Cluster $C_{1,1}$	60	2.555
Cluster $C_{1,2}$	62	1.808
Cluster $C_{1,3}$	54	2.369
Cluster $C_{1,4}$	71	3.058
Cluster $C_{1,5}$	36	2.46
Cluster $C_{1,6}$	81	3.681
Cluster $C_{1,7}$	84	3.508
Cluster $C_{1,8}$	69	2.793
Cluster $C_{1,9}$	55	2.738
Cluster $C_{1,10}$	70	3.123
Cluster $C_{1,11}$	58	3.1
Cluster $C_{1,12}$	61	2.197
Cluster $C_{1,13}$	40	2.113
Cluster $C_{1,14}$	75	2.703

Table 7.6: The averaged statistics grades and GPA-scores over the $\mathbf{C}_1$-clustering.

Clustering $\mathbf{C}_2$	Averaged statistics grade	Averaged GPA score
Cluster $C_{2,1}$	60.75	2.279
Cluster $C_{2,2}$	70	2.991
Cluster $C_{2,3}$	54.5	2.554
Cluster $C_{2,4}$	36	2.46
Cluster $C_{2,5}$	81	3.681
Cluster $C_{2,6}$	84	3.508
Cluster $C_{2,7}$	58	3.1
Cluster $C_{2,8}$	40	2.113
Cluster $C_{2,9}$	75	2.703

Table 7.7: The averaged statistics grades and GPA-scores over the $\mathbf{C}_2$-clustering.

Clustering $\mathbf{C}_3$	Averaged statistics grade	Averaged GPA score
Cluster $C_{3,1}$	60.2	2.443
Cluster $C_{3,2}$	70	2.991
Cluster $C_{3,3}$	54.5	2.554
Cluster $C_{3,4}$	36	2.46
Cluster $C_{3,5}$	81	3.681
Cluster $C_{3,6}$	84	3.508
Cluster $C_{3,7}$	40	2.113
Cluster $C_{3,8}$	75	2.703

Table 7.8: The averaged statistics grades and GPA-scores over the C_3-clustering.

Clustering $\mathbf{C}_4$	Averaged statistics grade	Averaged GPA score
Cluster $C_{4,1}$	58.57	2.475
Cluster $C_{4,2}$	70	2.991
Cluster $C_{4,3}$	82.5	3.594
Cluster $C_{4,4}$	36	2.46
Cluster $C_{4,5}$	40	2.113
Cluster $C_{4,6}$	75	2.703

Table 7.9: The averaged statistics grades and GPA-scores over the $\mathbf{C}_4$-clustering.

Clustering $\mathbf{C}_5$	Averaged statistics grade	Averaged GPA score
Cluster $C_{5,1}$	58.57	2.475
Cluster $C_{5,2}$	71.25	2.919
Cluster $C_{5,3}$	82.5	3.595
Cluster $C_{5,4}$	38	2.287

Table 7.10: The averaged statistics grades and GPA-scores over the $\mathbf{C}_5$-clustering.

Clustering $\mathbf{C}_6$	Averaged statistics grade	Averaged GPA score
Cluster $C_{6,1}$	58.57	2.475
Cluster $C_{6,2}$	75	3.144
Cluster $C_{6,4}$	38	2.287

Table 7.11: The averaged statistics grades and GPA-scores over the $\mathbf{C}_6$-clustering.

Finally, the following chart contains Pearson's correlation coefficients between the averaged grades and GPA-scores for every level of clustering:

Clustering level	$\mathbf{C}_0$	$\mathbf{C}_1$	$\mathbf{C}_2$	$\mathbf{C}_3$	$\mathbf{C}_4$	$\mathbf{C}_5$	$\mathbf{C}_6$
Correlation coefficient	0.659	0.671	0.788	0.850	0.832	0.925	0.929

Table 7.12: The Pearson coefficient of correlation.

We see that every next level of clustering. except for $\mathbf{C}_4$, improves correlation of the final grades and the GPA scores. This observation validates our procedure.

ANSWERS TO EXERCISES

Exercise 1 {DE}, {FL, LA}, {MI}, {MD, SC}, {AL}, { GA, NC}, {KY, MO}, {TN, VA}, {WV}, {TX}.

Exercise 2 Four clusters: {DE, FL, LA, MD, MI, SC}, {AL, GA, KY, MO, NC}, {TN, VA}, {WV, TX}.

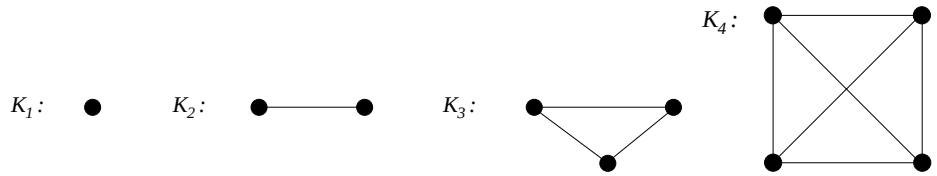
Five clusters: {DE, FL, LA}, {AL, MD, MI, SC}, {GA, KY, MO, NC}, {TN, VA}, {WV, TX}.

Exercise 3 This dissimilarity value is 3.

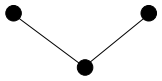
Exercise 4 $(v_1, e_1, v_2, e_3, v_3, e_2, v_1)$; $(v_1, e_1, v_2, e_3, v_3, e_4, v_4)$.

Exercise 5 Both graphs are connected.

Exercise 6



Exercise 7 The minimal number of vertices is 3:



Exercise 8 G_1 consists of two connected components, G_2 is connected itself.

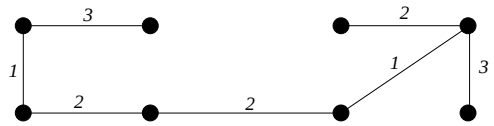
Exercise 9 If such an edge does exist, then it must belong to a cycle.

Exercise 10 Use Mathematical Induction with respect to n.

Exercise 11 Any tree satisfies this condition. See also answer to Exercise 13.

Exercise 12 $\{e_1, e_4, e_5\}$; $\{e_1, e_3, e_5\}$; $\{e_1, e_2, e_5\}$; $\{e_3, e_4, e_5\}$; $\{e_2, e_4, e_5\}$.

Exercise 13

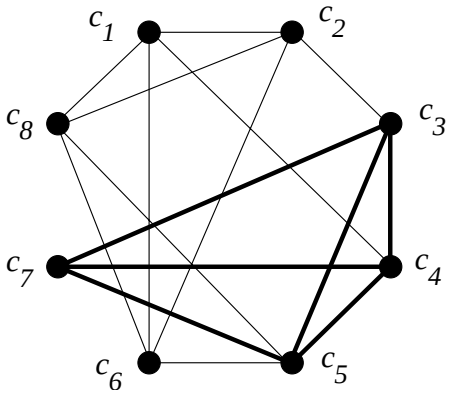


Exercise 14 a) Every whole number is nonnegative and every integer is either positive or zero (that is, it is a whole number) or negative.

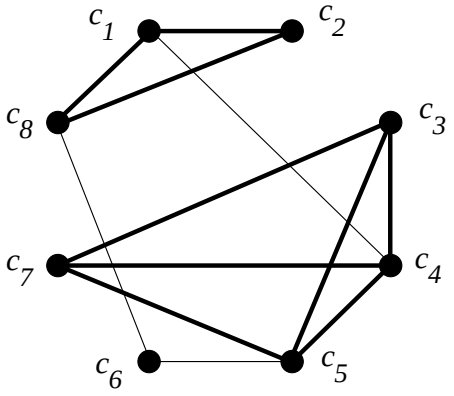
b) Every integer is either positive, or zero, or negative, and these three subsets are, obviously, disjoint.

c) The partition in b) is nested in the partition in a).

Exercise 15 The following road maps correspond to the clustering $\mathbf{C}_3$:



and clustering $\mathbf{C}_5$:



Exercise 16 For example, the clustering $\mathbf{C}_0$ corresponds to a trivial disjoint partition, where every point is equivalent to itself only; $\mathbf{C}_7$ corresponds to another trivial partition, where there is only one equivalence class, that is, each point is equivalent to every other one. The partition, generated by $\mathbf{C}_3$, consists of five equivalence classes $\{c_3, c_4, c_5, c_7\}$, $\{c_1\}$, $\{c_2\}$, $\{c_6\}$, $\{c_8\}$.

Exercise 17 If a dissimilarity table of size n contains the maximal possible number of different entries, that is, these entries are $1, 2, 3, \ldots, n(n-1)/2$, it generates $n(n-1)/2+1$ different threshold graphs. Each tie in the table decreases this number by one.

Exercise 18 There may be different intermediate clustering on levels 2 or 6, depending on the order we resolve the ties; all the other clusterings are the same.

Exercise 19 The problem contains the ties starting from the dissimilarity level of 2. Therefore, the first-level clustering is unique: $\{x_2, x_5\}$, $\{x_1\}$, $\{x_3\}$, $\{x_4\}$, $\{x_6\}$. However, the second-level clustering depends upon what edge with the dissimilarity of 2 is chosen first. This clustering may be $\{\{x_2, x_3, x_5\}, \{x_1\}, \{x_4\}, \{x_6\}\}$ or $\{\{x_2, x_5\}, \{x_4, x_6\}, \{x_1\}, \{x_3\}\}$. In turn, these clusterings lead to different clusterings of the next level. After that, the coming conjoint clustering is, of course, unique.

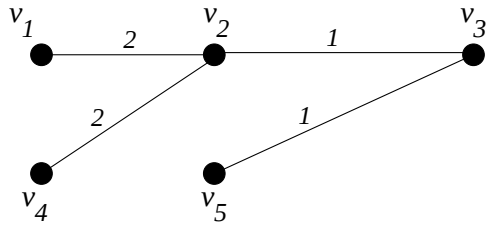
Exercise 20 Of course, Hubert's algorithm generates the same clustering as in Exercise 19. For example, the first updated dissimilarity table is

	$\{x_1\}$	$\{x_2, x_5\}$	$\{x_3\}$	$\{x_4\}$	$\{x_6\}$
$\{x_1\}$	0	4	8	3	8
$\{x_2, x_5\}$		0	2	4	4
$\{x_3\}$			0	6	3
$\{x_4\}$				0	2
$\{x_6\}$					0

Exercise 21 See answer to Exercise 18.

Exercise 22 See [11], Section 8.6.

Exercise 23



Exercise 24 Because of ties, it is possible to generate different chains of clusterings, for example,

$$\{v_2\}, \{v_3\}, \{v_5\}, \{v_1\}, \{v_4\};$$
$$\{v_2, v_3\}, \{v_5\}, \{v_1\}, \{v_4\};$$
$$\{v_2, v_3, v_5\}, \{v_1\}, \{v_4\};$$
$$\{v_2, v_3, v_5, v_1\}, \{v_4\};$$
$$\{v_2, v_3, v_5, v_1, v_4\}.$$

Exercise 25 Level 1: $\{x_1, x_3\}, \{x_2\}, \{x_4\}, \{x_5\}$; Level 4: $\{x_1, x_2, x_3\}, \{x_4\}, \{x_5\}$; Level 6: $\{x_1, x_2, x_3, x_4\}, \{x_5\}$; Level 10: $\{x_1, x_2, x_3, x_4, x_5\}$.

Exercise 26 Consider the following dissimilarity table:

	x_1	x_2	x_3	x_4
x_1	0	1	6	5
x_2		0	2	4
x_3			0	3
x_4				0

References

[1] Anderberg, M. R., *Cluster Analysis for Applications.* Academic Press, 1973.

[2] Cox, C., P. Hansen, B. Julesh, Eds. *Partitioning Data Sets.* AMS, 1995.

[3] Everitt, B., *Cluster Analysis.* Heinemann Educational Books, 1974.

[4] Gordon, A. D., *Classification.* Chapman and Hall, 1981.

[5] Hartigan, J. A., *Clustering Algorithms.* Wiley, 1975.

[6] Hubert, L. J., Some applications of graph theory to clustering. *Psychometrika,* Vol.39(1974), 283-309.

[7] Jain, A. K., R. C. Dubes, *Algorithms for Clustering Data.* Prentice Hall, 1988.

[8] Johnson, R. A., D. W. Wichern, *Applied Multivariate Statistical Analysis.* Chap. 12. Prentice Hall, 1988.

[9] Kaufman, L., P. J. Rousseeuw, *Finding Groups in Data: An Introduction to Cluster Analysis.* Wiley, NY, 1990.

[10] Mirkin, B., *Mathematical Classification and Clustering.* Kluwer, 1996.

[11] Rosen, K. H., *Discrete Mathematics and its Applications,* 4th ed. Mcgraw-Hill, 1999.