A Randomized Algorithm for Finding Maximum with $O((\log n)^2)$ Polynomial Tests

Hing F. Ting[1] and Andrew C. Yao[2]

Abstract

A well-known result by Rabin [1] implies that $n-1$ polynomial tests are necessary and sufficient in the worst case to find the maximum of n distinct real numbers. In this note we show that, for any fixed constant $c > 0$, there is a randomized algorithm with error probability $O(n^{-c})$ for finding the maximum of n distinct real numbers using only $O((\log n)^2)$ polynomial tests.

keywords: algorithms, decision trees, maximum, parity tests, randomized algorithms.

[1]Department of Computer Science, Chinese University of Hong Kong, Shatin, Hong Kong.

[2]Department of Computer Science, Princeton University, Princeton, New Jersey 08544, USA.

A *polynomial decision tree* is a decision tree using tertiary tests of the form $p(x_1, x_2, \cdots, x_n) : 0$ where p are polynomials over the reals. Clearly, there is a polynomial decision tree of height $n-1$ for finding the maximum of n distinct input numbers $x_1, x_2, \cdots, x_n$. A well-known result by Rabin [1, Theorem 4] implies that this is also a lower bound. In this paper, we show that the number of tests can be drastically reduced if randomization is allowed.

Theorem 1 Let $c > 0$ be any fixed constant. There is a randomized polynomial decision tree of height $O((\log n)^2)$ for finding the maximum of n distinct real numbers $x_1, x_2, \cdots, x_n$ with error probability less than $O(n^{-c})$.

Remark. It is an interesting open problem whether the height $O((\log n)^2)$ can be further reduced.

The rest of the paper is devoted to a proof of Theorem 1. Without loss of generality, we can assume that $c > 10$ and $n \geq 2$. In what follows, $x_1, x_2, \cdots, x_n$ are understood to be distinct. The notation $[n]$ stands for the set $\{1, 2, \cdots, n\}$. For any finite set of distinct real numbers, an element of the set is said to have *rank* r, if it is the r-th largest in the set.

For any $i \in [n]$ and $B \subseteq [n] - \{i\}$, let $P(i, B)$ denote the test $\prod_{j \in B}(x_i - x_j) : 0$. Clearly, the answer to the test is "$>$" if the rank of x_i in the set $\{x_j \mid j \in B\} \cup \{x_i\}$ is odd, and "$<$" if the rank is even. We call $P(i, B)$ a *parity test.* The only polynomial tests we employ in the proof will be parity tests.

Let $i \in [n]$. We will show that, in a small number of random parity tests, one can either find a random $i' \in [n]$ such that $x_{i'} > x_i$ or declare with confidence that x_i is the maximum element we seek. Let m be any positive integer.

Lemma 1 There is a randomized algorithm using no more than $m + \lceil \log_2(n-1) \rceil$ parity tests that (a) either declares "i *is max*" or outputs an $i' \in [n]$ such that $x_{i'} > x_i$, and (b) satisfies the following properties: if x_i is the maximum of $\{x_1, x_2, \cdots, x_n\}$, the algorithm always declares that i is max; if x_i is not the maximum, then with probability exactly equal to $1 - 2^{-m}$, the algorithm will output an i' . Furthermore, the conditional distribution of i' is uniform over the set $\{j \mid j \in [n], x_j > x_i\}$, conditioned upon the event that the algorithm does give an output.

Let us assume that Lemma 1 is true for the moment, and show how to use it to prove the main theorem. We describe below a procedure for finding the maximum. Let $m = \lceil 3c \log_2 n \rceil$. Take a random $i \in [n]$, and use the algorithm in Lemma 1 to generate an i'; use i' as the new i and repeat. The procedure halts when either a declaration "i is max" is made or when the number of iterations completed equals m' where $m' = \lfloor 10c \log_2 n \rfloor$.

The output of the algorithm is the value of i when the procedure halts. Let us call this procedure FINDMAX.

The number of parity tests made by FINDMAX is no greater than $(m + \lceil \log_2(n-1) \rceil)\, m' = O((\log n)^2)$. Thus, FINDMAX gives rise to a randomized polynomial decision tree of height $O((\log n)^2)$.

Let $(x_1, x_2, \cdots, x_n)$ be any input of distinct real numbers. We need to show that the error probability ϵ for FINDMAX to output some false maximum element is $O(n^{-c})$. An error may be made either when the procedure has completed m' iterations without making a declaration or when the algorithm makes a declaration which turns out to be inaccurate. Let ϵ_1, ϵ_2 be the error probabilities due to these two cases, then $\epsilon = \epsilon_1 + \epsilon_2$. From Lemma 1, for each $1 \leq \ell \leq m'$, the probability of reaching the ℓ-th iteration and making an incorrect declaration during that iteration is at most 2^{-m}. This implies that $\epsilon_2 \leq m'\, 2^{-m} = O(n^{-c})$.

It remains to prove that $\epsilon_1 = O(n^{-c})$. Consider the following process $\mathcal{G}$ of generating a random decreasing sequence of integers $r_1, r_2, \cdots, r_s$: take a random r_1 uniformly from integers in $[1, n]$; for $\ell \geq 1$, take a random $r_{\ell+1}$ uniformly from $[1, r_\ell - 1]$, and stop when we get $r_s = 1$. For each $\ell \geq 1$, let λ_ℓ be the probability that $s \geq \ell$, i.e., the sequence generated has length greater than or equal to ℓ.

We claim that $\epsilon_1 \leq \lambda_{m'}$. To see this, consider the sequence of values assumed by i at the beginning of successive iterations when FINDMAX is executed; call this sequence $i_1, i_2, \cdots, i_s$, where s is the total number of iterations carried out (and thus $s \leq m'$). If we denote by r_j the rank of x_{i_j} in the input set $\{x_1, x_2, \cdots, x_n\}$, then by the last property listed in Lemma 1, the sequence $r_1, r_2, \cdots, r_s$ has the probability distribution as the sequence generated by $\mathcal{G}$ and truncated, if necessary, after the first m' terms. This proves our claim.

Consider a random sequence $r_1, r_2, \cdots, r_s$ generated by $\mathcal{G}$. For $1 \leq \ell \leq s$, we call ℓ a *big step* if $r_\ell \leq (r_{\ell-1} - 1)/2$ (r_0 is understood to be $n+1$); otherwise, ℓ is a *small step*. There can be at most $\lceil \log_2 n \rceil$ big steps, since each big step decreases the value of r_ℓ by at least a factor of 2. Let Δ be the set of all $\delta = (j_1, j_2, \cdots, j_t)$, where $t = m' - \lceil \log_2 n \rceil$ and $1 \leq j_1 < j_2 < \cdots < j_t \leq m'$. Then

$$\lambda_{m'} \leq \sum_{\delta \in \Delta} q_\delta, \tag{1}$$

where q_δ is the probability that $s \geq m'$ and that the steps $j_1, j_2, \cdots, j_t$ are small steps (there may be other small steps).

Let $\mathbf{E}'_\mathbf{1}$ be the event that 1 is a small step. For each $2 \leq k \leq t$, let $\mathbf{E_k}$ be the event that

$r_{j_k-1} > 1$ and that $j_1, j_2, \cdots, j_{k-1}$ are small steps; let $\mathbf{E}'_{\mathbf{k}}$ be the event that j_k is a small step. Let $\alpha_1 = \Pr\{\mathbf{E}'_{\mathbf{1}}\}$, and for $2 \leq k \leq t$, let $\alpha_k = \Pr\{\mathbf{E}'_{\mathbf{k}} \mid \mathbf{E}_{\mathbf{k}}\}$. Then $q_\delta \leq \prod_{1 \leq k \leq t} \alpha_k$. As each α_k is no greater than $2/3$, we have

$$q_\delta \leq (2/3)^t. \tag{2}$$

It follows from (1) and (2) that

$$\begin{aligned}
\epsilon_1 &\leq \lambda_{m'} \\
&\leq \binom{m'}{t} \left(\frac{2}{3}\right)^t \\
&= \binom{m'}{\lceil \log_2 n \rceil} \left(\frac{2}{3}\right)^t \\
&\leq \left(\frac{m'e}{\lceil \log_2 n \rceil}\right)^{\lceil \log_2 n \rceil} \left(\frac{2}{3}\right)^t \\
&\leq \left(\frac{2}{3}\right)^{4c \log_2 n} \\
&= O(n^{-c}).
\end{aligned}$$

We have thus shown that Lemma 1 implies Theorem 1. To complete the proof of Theorem 1, we now turn to the proof of Lemma 1. We break the discussion into two lemmas.

We introduce some notations. Let F be any finite set. For any integer $d \geq 0$, let $\mathcal{F}_d$ be the family of size-d subsets of F. When we say *generating a uniform random d-subset F' of F*, we mean that we generate a random F' uniformly chosen from $\mathcal{F}_d$. We also describe the random F' generated this way as *a uniform random d-subset of F*. Similarly, *generating a uniform random subset F' of F* means generating a random F' uniformly chosen ¿from $\cup_{d \geq 0} \mathcal{F}_d$. Also, F' is *a uniform random odd-subset of F* means that F' is uniformly chosen from $\cup_{d=1,3,5,\ldots} \mathcal{F}_d$.

Lemma 2 Let m be any positive integer. There is a randomized algorithm using m parity tests that, for any input $i \in [n]$ and $(x_1, x_2, \cdots, x_n)$, (a) either declares "i *is max*" or outputs a set $B \subseteq [n] - \{i\}$, and (b) satisfies the following properties: if x_i is the maximum, the algorithm always declares "i is max"; if x_i is not the maximum, the algorithm will, with probability exactly equal to $1 - 2^{-m}$, output a subset $B \subseteq [n]$ such that $\prod_{j \in B}(x_i - x_j) < 0$. Furthermore, conditioned on the event that the algorithm does output a B, the set $\{j \mid x_j > x_i, j \in B\}$ is a uniform random odd-subset of the set $\{j \mid x_j > x_i, j \in [n]\}$.

Proof The algorithm works as follows. Generate m independent uniform random subsets B_k of $[n] - \{i\}$, $1 \leq k \leq m$. Ask queries $P(i, B_k)$, $1 \leq k \leq m$. If none of the anwsers is

"$<$", the algorithm declares "i is max". Otherwise, the algorithm outputs B_k, where k is the smallest integer with a "$<$" answer to $P(i, B_k)$.

Clearly, if x_i is the maximum element, the algorithm will declare "i is max". If x_i is not the maximum element, then for each k, the probability that a random B_k satisfies $\prod_{j \in B_k}(x_i - x_j) < 0$ is equal to $1/2$. Therefore, the probability for at least one of the m answers being "$<$" is exactly equal to $1 - 2^{-m}$.

Finally, when x_i is not the maximum element, let J be the set of $j \in [n]$ satisfying $x_j > x_i$. Any $B \subseteq [n]$ that includes an odd number of elements from J is equally likely to be the output B. Now, for each $J' \subseteq J$, there are exactly $2^{n-1-|J|}$ sets $B \subseteq [n] - \{i\}$ satisfying the equation $J' = B \cap J$. It follows that the set $\{j \mid x_j > x_i, j \in B\}$ is a uniform random odd-subset of the set J. □

Lemma 3 Let $i \in [n]$ and $B \subseteq [n] - \{i\}$. There is a randomized algorithm using at most $\lceil \log_2 |B| \rceil$ parity tests and satisfying the following properties: for any input $(x_1, x_2, \cdots, x_n)$ satisfying $\prod_{j \in B}(x_i - x_j) < 0$, the algorithm will output an $i' \in B$ with $x_{i'} > x_i$, and furthermore, i' is uniformly distributed over the set $\{j \mid x_j > x_i, j \in B\}$.

Proof We prove by induction. The lemma is obviously true for $n = 2$. Let $n > 2$ and suppose we have proved the lemma for all lesser values of n. Let $(x_1, x_2, \cdots, x_n)$ be any input. The algorithm first generates a uniform random $\lceil |B|/2 \rceil$-subset B_1 of B; let $B_2 = B - B_1$. Make a parity test $P(i, B_1)$.

If the answer is "$<$", then by the induction hypothesis we can find in no more than additional $\lceil \log_2 |B_1| \rceil \leq \lceil \log_2 |B| \rceil - 1$ parity tests a random i' uniformly distributed over the set M, where $M = \{j \mid x_j > x_i, j \in B_1\}$.

Similarly, if the answer is "$>$", we apply the induction hypothesis to find, in no more than additional $\lceil \log_2 |B_2| \rceil \leq \lceil \log_2 |B| \rceil - 1$ parity tests, a random i' uniformly distributed over M, where $M = \{j \mid x_j > x_i, j \in B_2\}$.

Thus, we have found in either case an i' with $x_{i'} > x_i$, using no more than $\lceil \log_2 |B| \rceil$ parity tests. Let $L = \{j \mid x_j > x_i, j \in B\}$. It is easy to verify that the random set M has the property that $\Pr\{M = L'\} = \Pr\{M = L''\}$ for any $L', L'' \subseteq L$ with $|L'| = |L''|$. It follows that i' is uniformly distributed over L. This completes the inductive step. □

We now describe a randomized algorithm for Lemma 1. Let $(x_1, x_2, \cdots, x_n)$ be any input and $i \in [n]$. Using the randomized algorithm in Lemma 2, we make m parity tests and either makes a declaration "i is max" or obtains a $B \subseteq [n] - \{i\}$ satisfying $\prod_{j \in B}(x_i - x_j) < 0$. In the former case, we just make the declaration and halt. In the latter case, we use the randomized algorithm in Lemma 3 to generate an output i'

uniformly distributed over the set $L = \{j \mid x_j > x_i, j \in B\}$. It is straightforward to verify that the algorithm satisfies all the requirements given in Lemma 1. This completes the proof of Theorem 1.

References

[1] M. Rabin, "Proving simultaneous positivity of linear forms," *Journal of Computer and System Sciences* **6** (1972), 639-650.