Randomized Selection Algorithm

Lecture Scribed by: Mohit Singh

February 5, 2003

1 Problem

Selecting the k^{th} ranked element in a given input of n numbers.

We can find the solution in $O(nlog\,n)$ by sorting the elements. There also exists an $O(n)$ deterministic algorithm similar to linear time algorithm for median finding. We will consider some randomized algorithm for the problem.

2 Algorithm I

Input: Set $S = \{x_1, x_2, \ldots, x_n\}, k$
Output: k^{th} ranked element in S.

$find(k, S)$

1. Choose a *random* element $r \in S$.
2. Find the rank of r in S and $L_r = \{x | x \leq r, x \in S\}$ and $U_r = \{x | x > r, x \in S\}$
3. (a) If $rank(r, S) = k$ then got it, return r.
   (b) If $rank(r, S) > k$ then return $find(k, L_r)$.
   (c) If $rank(r, S) < k$ then return $find(k - rank(r, S), U_r)$.

Here, we use a uniform random number generator. Given a range $[a, b]$, it returns an element $x, a \leq x \leq b$ with equal probability $\frac{1}{b-a+1}$ for any x.
Clearly, the recursive algorithm returns the k^{th} ranked element in S. Let us analyze the time taken by this algorithm.
Let a recursive call to $find(r, S)$ be defined as *successful* if $\frac{|S|}{2} \leq rank(r, S) \leq \frac{3|S|}{4}$ and *unsuccessful* or a *failure* otherwise. Clearly, probability of *success* $\geq \frac{1}{2}$ and probability of *failure* $\leq \frac{1}{2}$.

Everytime a successful call takes place we get rid of atleast 3/4 fraction of the elements. Therefore there cannot be more than $log_{\frac{3}{4}} n$ successful calls as we start with n elements. Expected number of calls before we get a successful calls is ≤ 2. To see this, observe that this problem is same as that of finding the number of tosses to take place to find the first head where the probability of head $\geq \frac{1}{2}$.

Expected running time of the selection algorithm I
$=\sum_{i=1}^{log_{\frac{3}{4}} n}$ Expected running time for i^{th} success
$\leq \sum_{i=1}^{\infty} 2 * (\frac{3}{4})^{i-1} n$
$= \frac{7}{2} n.$
where, the inequality holds as after i successful calls, there would be atmost $(\frac{3}{4})^{i} n$ elements left to be searched for and each call takes time equal to the number of elements in the set in which we are searching.

This shows that the expected running time of this algorithm is linear. What we search for in randomized algorithms is stronger guarantee, that the probability of this algorithm performing badly is very small, for example, in this case, we would like to say that the $Prob(T(n) > \alpha cn) < \frac{1}{n^{\alpha}}$ for some constant α which would give us more confidence in the algorithm.
The following inequality helps us prove such claims.
Theorem 1(Markov's Inequality): For a random variable X assuming only non-negative integer values, the following inequality holds,

$$Pr[X \geq k] \leq \frac{E[X]}{k}$$

for any positive integer k.
Proof:

$$E[X] = \sum_{0}^{\infty} tPr[X = t]$$

$$\geq \sum_{k}^{\infty} tPr[X = t]$$

$$\geq k \sum_{k}^{\infty} Pr[X = t]$$

$$\geq kPr[X \geq k]$$

as desired.

Using Markov's Inequality we get that $Pr[T(n) > k\frac{7n}{2}] < \frac{1}{k}$ for any k which is very weak compared to what we wanted. We will get much stronger bounds for the randomized selection algorithm II. Before we discuss algorithm II, we need to introduce Chernoff Hoeffding bounds.

3 Chernoff-Hoeffding Bounds

We will deal with random variable $X = X_1 + X_2 + \ldots + X_n$ where X_i's are poisson trials i.e X_i are independent random variables and X_i take values 1 and 0 only with probability p_i and $1 - p_i$. We will get stronger bounds to the tail probability $Pr[X > s]$ by using the Markov's inequality on the Moment Generating function of X. Clearly, $Pr[X > s] = Pr[e^{tX} > e^{ts}]$ for any $t > 0$. Applying Markov's inequality now we get $Pr[X > s] \leq \frac{E[e^{tX}]}{e^{ts}}$. To get a strong bound, we choose the t which minimizes the RHS which gives us the following theorem.

Theorem(Chernoff-Hoeffding): Let $X = X_1 + X_2 + \ldots + X_n$ where X_i's are Poisson trials with $Pr[X_i = 1] = p_i$ that are independent. Let $\mu = E[X] = E[\sum_i X_i] = \sum_i p_i$. Then

$$Pr[X > (1+\epsilon)\mu] < (\frac{e^{\epsilon}}{(1+\epsilon)^{(1+\epsilon)}})^{\mu} < e^{-\frac{\epsilon^2}{2}\mu}$$

Proof: By applying Markov's inequality we get

$$\begin{aligned} Pr[X > (1+\epsilon)\mu] &\leq \frac{E[e^{tX}]}{e^{t(1+\epsilon)\mu}} \\ &= \frac{E[e^{t\sum_i X_i}]}{e^{t(1+\epsilon)\mu}} \\ &= \frac{E[\prod_i e^{tX_i}]}{e^{t(1+\epsilon)\mu}} \\ &= \frac{\prod_i E[e^{tX_i}]}{e^{t(1+\epsilon)\mu}} \end{aligned}$$

where, the last equality holds because X_i's are independent random variables. Now,

$$\begin{aligned} E[e^{tX_i}] &= (1-p_i)e^{t.0} + p_i e^{t.1} \\ &= 1 - p_i + p_i e^t = 1 + p_i(e^t - 1) \\ &\leq e^{p_i(e^t-1)}. \end{aligned}$$

This implies that

$$\begin{aligned} \frac{\prod_i E[e^{tX_i}]}{e^{t(1+\epsilon)\mu}} &\leq \frac{e^{(e^t-1)(\sum_i p_i)}}{e^{t(1+\epsilon)\mu}} \\ &= \frac{e^{(e^t-1)\mu}}{e^{t(1+\epsilon)\mu}} \end{aligned}$$

which holds for any t. Now, choose $t = ln(1+\epsilon)$ to get

$$Pr[X > (1+\epsilon)\mu] < [\frac{e^{\epsilon}}{(1+\epsilon)^{1+\epsilon}}]^{\mu}$$

proving the first inequality.

4 Selection Algorithm II

We now present a set of related randomized algorithms for selection and their analysis.

Algorithm II

1. Choose a random subset $R \subset S$ by selecting each element in S with probability $r = \frac{c}{\sqrt{n}}$.

2. Sort R and choose $l = max(1, \frac{kc}{\sqrt{n}} - 2\sqrt{c}n^{1/4})$ and $r = min(n, \frac{kc}{\sqrt{n}} + 2\sqrt{c}n^{1/4})$. If $k < n^{\frac{3}{4}}$ then choose $l = 1$ and if $k > n - n^{\frac{3}{4}}$ then choose $r = n$. Let X_l and X_r denote the l^{th} and r^{th} element in R. Find $L_l = \{x | x < X_l, x \in S\}$ and $L_r = \{x | x > X_R, x \in S\}$

3. (a) if $|L_l| > k$ then sort S and return the k^{th} element.(Net did not catch the k^{th} element.

   (b) if $|L_r| < n - k$ then sort S and return the k^{th} element.(Net did not catch the $k^t h$ element.

   (c) else Sort $S - L_r - L_l$ and return the $(k - |L_l|)^{th}$ element of $S - L_l - L_r$.

We say that step 2 *fails* if either of $3(a)$ or $3(b)$ is true or if net is too loose i.e.,$|S - L_r - L_l| > n^{3/4}$. We can get other variants of Algorithm II by changing step 3. For instance, we can repeat steps 1 and 2 incase we observe that step 2 fails. We will analyze variant 1 and other variants are very similar to analyze. Observe that step 2 takes $2n$ time and step 3 takes $n^{3/4} logn^{3/4} = o(n)$ if step 2 does not fail. Also, if the step 2 fails, then we take $O(nlog\, n)$ time. Hence, if the probability of step 2 failing is pretty low then we will be able to show good bounds for the algorithm. Let p_1' and p_2' denote the probability of event $3(a)$ and $3(b)$ respectively. Let p' denote the probability that net is loose. Clearly, the probability of step 2 failing $\leq p_1' + p_2' + p'$.

If the net fails to catch x_k then either we have chosen very few elements less than x_k or we have chosen too many elements greater than x_k. We will calculate the probability p_1 that we chosen less than l elements less than x_k. Clearly, then $k \geq n^{\frac{3}{4}}$ else we would have set $l = 1$ and cannot have missed x_k.

p_1'=Pr[number of elements in the interval $x_1, x_2, \ldots, x_k \leq c\sqrt{n} - 2\sqrt{c}n^{\frac{1}{4}}$]$\leq$

$$e^{\frac{-2\sqrt{c}^2\sqrt{n}}{kc/\sqrt{n}}}$$

$$= e^{\frac{-4n}{k}}$$

$$\leq e^{-4n^{\frac{1}{4}}}$$

where, the first inequality holds due to chernoff bounds and the last inequality holds as $k > n^{\frac{3}{4}}$.
p'_2 can be bounded by exactly the same argument to give that the probability that the net misses $x_k \le \frac{1}{2^{\epsilon n^{\frac{1}{4}}}}$ for some constant ϵ.

Now, we will try to bound the probability that the net is large. Clearly, the number of elements chosen $\le$ Maximum number of elements in an interval of R * number of intervals chosen. We will now bound the maximum number of elements in an interval of R. Given a $R \subset N$, where $|N| = n$.
Let σ denote an interval $[x_i, x_j]$ among all possible $\binom{n}{2}$ intervals. Let $p(\sigma, R)$ denote the probability that no elements in $\sigma \bigcap N$ is chosen in R given that the endpoints of σ are chosen in R.
We say that σ is *Bad* if $l(\sigma) \ge \alpha \frac{n}{R} lnt$ for some constants α and t.
Probability that *Bad* σ survives in R(i.e., none of the $l(\sigma) \ge \alpha \frac{n}{R}\ lnt$ elements have been chosen) $\le (1 - \frac{R}{n})^{\alpha \frac{n}{R} lnt} \le e^{-\alpha lnt} = t^{-\alpha}$
Let $q(\sigma, R)$ denote the probability that endpoints of σ are chosen.
Probability that a *Bad* σ survives $\le t^{-\alpha} q(\sigma, R)$.
Hence, the probability that any of *Bad* σ survives $\le \sum_\sigma t^{-\alpha} q(\sigma, R)$

$$= t^{-\alpha} \sum_\sigma q(\sigma, R)$$

$$= t^{-\alpha}.(\ Expexted\ No\ of\ intervals\ in\ R)$$

$$\le t^{-\alpha} R^2$$

For example, now put $t = n$ and $\alpha = 4$ to get
max gap $\le 3 \frac{n}{R} ln\ n$ with probability $\ge 1 - \frac{1}{n^2}$.
For selection we have chosen $4\sqrt{c} n^{\frac{1}{4}}$ intervals.
Hence, total number of elements in the net $\le 12\sqrt{c} n^{\frac{3}{4}} ln\ n$ with probability $\ge 1 - \frac{1}{n^2}$.

Hence, the probabilty that step 2 fails $\le p'_1 + p'_2 + p' \le \frac{2}{n^2}$.
So, the algorithm works in time $2n + o(n)$ with probability $1 - \frac{2}{n^2}$ and also the expected running time is also $2n + o(n)$.