CPSC 500: Fundamentals of Algorithm Design and Analysis
Oct. 2, 2013.
Scribe: Fang Yuan Chi

1. Overview

In the first part of this lecture, we continued our discussion on the K-select problem. We first reviewed the selection via random sampling approach, which was introduced during the last lecture. We then discussed the running time and the probability of failure for this approach. In the second part of this lecture, we talked about two different methods, particularly the decision tree and adversary argument, on finding the lower bound of finding the median.

2. Selection via random sampling

2.1 Review of basic algorithm

Let $S = \{S_1, S_2, S_3, \dots, S_n\}$ be a sequence of n distinct numbers. The K-select algorithm takes S as an input and outputs $S_{(k)}$, the k^{th} smallest number in S, which can be achieved by an algorithm called selection via random sampling (with the assumption that k ∈ $[n^{1/4}, n - n^{1/4}]$). Before we do any further analysis on this algorithm, we first take a closer look at the steps this algorithm takes in order to compute the output, as follows:

Step 1: Select $n^{3/4}$ numbers from S at random (with replacement). Call the resulting sequence R.

Step 2: Sort R.

Step 3: Introduce two variables, a and b:

- Let variable a = $\left(\frac{k}{n^{1/4}} - \sqrt{n}\right)^{th}$ smallest number in R.
- Let variable b = $\left(\frac{k}{n^{1/4}} + \sqrt{n}\right)^{th}$ smallest number in R.

Step 4: Partition S into S_L, S_O and S_R, where

$$S_L = \{x \in S | x < a\}$$
$$S_O = \{x \in S | a \leq x \leq b\}$$
$$S_R = \{x \in S | x > b\}$$

Step 5: If $|S_L| \geq k$ *then FAIL*.

Step 6: If $|S_L| + |S_O| < k$ *then FAIL*.

Step 7: If $|S_O| > 4n^{3/4}$ *then FAIL*.

Step 8: Sort S_O and return $(k - |S_L|)^{th}$ smallest from S_O.

Note that by *FAIL* we mean the algorithm goes back and starts from step 1 again. The size and numbers in R are different every time we run the algorithm since we randomly select the numbers to form R. Also, we may select the same number in S more than once, where in this case, we simply replace the existing number with the new number.

The next thing we want to know is how well this algorithm performs. That is, how many times the algorithm has to run in order to get the reasonable output? The answer to this question depends on the probability of failure as the algorithm goes back and starts from step 1 if it fails.

2.2 An Examination into probability of failure

Failure at steps 5, 6 and 7 are mutually exclusive and not dependent on the result of one another. We fail at either step 5, 6 or 7 each time we run the algorithm. Therefore, the overall probability of failure will be the sum of probability of failure at each step. With this in mind, we begin to look at the failure at each step separately.

2.2.1 Failure at step 5

Failure occurs at step 5 if and only if $|S_L| \geq k$. This means that $a > S_{(k)}$, which means our sample R contains less than $\frac{k}{n^{1/4}} - \sqrt{n}$ numbers that are $\leq S_{(k)}$ (Figure 1).

Figure 1

Suppose that X is the number of samples in R that are $\leq S_{(k)}$. We know that if $X < \frac{k}{n^{1/4}} - \sqrt{n}$, the algorithm fails. Thus, the probability of failure at step 5 is essentially

$$P[|S_L| \geq k] = P\left[X < \frac{k}{n^{1/4}} - \sqrt{n}\right]$$

We can use Chebyshev's inequality to compute this probability. Recall from last lecture that in order to use Chebyshev's inequality, we first need to compute the expectation, variance, and standard deviation of X.

Let's first look at the expectation of X. Suppose that we have an indicator random variable x_i (i_{th} *sample that was chosen in step* 1), where

$$x_i = \begin{cases} 1, & x_i \leq S_{(k)} \\ 0, & x_i > S_{(k)} \end{cases}$$

and

$$X = \sum_{i=1}^{n^{3/4}} x_i$$

Since we know that $E[x_i = 1] = \frac{k}{n}$, we can compute $E[X]$.

$$E[X] = \sum_{i=1}^{n^{3/4}} E[x_i] = \sum_{i=1}^{n^{3/4}} \frac{k}{n} = n^{3/4}\frac{k}{n} = \frac{k}{n^{1/4}}$$

We now have the expectation and need to compute the variance of X. We first compute variance of x_i

$$Var[x_i] = E[x_i^2] - E[x_i]^2 = \frac{k}{n} - \left(\frac{k}{n}\right)^2$$

and

$$Var[X] = \sum_{i=1}^{n^{\frac{3}{4}}} Var[x_i] = n^{3/4}\left(\frac{k}{n} - \left(\frac{k}{n}\right)^2\right) = n^{3/4}\left(\frac{k}{n}\right)\left(1 - \frac{k}{n}\right)$$

We see that this equation is maximized if we assign $k = \frac{n}{2}$. Substitute $k = \frac{n}{2}$ into $Var[X]$, and we get $Var[X] \leq \frac{1}{4}n^{3/4}$.

Both variance and expectation are derived and we still need to derive standard deviation in order to apply Chebyshev's inequality. We expect $X = \frac{k}{n^{1/4}}$, but what we really have is $X < \frac{k}{n^{1/4}} - \sqrt{n}$. Thus, the standard deviation is $\sqrt{n}$. At this point, we have everything we need; we can proceed with Chebyshev's inequality.

$$P\left[X < \frac{k}{n^{1/4}} - \sqrt{n}\right] \leq P\left[\left|X - \frac{k}{n^{1/4}}\right| \geq \sqrt{n}\right] \leq \frac{\frac{n^{3/4}}{4}}{n} = \frac{1}{4n^{1/4}}$$

2.2.2 Failure At Step 6

Step 6 is the opposite of step 5. The algorithm fails at this step if and only if fewer sample chosen where a $> S_{(k)}$. By applying a similar argument, we have

$$P[|S_L| + |S_O| < k] \leq \frac{1}{4n^{1/4}}$$

2.2.3 Failure At Step 7

We fail at step 7 if and only if $|S_O| > 4n^{3/4}$, which means that the number of samples between a and b is greater than $4n^{3/4}$. If this happens, then we have either $a \leq S_{(k-2n^{3/4})}$ or $b \geq S_{(k+2n^{3/4})}$(Figure 2).

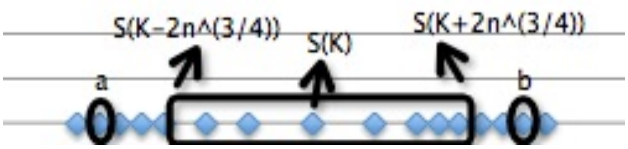

Figure 2

First we compute the expectation of Y, number of samples that are $< S_{(k-2n^{3/4})}$

$$E[Y] = \frac{k - 2n^{3/4}}{n^{1/4}} = \frac{k}{n^{1/4}} - 2\sqrt{n}$$

We know that a is the $\left(\frac{k}{n^{1/4}} - \sqrt{n}\right)^{th}$ smallest number in R and $a \leq S_{(k-2n^{3/4})}$. These facts imply that we have at least $\frac{k}{n^{1/4}} - \sqrt{n}$ number of variables that are $\leq S_{(k-2n^{3/4})}$. However, as we just computed and saw that we only expect $\frac{k}{n^{1/4}} - 2\sqrt{n}$ variables that are $\leq S_{(k-2n^{3/4})}$, we know that we have at least $\sqrt{n}$ more samples that are $\leq S_{(k-2n^{3/4})}$ than what we expected.

Then, we apply the Chebyshev's inequality.

$$P\left[a \leq S_{(k-2n^{3/4})}\right] \leq P\left[|Y - E[Y]| \geq \sqrt{n}\right] \leq \frac{1}{4n^{1/4}}$$

and, similar arguments apply to b

$$P\left[b \geq S_{(k+2n^{3/4})}\right] \leq \frac{1}{4n^{1/4}}$$

Combine these terms together, and we have

$$P\left[|S_O| > 4n^{3/4}\right] \leq \frac{1}{4n^{1/4}} + \frac{1}{4n^{1/4}} = \frac{1}{2n^{1/4}}$$

2.2.4 Total probability of failure

We can now get the total probability of failure by combining the probability of failure at each step.

$$P[FAIL] \leq \frac{1}{4n^{1/4}} + \frac{1}{4n^{1/4}} + \frac{1}{2n^{1/4}} = \frac{1}{n^{1/4}}$$

This implies that as n gets larger, the probability of failure gets smaller (approaches to 0), which means the number of times the algorithm has to run to get a result gets smaller (approaches to 1).

2.4 Running time of the algorithm

The most costly step in the algorithm is step 4, the partition of S. As each variable in S has to compare to both a and b in order to decide which subset it belongs to. Thus, the running time for this step is $2n$. The running time for the rest of the steps is $o(n)$. Combining these terms together we get the running time of the algorithm to success at first time to be $2n + o(n)$. Considering the fact the algorithm may fail, we compute the running time:

$$E[T(n)] = 2n + o(n) + \frac{1}{n^{1/4}} E[T(n)]$$

Note that the last term of the running time is actually $o(n)$. Substitute this fact into the running time. We get $E[T(n)] = 2n + o(n)$.

3. Lower Bound on finding median

The running time of an algorithm has an upper and lower bound, which can be proven mathematically. The lower bound of a problem states that for a given problem, every algorithm that can exist for the problem has a worst-case running time that is lower bounded by the given bound. In other words, every algorithm that solves the problem has a worst-case running time $\Omega(f(n))$, and no algorithm that solves the problem runs in $o(f(n))$ time.

3.1 Decision Tree

There are many different sorts of algorithm out there and we won't be able to use the same technique to analyze all algorithms. In this lecture, we take sorting algorithm as an example, and the approach we used is a decision tree. In sorting algorithm, which sorts a sequence of n different numbers, we compare each value to other values in the input set. When the algorithm terminates, there will be $n!$ number of leaves, and the depth will be $\log n!$, which is the lower bound of the worst-case running time of the algorithm.

3.1.1 Go back to finding the median

Note that the K-select problem is essentially a finding median problem if we assign $k = n/2$. If we use decision tree on K-select problem, the number of leaves in decision tree will be $\geq n$ and the depth will be $\geq \log n$. It is a true lower bound as we follow the decision tree. However, it is also lame. In order to find the median, we need to look at each and every variable in input sequence, which will take at least n steps. With the fact that $\log n < n$, we proved that $\log n$ is actually a lame lower bound.

3.2 Adversary Argument

Another technique that can be used to find the lower bound is called adversary argument. The adversary argument pretends that there is some sort of "opponent" which works against the algorithm provided, and its goal is to provide the most work for the algorithm. In our case, at each step of the decision tree, the adversary provides an answer to the algorithm that will result in a maximum amount of work. Using this method, we can find the lower bound of an algorithm. By following the adversary argument, the lower bound on finding the median will be $\frac{3}{2}n - 2$. However, we did not have enough time to explain how do we get this lower bound. Hopefully this will be explained in the future.

4.0 Conclusion

In this lecture, we focused mainly on selection via random sampling. We found the running time of the problem by carefully examining the probability of failure and the running time for each step that the algorithm takes. We then talked very briefly about the lower bound on finding the median. Two approaches are introduced on finding the lower bound: decision tree and adversary argument. We found that by using decision tree, what we get is a true but lame lower bound. With adversary argument, the lower bound will be $\frac{3}{2}n - 2$. However, we did not get a chance to talk about adversary argument in detail.

Reference:

[1] http://www.cs.uiuc.edu/~jeffe/teaching/algorithms/
[2] http://en.wikipedia.org/wiki/Selection_algorithm
[3] http://en.wikipedia.org/wiki/Big_O_notation