CPSC 500
September 30, 2013

Scribe: Alireza Shafaei

Contents

1 Introduction

Probabilistic analysis of algorithms is an estimation method for computation complexity of computational problems. We usually start with making assumptions about the probability distribution of input and then, try to derive the complexity of the proposed algorithm.
In another approach which is usually refered to as 'Randomized Algorithm', we try to add randmoness to algorithm's logic to some degree. The proposed idea gives rise to two groups of algorithms: (i) Las Vegas Algorithms, (ii) Monte Carlo algorithms. Las Vegas algorithms always give correct answers, but vary in running time due to the random choices the algorithm takes. In turn, Monte Carlo algorithm may produce an incorrect solution, but the probability of the incorrect solution could be bounded[1].
After finishing the probabilistic analysis of a certain problem called 'Coupon Collector's Problem', we'll introduce a randomized algorithm to solve the K-selection problem which defeats determnistic algorithms in terms of running time. It could be further proved that the proposed algorithm is in fact faster than any deterministic algorithm by proving a lower bound of time over the K-Selection problem for deterministic approach!

[1]For a more detailed description and some interesting examples, you can check [1], chapter 1.

2 Overview

The first section of these notes is a review of probability theory to the extent of our need. The reader is strongly encouraged to completely understand the presented material before proceeding to the next parts.
During the previous lecture we introduced the Coupon Collector's Problem [2]. As only the preliminary analysis of this problem was discussed, we're going to finish the analysis and then proceed to another problem.
The second problem that we're going to go through is the problem of finding the K_{th} smallest number in an input array. One popular deterministic solution to this problem, which runs in $10n = O(n)$ in average, will be discussed; and then a randomized algorithm with a better constant factor that runs in $1.5n = O(n)$ will be analyzed.

3 Short Review of Probability Theory

3.1 Expected Value, Variance and Independency

If we have X as discrete random variable which takes values $x_1, x_2, x_3, \dots, x_n$ with probabilities of $p_1, p_2, p_3, \dots, p_n$ respectively, then the expected value[3] of this random variable is defined as follows:

$$\mu = \mathbb{E}[X] = \sum_{i=1}^{n} x_i \, p_i, \tag{1}$$

Expected value is a linear function, meaning that $\mathbb{E}[aX + b] = a\mathbb{E}[X] + b$.
X and Y are called independent if

$$\mathbb{P}(X = x \text{ and } Y = y) = \mathbb{P}(X = x).\mathbb{P}(Y = y) \tag{2}$$

Variance of random variable X, which in a sense is a measure of deviation from the mean, is described as:

$$\sigma^2 = \mathrm{Var}[X] = \mathbb{E}[(X - \mathbb{E}[X])^2] \tag{3}$$

And finally, if random variables X and Y are independent, the following statement is true:

$$Var[X + Y] = Var[X] + Var[Y] \tag{4}$$

3.2 Markov's Inequality

If Z is a nonnegative r.v, and $a > 0$ then:

$$\mathbb{P}(Z \geq a) \leq \frac{\mathbb{E}[Z]}{a}. \tag{5}$$

3.3 Chebyshev's Inequality

With Chebyshev's inequality and the variance we can find a bound on the probability that a r.v. is far from the E.V. Formally, the Chebyshev's inequality is defined as follows:

$$\mathbb{P}(|X - \mathbb{E}[X]| \geq a) \leq \frac{\sigma^2}{a^2} \text{ or alternatively } \mathbb{P}(|X - \mathbb{E}[X]| \geq t\sigma) \leq \frac{1}{t^2} \tag{6}$$

[2]The problem of collecting all the coupons in a contest in order to win; for more information see [2].
[3]Also refered to as EV, mean and first moment

Proof. Let $Z = (X - \mathbb{E}[X])^2$ then $\mathbb{E}[Z] = \sigma^2$. By Markov's inequality, we'd have:

$$\mathbb{P}(Z \geq a^2) \leq \frac{1}{a^2}\sigma^2 \tag{7}$$

$$\text{and also } \mathbb{P}(Z \geq a^2) = \mathbb{P}((X - \mathbb{E}[X])^2 \geq a^2) = \mathbb{P}(|X - \mathbb{E}[X]| \geq a) \tag{8}$$

$$\text{therefore, } \mathbb{P}(|X - \mathbb{E}[X]| \geq a) \leq \frac{\sigma^2}{a^2} \tag{9}$$

This means if you can find the mean and variance of a distribution, you can also have an upper bound of the probability of getting far away from the mean by value of a. As you'd see in the future, this helps us analyze behaviour of randomized algorithms more clearly.

4 Coupon Collector's Analysis Continued

As we've seen before, the expected number of tries to collect all the N coupons was approximately $n \ln n$. We defined r.v. $Y = X_0 + X_1 + \ldots + X_{n-1}$, X_i denoting the probability of getting a new coupon after having collected i coupons. In other words, we found $\mathbb{E}[Y]$ to be $n \ln n$. These variables X_i are mutually idependent, because they only depend on your current state rather than your history. Thus using equation 4 we have:

$$Var[Y] = Var[X_0] + Var[X_1] + \ldots + Var[X_{n-1}] \tag{10}$$

X_i being a Bernoulli trial with success probability $p = \frac{n-i}{n}$, for $Var[X_i]$ and $Var[Y]$ we'd have:

$$Var[X_i] = \frac{1-p}{p^2} \Rightarrow Var[X_i] = \frac{ni}{(n-i)^2} \tag{11}$$

$$\text{therefore, } Var[Y] = \sum_{i=0}^{n-1} \frac{ni}{(n-i)^2} \Rightarrow Var[Y] = n(n\sum_{i=1}^{n}\frac{1}{i^2} - \sum_{i=1}^{n}\frac{1}{i}) \tag{12}$$

$$\text{given that, } \lim_{i \to +\infty}(\sum_{i=1}^{n}\frac{1}{i^2}) = \frac{\pi^2}{6} \tag{13}$$

$$Var[Y] \simeq \frac{n^2\pi^2}{6} \tag{14}$$

So $Var[Y]$ is asymptomically $O(n^2)$.
Now that we have $Var[Y] \simeq n^2\frac{\pi^2}{6}$ and $E[Y] = nH_n$[4], using Chebyshev's inequality we can say:

$$\mathbb{P}(Y \geq b\mathbb{E}[Y]) = \mathbb{P}(Y \geq bnH_n) \tag{15}$$

$$\mathbb{P}(Y \geq bnH_n) \leq \mathbb{P}[|Y - nH_n| \geq (b-1)nH_n] \tag{16}$$

$$\mathbb{P}[|Y - nH_n| \geq (b-1)nH_n] \leq \frac{n^2\frac{\pi}{6}}{(b-1)^2n^2H_n^2} \simeq \frac{\pi^2}{6(b-1)^2(\ln n)^2} \tag{17}$$

This means that the probability of number of trials being b times more than $n \ln n$ decreases as n increases. So as the n increases, it's more likely to collect n coupons with just $n \ln n$ trials.

5 K-Selection

Let's say we have n distinct real numbers $S[1...n]$. The goal is to find the K_{th} smallest number in S. The easiest way to do so, is by sorting the array and then returning the K_{th} element, but that will take $O(n \lg n)$; can we do better? The answer is yes.

[4] H_n is sum of first n harmonic numbers, i.e : $\sum_{i=1}^{n}\frac{1}{i}$

5.1 Using Quick Sort

We can do in $O(n)$ by slightly modifying the quick sort algorithm. As you recall in quick sort, we first pick a pivot and then move all the elements less than pivot to left, and all the elements bigger than pivot to right side of the array. This can be done in linear time. We then recursively do the same thing for the two subsequent subarrays until the length of array is one.
Given that, in order to find the K_{th} element we don't need to process the subarray which does not include K_{th}; the idea is to just continue the process with the subarray which includes the K_{th} element. The new algorithm will look as follows:

```
function select(array $S$, number $K$):
if $|S| = 1$
  return $S[1]$;
$P$ = pickPivot($S$);
$L$ = $\{S[i]|S[i] < P\}$;
$R$ = $\{S[i]|S[i] \geq P\}$;
if $|L| \geq K$
  return select(L, K);
else if $|L| = K - 1$
  return $P$;
else return select($R$,$K - |L| - 1$);
```

We have two options for function call pickPivot. (i)Pick a random element (ii)Good pivot. In order to pick a good pivot we can modify the input into $\frac{n}{5}$ groups of 5 numbers, then find the median within each and then return the median within medians.
The running time of the algorithm is $T(n) \leq T(\frac{n}{5}) + n + T(\frac{7n}{10}) \to T(n) \leq n + T(\frac{9}{10}n)$ which is almost $10n$. It's generally a good thing to be able to select K_{th} smallest number in linear time because we can use it in other algorithms as a building block.

5.2 Selection via Random Sampling

First we assume k is resonably big $K \in [n^{\frac{1}{4}}, n - n^{\frac{1}{4}}]$. if not, we can use delete on a minheap or maxheap k times. Heapification could take place in $\theta(n)$, and delete takes $O(\lg n)$, so it could be done in $n + O(k \lg(n))$

The idea is to reduce the number of inputs by randomly picking a subset of the input of length $n^{\frac{3}{4}}$. Note that we can sort the subset in sublinear time $o(n)$. After sorting this new set, we'd try to find good estimates for the K_{th} number. More specifically, we'd expect the K_{th} number to fall between $(\frac{K}{n^{\frac{1}{4}}} - \sqrt{n})_{th}$ and $(\frac{K}{n^{\frac{1}{4}}} + \sqrt{n})_{th}$ number in subset. Naming those two bounds a and b, we'd partition the original input based on these numbers into three groups S_L, S_0 and S_R. We'd be expecting the K_{th} number to be in S_0, but there's a low probability that we pick bad numbers a and b. After two tests, we're able to determine whether the K_{th} number is in S_0 or not, and the final test checks whether S_0 array is sufficiently small. After sorting the S_0 we'd return the appropriate value which is identified to be the K_{th} number in the original set.

```
function selectionViaRandomSampling(array $S$, number $K$):
R = select $N^{\frac{3}{4}}$ numbers from S at random (with replacement)
Sort R on $\frac{3}{4}n^{\frac{3}{4}} \lg n = o(n)$ time
let a be the ($\frac{K}{n^{\frac{1}{4}}} - \sqrt{n}$) smallest in R
```

```
   let b be the $(\frac{K}{n^{\frac{1}{4}}} + \sqrt{n})$ smallest in R
4  Partition original array S into
       $S_L = \{x \in S | x < a\}$
       $S_0 = \{x \in S | a \leq x \leq b\}$
       $S_R = \{x \in S | b < x\}$
5  if $|S_L| \geq K$ then FAIL
6  if $|S_R| + |S_0| < K$ then FAIL
7  if $|S_0| > 4n^{\frac{3}{4}}$ then FAIL
8  Sort $S_0$ and return $(K - |S_L|)$ smallest from $S_0$
```

During the next lecture we'd analyze running time and failure probability of this algorithm.

6 Conclusions

During this lecture, we finished the analysis of Coupon Collector's problem and started discussing a new problem called K-Selection problem. We described a deterministic algorithm to solve the latter problem and proposed a randomized algorithm. In future, we'd analyze the proposed algorithm and will demonstrate that this randomized algorithm performs better than any possible deterministic algorithm for this problem.

References

[1] Motwani, R., and Raghavan, P. *Randomized algorithms.* Cambridge University Press, New York, NY, USA, 1995.

[2] Wikipedia. Coupon collector's problem. `http://en.wikipedia.org/wiki/Coupon_collector's_problem`, October 2013.