Midterm Solutions

Ashraf Osman
Department of Computer Science and Electrical Engineering,
West Virginia University,
Morgantown, WV
osman@csee.wvu.edu

1

Given: Experiment of tossing a coin, which has a probability p of turning up HEADS and $1-p$ of turning up TAILS on any one toss.
Required:

- Random variable to characterize the number of tosses required to obtain n heads,
- PMF of this random variable,
- Mean,
- Standard deviation.

Let X_i be a random variable to indicate the number of tosses required until we get a head in the first trial. Let X be the total number of tosses required to obtain n heads. Accordingly, X is the summation of all the n trials.Therefore,

$$X = X_1 + X_2 + ... + X_n.$$

X_i is a geometric random variable which represents the total number of Bernoulli trials required to get the first successful result (Heads).

The probability mass function of X_i is given by:

$$Pr\left(X_i = n\right) = (1-p)^{n-1} p$$

The expectation of X_i is given by:

$$\begin{aligned} E\left[X_i = n\right] &= \sum_{n=1}^{\infty} np\left(1-p\right)^{n-1} \\ &= \frac{p}{1-p}\sum_{n=1}^{\infty} n\left(1-p\right)^{n} \qquad \text{,but } \sum_{i=0}^{\infty} ic^i = \frac{c}{\left(1-c\right)^2}, |c| < 1 \\ &= \frac{p}{1-p} \times \frac{1-p}{\left(1-(1-p)\right)^2} \\ &= \frac{1}{p} \end{aligned}$$

The Variance of X_i is given by:

$$Var[X_i] = E[{X_i}^2] - E[X_i]^2$$

$$E[{X_i}^2] = \sum_{n=1}^{\infty} n^2 p\left(1-p\right)^2, \qquad \text{let } q = 1-p$$

$$
\begin{aligned}
&= p\sum_{n=1}^{\infty} n^2 q^{n-1}, && \frac{d\left(nq^n\right)}{dq} = n^2 q^{n-1} \\
&= p\sum_{n=1}^{\infty} \frac{d}{dq}\left(nq^n\right) = p\frac{d}{dq}\left(\sum_{n=1}^{\infty} nq^n\right) \\
&= p\frac{d}{dq}\left(\frac{q}{(1-q)^2}\right) && dq = -dp \\
&= -p\frac{d}{dq}\left(\frac{1-p}{p^2}\right) = -p\left(\frac{-2}{p^3} + \frac{1}{p^2}\right) \\
&= \frac{2-p}{p^2}
\end{aligned}
$$

Therefore, the variance is:

$$
\begin{aligned}
Var[X_i] &= E[{X_i}^2] - E[X_i]^2 \\
&= \frac{2-p}{p^2} - \left(\frac{1}{p}\right)^2 \\
&= \frac{1-p}{p^2} \\
&= \frac{q}{p^2}
\end{aligned}
$$

Therefore the standard deviation is:

$$\sigma_{X_i} = \sqrt{\frac{q}{p^2}}$$

Therefore, X is the random variable we use to study the number of tosses required to obtain n heads. X is the summation of geometric random variables that characterize the number of tosses required to obtain one head only.

$$X = X_1 + X_2 + ... + X_n.$$

The expectation of X is given by:

$$
\begin{aligned}
E[X] &= E\left[\sum_{i=1}^{n} X_i\right] = \sum_{i=1}^{n} E[X_i], && \text{from linearity of expectation} \\
&= \sum_{i=1}^{n} \frac{1}{p} \\
&= \frac{n}{p}
\end{aligned}
$$

The standard deviation of X is given by: Each trial is independent on all the other trials. Therefore, we may use the theorem which states, for independent random variables, the variance of the sum is equal to the sum of the variances.

$$
\begin{aligned}
Var[X] &= Var\left[\sum_{i=1}^{n} X_i\right] = \sum_{i=1}^{n} Var[X_i] \\
&= \sum_{i=1}^{n} \frac{q}{p^2} \\
&= \frac{nq}{p^2}
\end{aligned}
$$

Therefore, the standard deviation of X is:

$$\sigma_{X_i} = \sqrt{\frac{nq}{p^2}}$$

2

Given: Select the smallest k^{th} element of an array using partition function.
Required:
Calculate the worst case complexity of the algorithm. If partition function is implemented using randomization, show that the expected time taken by the algorithm is $O(n)$.

The worst case occurs if partitioning n elements produces a sequence with one element and a sequence with $n-1$ elements. Then the total number of comparisons (time taken) maybe given by:

$$\begin{aligned} T(n) &= (n-1) + T(n-1) \\ T(n-1) &= (n-2) + T(n-3) \\ T(n) &= (n-1) + (n-2) + (n-3) + \ldots + 1 \\ T(n) &= \frac{n(n-1)}{2} \\ T(n) &= O(n) \end{aligned}$$

Consider a general case, in which the random pivot is equally likely to select the i^{th} ranked element in the array. Firstly, it takes $(n-1)$ comparisons to partition the array into two partitions. Thus all other partitions $(0:n-1), (1:n-2), \ldots, (n-1:0)$ occur with probability of $\frac{1}{n}$. We will only recurse the partition that contains the k^{th} smallest element. This could be any one of the two partitions. To give an upper bound, we will assume that k is always in the bigger partition.
Let $E(n)$ be the random variable corresponding to the running time (number of comparisons) of randomized select for an input of length n. Then,by summing the larger partitions for upper boundary we obtain:

$$E(n) \leq (n-1) + \frac{2}{n} \sum_{i=\frac{n-1}{2}}^{n-1} E(i)$$

There are a number of methods to solve this recursion, like "guess and check", and "induction" methods.
Using the induction method:
Let $B(n) = cn$ where c is a constant bigger than 1. We want to prove that if we choose c large enough then $E(n) \leq B(n)$ for every n, and so $E(n) = O(n)$.
For $n = 1, E(1) = 1 < c$, therefore $E(1) < B(1)$.
Assume $E(n') \leq B(n')$ for $n' < n$. Required to prove that $E(n) \leq B(n)$.

$$\begin{aligned} E(n) &\leq (n-1) + \frac{2}{n} \sum_{i=\frac{n-1}{2}}^{n-1} B(i) \\ &\leq n + \frac{2}{n} \sum_{i=\frac{n}{2}}^{n} ci \\ &\leq n + \frac{2c}{n} \left(\frac{n}{2}\right) \left(\frac{n}{2} + n\right) \\ &\leq n + \frac{3cn}{2} \\ &\leq \left(1 + \frac{3c}{2}\right) n \end{aligned}$$

Therefore, $E(n) \leq B(n)$. Thus, the expected time taken by the algorithm is $E(n) = O(n)$.

3

Definition of RP: The class RP consists of all languages L that have a randomized algorithm A_1 running in worst-case polynomial time such that for any input $x \in \sum^*$,

$$x \in L \Rightarrow Pr[A_1(x) accepts] \geq \frac{1}{2}.$$
$$x \notin L \Rightarrow Pr[A_1(x) accepts] = 0.$$

RP algorithm can only err when $x \in L$ with a probability $< \frac{1}{2}$.

Definition of coRP: The class $coRP$ consists of all languages L that have a randomized algorithm A_2 running in worst-case polynomial time such that for any input $x \in \sum^*$,

$$x \in L \Rightarrow Pr[A_2(x) accepts] = 1.$$
$$x \notin L \Rightarrow Pr[A_2(x) accepts] < \frac{1}{2}.$$

$coRP$ algorithm can only err when $x \notin L$ with a probability $< \frac{1}{2}$.

Definition of ZPP Class The class ZPP is a class of languages that have Las Vegas algorithms running in expected polynomial time. ZPP algorithms do not err at all.

Consider the following algorithm A', to decide L. Given any arbitrary input x, run the RP algorithm A_1 and then on the $coRP$ algorithm A_2. If A_1 accepts, declare $x \in L$. If A_2 rejects, declare $x \notin L$. Otherwise, repeat the execution with the same input x. It is easy to see that algorithm A' uses other random algorithms for deciding language L , but it does not give any errors. In order to prove that this A' algorithm belongs to class ZPP, we have to prove that it decides the language in polynomial time.

Let P_1 and P_2 represent the time taken by algorithm A_1 and A_2 to execute respectively. For each trial, A_1 and A_2 are executed and evaluated. The trials terminiate if A_1 accepts or A_2 rejects, i.e. we repeat the trials only if A_1 rejects and A_2 accepts. Hence, the probabilty of terminating in the first trial is $\frac{3}{4}$. It is required to calculate the expected number of trials required to reach a decision. Let X be a geometric random variable that represents the total number of trials executed to reach a decision with a probability of success $p = \frac{3}{4}$. This is shown as follows:

$$\text{Algorithm terminates after } (P_1 + P_2) \text{with a probability of } \tfrac{3}{4}$$
$$\text{Algorithm terminates after } 2(P_1 + P_2) \text{with a probability of } \left(\tfrac{1}{4}\right)\tfrac{3}{4}$$
$$\text{Algorithm terminates after } 3(P_1 + P_2) \text{with a probability of } \left(\tfrac{1}{4}\right)^2\tfrac{3}{4}$$

The expected number of trials required by a geometric random variable is given by $E[X] = \frac{1}{p}$. Therefore, the expected number of trials required is $\frac{4}{3}$. Hence, we can conclude that this strategy A' decides the language without an error in polynomial time, i.e a subset of ZPP.

4

Given: With probability $1 - O(n^{-1/4})$, Lazyselect finds $S(k)$ on the first pass through the steps of the algorithm, and thus performs only $2n + o(n)$ comparisons.
Required:
Show that, the expected running time of the Lazyselect algorithm is $2n + o(n)$.

$$\text{Algorithm terminates after } (2n + o(n)) \text{comparisons with a probability of } 1 - O(n^{-1/4})$$
$$\text{Algorithm terminates after } 2(2n + o(n)) \text{comparisons with a probability of } O(n^{-1/4})\left(1 - O(n^{-1/4})\right)$$
$$\text{Algorithm terminates after } 3(2n + o(n)) \text{comparisons with a probability of } O(n^{-1/4})^2\left(1 - O(n^{-1/4})\right)$$

Let X be a random variable that counts the number of passes (trials) of the algorithm required to reach the solution. Each trial has a probability of success of $p = 1 - O(n^{-1/4})$ and a failure of $O(n^{-1/4})$. Since each trial is independent of the previous trials, then X is a geometric random variable with the following PMF.

$$Pr[X = n] = O(n^{-1/4}) \left(1 - O(n^{-1/4})\right)$$

Therefore, the expected number of passes is given by:

$$E[X = n] = \frac{1}{p} = \frac{1}{1 - O(n^{-1/4})}$$

For large n the expected number of passes converges to 1, which means that we need only one pass to reach the answer. Since the expected running time is measured by the number of comparisons, then, we will need only $2n + o(n)$ comparisons. Therefore, the expected running time of the Lazyselect algorithm is $2n + o(n)$.

5

Given: n balls independently distributed in n bins.
Required:

- For large n, the expected number of empty bins approaches $\frac{n}{e}$.
- Expected number of empty bins when m balls are thrown into n bins.

Let X_i be a random variable to indicate that bin i is empty. Therefore, $X_i = 1$, if bin i is empty and $X_i = 0$, otherwise.

Probability of bin i being empty after n throws is given by:

$$\begin{aligned} P[X_i = n] &= \left(1 - \frac{1}{n}\right)^n \\ &\cong \frac{1}{e} \qquad \text{for large } n \end{aligned}$$

Since X_i is an indicator random variable, then

$$E[X_i] = Pr[X_i]$$

Let X be the total number of empty bins.

$$X = X_1 + X_2 + \ldots + X_n.$$

Therefore, the expected number of empty bins is:

$$\begin{aligned} E[X] &= E\left[\sum_{i=1}^{n} X_i\right] = \sum_{i=1}^{n} E[X_i], \qquad \text{from linearity of expectation} \\ &\cong n \times \frac{1}{e} = \frac{n}{e} \end{aligned}$$

For m balls thrown into n bins, we repeat the same argument. Let X_i be a random variable to indicate that bin i is empty. Therefore, $X_i = 1$, if bin i is empty and $X_i = 0$, otherwise. Probability of bin i being empty after m throws is given by:

$$\begin{aligned} P[X_i = n] &= \left(1 - \frac{1}{n}\right)^m \\ &= \left(1 - \frac{1}{n}\right)^{n \times \frac{m}{n}} \\ &\cong e^{-\frac{m}{n}} \qquad \text{for large } n \end{aligned}$$

Since X_i is an indicator random variable, then

$$E[X_i] = Pr[X_i]$$

Let X be the total number of empty bins.

$$X = X_1 + X_2 + \ldots + X_n.$$

Therefore, the expected number of empty bins is:

$$\begin{aligned} E[X] &= E\left[\sum_{i=1}^{n} X_i\right] = \sum_{i=1}^{n} E[X_i], && \text{from linearity of expectation} \\ &\cong ne^{-\frac{m}{n}} && \text{for large } n \end{aligned}$$