CS648 : Randomized Algorithms
Semester I, 2011-12, CSE, IIT Kanpur

Analysing the duration of a rand. algo. by partitioning it into stages

In this lecture we shall discuss four interesting problems involving some randomized algorithms. Each of these algorithms performs a sequence of steps or rounds and the objective is to calculate the expected duration (number of steps or rounds) of the algorithm. The solution of each of these problems is based on a common technique which can be explained as follows. Visualize the algorithm as passing through various *stages* such that once the algorithm crosses one stage, it does not return to that stage in future. The entire duration of the algorithm is nothing but the total sum of the number of steps spent by the algorithm in each of these stages. So it suffices to calculate the expected number of steps executed by the algorithm in each of these stages. Though simple it may appear, pursuing this approach requires ingenuity and is sometime very nontrivial. To realize this fact, the reader may try attempting the four problems discussed below before studying their solution.

As far as probability tools are concerned, we shall employ linearity of expectation, Markov inequality, Chernoff bound, and sometimes even elementary probability calculations. However, in order to focus on the technique of partitioning the randomized algorithm, we encourage the reader to go through the following subsection on recurrences as a warm up.

0.1 A warm-up on the recurrences

We consider the recurrences involving a fraction ϵ_i where ϵ_0 is defined as some constant c less than 1. We shall seek the answer to the following question for each of these recurrences.

Question 0.1 *What is the largest value of i such that $\epsilon_i > 1/n$ for a given positive integer n ?*

Let us consider or first recurrence.

Recurrence 1 $\epsilon_i = \frac{1}{2}\epsilon_{i-1}$ for all $i > 0$

By gradual unfolding the recurrence we observe that $\epsilon_i = (\frac{1}{2})^i c$. Hence solving $\epsilon_i > 1/n$ for largest possible value of i gives $i = \log_2 cn$.

Let us consider the second recurrence.

Recurrence 2 $\epsilon_i = \epsilon_{i-1}^2$ for all $i > 0$

By gradual unfolding, we observe that $\epsilon_i = (c)^{2^i}$. Hence solving $\epsilon_i > 1/n$ for the largest possible value of i gives $i = \log_2 \log_{1/c} n$.

Let us consider the third recurrence now.

Recurrence 3 $\epsilon_i = 2^{-\frac{1}{\epsilon_{i-1}}}$ for all $i > 0$

Reformulating the recurrence we get, $\frac{1}{\epsilon_i} = 2^{\frac{1}{\epsilon_{i-1}}}$ which can be unfolded to get

$$\frac{1}{\epsilon_i} = (\cdots((2)^2)^{\cdots i \text{ times} \cdots 2})^{\frac{1}{c}})$$

Hence the largest value of i such that $\epsilon_i > 1/n$ is bounded by $i = \log^* n$, where $\log^* n$ is the number of times we need to take logarithim on n to get 1. This is an extremely slow growing function and less than 6 for almost all practical purposes. However, theoretically it is not appropriate to consider it as a constant.

0.2 A warm-up on elementary probability problems

In a couple of the problems we shall solve, we shall essentially use the following coin tossing problems which we have discussed in the class many times :

- P_1 : Given a coin that gives head with probability p, what is the expected number of coin toss to get a head. It is easy to show that the answer is $1/p$.
- P_2 : Given a coin that gives head with probability p, what is the expected number of coin tosses to get k heads ? Using linearity of expectation and solution of P_1 , it follows that the expected number of tosses required in P_2 is k/p.

1 Coupon Collector Problem

Problem Statement :
There is a bag which contains n different types of coupons. Furthermore, the number of coupons of each type is infinite in the bag We want to have at least one coupon for each of these types. To achioeve this goal, we repeat the following sampling step : take out a coupon from the bag uniformly and randomly. In other words, during each sampling step, the coupon drawn is equally likely to be of any of the n types of coupons. What is the expected number of sampling steps required to collect one coupon of each type ?

Let X be the random variable for the number of steps needed to collect n types of coupons. It is easy to observe that calculating $E[X]$ from definition is at least not easy (if not impossible). So we should try to express it as a sum of random variables. But compared to the previous problems we solved (quick sort, red black balls, empty bins), where it was much easier to express the random variable as a sum of random variables, it is not so easy in this problem (try to convince yourself). So how should we proceed ? We shall partition the entire algorithm into stages. For this objective, the reader is advised to go through the execution of this algorithm for a couple of times.

The stage of the algorithm after jth sampling step can be defined by $\nu(j)$ which is the number of distinct types of coupons which have been collected at the end of jth sampling step. In the beggining $\nu(0) = 0$ and when the algorithm ends, say at kth sampling step, we have $\nu(k) = n$. The entire algorithm can be viewed as progressing from stage 0 to stage n though the number of steps spent in each stage is a random variable. One possible execution of the algorithm is shown below.

0 1 1 1 2 2 3 4 4 4 4 5 5 6 6 6 6 7 8 8 8 8 8 8 $\cdots$(n-1) (n-1) n

The algorithm takes 1 step to move to stage 1 from 0, takes 3 steps to move to stage 2 from 1, and so on. Let X_i denote the random variable for the number of steps taken by the algorithm in stage i to move to $i+1$. Observe that the stage of the algorithm increases monotonically and in discrete steps of one unit. Owing to this fact it follows that $X = \sum_{i=0}^{n-1} X_i$. So using linearity of expectation, it suffices to calculate $\mathbf{E}[X_i]$. Here, note the following crucial observation which follows from the uniformity and independence underlying the sampling steps.

Observation 1.1 *The value taken by X_i is not governed by the value taken by X_j,$j \neq i$. Secondly, the value taken by X_i is not governed by which specific i different types of coupons we have collected.*

The above observation helps us in focusing only on the following aspect of a sampling step when the algorithm is in stage i : *the type of the coupon selected compared to those which we had collected before.* It follows from uniformity underlying the sampling steps that if the algorithm is in stage i then a sampling step will produce a coupon of *new* type with probability $\frac{n-i}{n}$. As soon as this happens, the algorithm moves to stage $i+1$ otherwise the algorithm stays in the same stage and repeats the sampling step. Does it remind you of some well known randomized process we discussed earlier ? Well, it is Problem P_1; the random variable X_i has same probability distribution as the number of tosses to get a HEADS if the probability of getting HEADS is $p = \frac{n-i}{n}$. Therefore,

$$E[X_i] = \frac{n}{n-i} \tag{1}$$

Hence using linearity of expectation, and Equation 1 it follows that

$$E[X] = \sum_{i=0}^{n-1} E[X_i] = n[\frac{1}{n} + \frac{1}{n-1} + \cdots + \frac{1}{1}] = nH_n = \Theta(n \ln n)$$

2 Discrete Random walk on a line

The problem can be defined as follows. Consider the positive half of the real line extending to infinity and partitioned into discrete steps of one unit each. A particle starts performing a random walk starting from origin. In each second the particle moves one step to the left or right with equal probability. From origin, it always moves to 1 will probability 1. What is the expected number of steps required to reach a distance of n units from origin ?

Let X be a random variable for the number of steps taken by the particle to reach n starting from 0. We need to calculate $\mathbf{E}[X]$. Let us define random variable X_i, $0 \leq i < n$ as follows.

X_i is the number of steps taken by the particle to reach $i+1$ for the first time when the walk starts at location i.

Give suitable arguments to show that

$$X = \sum_{i=0}^{n-1} X_i$$

Hence applying linearity of expectation, it follows that we need to calculate $\mathbf{E}[X_i]$. Here provide suitable arguments based on conditional probability that $\mathbf{E}[X_0] = 1$ and for any $i > 0$,

$$\mathbf{E}[X_i] = 2 + \mathbf{E}[X_{i-1}]$$

Hence derive, by unfolding the above recurrence that, $\mathbf{E}[X_i] = 2i + 1$. Hence $\mathbf{E}[X] = \sum_{i=0}^{n-1} \mathbf{E}[X_i] = n^2$.

3 A Client Server Problem

Consider a distributed environment involving n clients and n servers. Each client has a job which requires one round of service from any of the servers. All servers are identical. As soon as job of a client is served by a server, the client *leaves*. Problem is that there is no centralized authority to allocate servers to clients. However, each client knows the address of each server and can communicate its job request to the server. The aim is to design a distributed algorithm so that all the clients get served as soon as possible. Here is a simple randomized algorithm which proves to be very efficient. In a given round the following activities take place.

1. Each client which has not been served yet sends a message to a randomly selected server.
2. Each server which receives just a single request, sends a reply to the corresponding client asking for the job, and completes that job in that round. (the corresponding client leaves after this).
3. All those clients which do not receive any reply to their request, try the same protocol in the next round.

The reader may consider the following ball-bin formulation of this problem : We start off by throwing n balls into n bins in the first round. After round $i \geq 1$, we remove every ball that occupied a bin by itself in round i (that, is it was the only ball in its bin). In the following round $(i+1)$, we throw the remaining balls into the bins. (one can imagine the lonely ball getting service, whereas none of the colliding balls receive service.) The process ends when no balls is left.

Let us first develop some familiarity with the experiment. So let us focus on the first round. Referring to some of the home work problems, we can note that the expected number of servers that receive request from exactly one client is $\approx \frac{n}{e}$. Therefore, the expected number of clients left unserved after the first round is nearly $(1 - \frac{1}{e})n$. In other words, the number of clients reduce by some constant fraction. With this observation, and recalling **Recurrence 1**, the first answer that comes to the mind is : the expected number of rounds would be $O(\log n)$. But we are missing the following crucial point about the algorithm : As the number of clients reduce in the successive rounds, the probability of multiple clients approaching a server also reduce in successive rounds. So as the experiment progresses, a client is less likely to be rejected when it approaches a server. Therefore, we expect fewer number of rounds than $(\log n)$. To explore this intuition, we should analyze arbitrary round and not the first round. Let X_i denote the number of clients left after ith round. Consider $(i+1)$th round. Let $X_i = m$, that is, the $(i+1)$th round begins with m clients. The expected number of server which receive request from exactly one of m clients

in this round would be $n\binom{m}{1}\frac{1}{n}\left(1-\frac{1}{n}\right)^{m-1} = m\left(1-\frac{1}{n}\right)^{m-1}$ Thus the expected value of X_{i+1} conditioned on $X_i = m$ can be expressed as

$$E[X_{i+1}|X_i = m] = m - m\left(1-\frac{1}{n}\right)^{m-1} \tag{2}$$

Consider the following expansion of $(1-\frac{1}{n})^{m-1}$

$$\left(1-\frac{1}{n}\right)^{m-1} = 1 - \frac{m-1}{n} + \frac{(m-1)(m-2)}{2!n^2} - \frac{(m-1)(m-2)(m-3)}{3!n^3} + \cdots$$

It can be seen that the expansion is a sum of terms, with alternating signs. Moreover, since $m < n$, the absolute value of the terms decrease as we go further right in this expansion. Hence we can conclude that

$$1 - \frac{m-1}{n} < \left(1-\frac{1}{n}\right)^{m-1} < 1 - \frac{m-1}{n} + \frac{(m-1)(m-2)}{2!n^2}$$

Hence we can conclude the following from Equation 2

$$E[X_{i+1}|X_i = m] = m - m\left(1-\frac{m-1}{n}\right) < \frac{m^2}{n}$$

Dividing both sides by n we get

$$\frac{1}{n}E[X_{i+1}|X_i = m] < \left(\frac{m}{n}\right)^2$$

Now note that $\frac{1}{n}E[X_{i+1}|X_i = m] = E[\frac{X_{i+1}}{n}|X_i = m]$ (try to give reason). So the above expression conveys the following crucial observation about the experiment.

Observation 3.1 *If after i rounds a fraction ϵ_i of clients are left, that is $X_i = \epsilon_i n$, then after $(i+1)$ rounds expected fraction of clients left would be less than ϵ_i^2.*

Once there is one client left, the algorithm will take only one more round. So what does the above observation convey ? (The reader is advised to recall **Recurrence 2** here). If we start with cn clients with $c < 1$, and each round goes as expected, i.e., the number of clients left is same as the expected number of clients which should be left then how many rounds would be required ? The answer is $O(\log\log n)$. But nothing goes as expected in life unless we try hard. So how do we use the above insight to formally show that the expected number of rounds taken by the algorithm will indeed be $O(\log\log n)$? (Ponder over this very very important question for afew minutes before proceeding further). We shall use the technique of partitioning the algorithm into stages to achieve this goal.

The execution of the algorithm begins with n clients and ends with no client. We can partition the experiment into three stages for a constant $c < 1$:

1. **Stage I:** The number of clients left unserved is more than cn.
2. **Stage II:** The number of clients unserved is more than 1 but and less than or equal to cn.
3. **Stage III:** There is one or no client left unserved.

It follows that the algorithm starts in stage I and ends in stage III. Stages of any execution can be visualized as follows:

$$\text{I I} \cdots \text{II II} \cdots \text{III}$$

We shall show that the expected number of stage I rounds will be constant. Whereas, the expected number of rounds for stage II would be $O(\log\log n)$. Can you come up with the proof for these claims ? You just need to look carefully. We have done similar stuff in the past as well. After a pause for a few minutes, proceed to the proof given below.

Expected number of rounds of Stage I is $O(1)$: The experiment begins with stage I and enters stage II or stage III (if no balls are left) when there are fewer than cn clients. Consider $(i+1)$th round

of the first stage. Let it begins with m_i clients, $m_i > cn$. Let X_{i+1} be the number of clients left after ith round. It follows from Equation 2 that the $E[X_{i+1}|X_i = m_i] = m_i - m_i\left(1 - \frac{1}{n}\right)^{m_i - 1}$ which is bounded by $m_i(1 - 1/e)$ for all $m_i < n$. Call a round successful if the number of clients left is no more than 3/4th of the previous round. Applying Markov inequality

$$Pr[X_{i+1} > \frac{3m_i}{4}|X_i = m_i] \leq \frac{1 - 1/e}{3/4} < \frac{8}{9}$$

Hence Probability that $(i + 1)$th round is successful is at least 1/9. It can be seen that after i successful rounds the fraction of clients will be reduced to $(3/4)^i$. Since the first stage terminates as soon as the fraction of clients fall below c, the number of successful rounds in the first stage of the experiment is no more than $\log_{3/4} c$. It is also clear that each round is successful or unsuccessful independent of the past rounds. So recalling Problem P_2, we can conclude that the expected number of rounds spent by the algorithm in stage I would be no greater than $9 \log_{3/4} c = O(1)$.

Expected number of rounds of Stage II is $O(\log \log n)$: Let Y_i denote the fraction of the clients left after ith round of the second stage. It can be seen that Y_i is a random variable. It follows from Observation 3.1 that

$$E[Y_{i+1}|Y_i = \epsilon] < \epsilon^2$$

Using Markov inequality, therefore, $Pr[Y_{i+1} \geq \epsilon^{3/2}|Y_i = \epsilon] \leq \frac{E[Y_{i+1}|Y_i = \epsilon]}{\epsilon^{3/2}} = \sqrt{\epsilon}$

Recall that the second stage of the experiment begins with fewer than cn balls. So $\epsilon < c$ holds always,

$$Pr[Y_{i+1} \geq \epsilon^{3/2}|Y_i = \epsilon] \leq \sqrt{c} \tag{3}$$

Therefore, $Pr[Y_{i+1} \geq Y_i^{3/2}] \leq \sqrt{c}$ (which probability lemmas do we use here?). Now the rest of the analysis is similar to that of expected number of rounds for Stage I. We define a round to be successful or unsuccessful based on the following condition. If the fraction of clients ϵ' left after the round is related to the the fraction ϵ of balls before the round by $\epsilon' < \epsilon^{3/2}$, then the round is successful otherwise it is unsuccessful. Inequality 3 states that probability of round to be successful is at least $1 - \sqrt{c}$. Now let us ask ourselves the following Question : *How many successful rounds will be there in this experiment ?* It follows from **Recurrence 2** that only $\log_{3/2} \log_{1/c} n$ successful rounds are required to transform the algorithm from stage II to stage III. So the problem is to calculate the expected number of rounds needed to get $\log_{3/2} \log_{1/c} n$ successful rounds given that each round is successful with probability $1 - \sqrt{c}$. This is similar to the problem P_2. Hence the expected number of rounds for stage II is bounded by $\frac{1}{1-\sqrt{c}} \log_{3/2} \log_{1/c} n = O(\log \log n)$.

This completes the analysis of the randomized distributed algorithm.

Homework: Try to show that the number of rounds of the experiment is concentrated around $O(\log \log n)$. In other words, show the following. For any constant $d > 1$, there exists a constant b such that the probability that the experiment has more than $b \log \log n$ rounds is less than $\frac{1}{(\log n)^d}$. (Hint : apply Chernoff bound carefully).