CSOR W4231: Analysis of Algorithms (sec.001) - Problem Set #1

Jiawen Huang (jh4179). Collaborators: A. Turing. - jh4179@columbia.edu

February 23, 2020

Problem 1

We assume probability of X_1 = [R contains at least 1 element of T] as P_1

$$P_1 = 1 - (\frac{n-1}{n})^n$$

The probability of X_2 = [R contains no element of S] as P_2 is

$$P_2 = (\frac{n-1}{n})^n$$

Since X_1 and X_2 is independent events, the probability that the sample is good is

$$P = P_1 * P_2 = (1 - (\frac{n-1}{n})^n) * (\frac{n-1}{n})^n$$

where

$$\frac{\partial (\frac{n-1}{n})^n}{\partial n} = \frac{(1-\frac{1}{n})^{n-1}}{n} > 0 \quad (when \quad n > 1)$$

from which we know that the $(\frac{n-1}{n})^n$ has min value when n=2, which is $\frac{1}{4}$
According to the hint

$$(1 - (1 - \frac{1}{n})^n) \geq 1 - e^{-1}$$

so

$$P = (1 - (\frac{n-1}{n})^n) * (\frac{n-1}{n})^n \geq (1 - e^{-1}) * \frac{1}{4}$$

Then we have the statement that the probability our sample is good is larger than some positive constant, such as any number less then $(1 - e^{-1}) * \frac{1}{4}$, which is about 0.158.

Problem 2

(a)

Pseudocode:

replace(a)

$item = a[1]$
for $k = 2$ to Length(a) **do**

$$item = \begin{cases} a[i] & \text{with probability } \frac{1}{k} \\ item & \text{with probability } 1 - \frac{1}{k} \end{cases}$$

end for

Correctness:
Base case: k = 2, the probabilities of storing a[1] and a[2] are both 1/2 then the statement is true.
Induction hypothesis: assume that the statement is true for $k \geq 2$.
Inductive step: show it true for case k+1
Since each of $a[1], a[2], ..., a[k]$ will be stored with a probability of 1/k, when $a[k+1]$ appears, there is a probability of $1 - \frac{1}{k+1}$ that the stored item will fall in the previous range $(a[1], a[2], ..., a[k])$, which means each of $a[1], a[2], ..., a[k]$ will be stored with a probability of

$$\frac{1}{k} * (1 - \frac{1}{k+1}) = \frac{1}{k}$$

There is also another probability of $\frac{1}{k+1}$ that the stored item will be replace with $a[k+1]$. To conclude, the item has the property that it is uniformly distributed over all the items we have already seen, so the statement is also true in case k+1.
Conclusion: it follows that the statement is true for $k \geq 2$.
Running Time:
The algorithm has only a for loop, so the running complexity is O(Length(a)), although we do not know the length in the process of the algorithm.
Space:
The space complexity is O(1).

(b)

It follows a distribution that when k-th items have been seen

$$p(a[i]) = \begin{cases} \frac{1}{2^{k-i+1}} & i \neq 1 \\ \frac{1}{2^{k-1}} & i = 1 \end{cases}$$

Correctness:

Base case: k = 2, the probabilities of storing a[1] and a[2] are both 1/2 then the statement is true.
Induction hypothesis: assume that the statement is true for $k \geq 2$.
Inductive step: show it true for case k+1
When k-th item appears, item $a[i](i \neq 0)$ will be stored with a probability of

$$\frac{1}{2^{k-i+1}}$$

and the probability for item $a[1]$ is $\frac{1}{2^{k-1}}$.
When $a[k+1]$ appears, there is a probability of $\frac{1}{2}$ that the stored item will fall in the previous range $(a[1], a[2], ..., a[k])$, which means item $a[i](i \neq 0)$ will be stored with a probability of

$$\frac{1}{2} * \frac{1}{2^{k-i+1}} = \frac{1}{2^{k-i+2}}$$

and the probability for item $a[1]$ is $\frac{1}{2} * \frac{1}{2^{k-1}} = \frac{1}{2^k}$.
There is also another probability of $\frac{1}{2}$ that the stored item will be replace with $a[k+1]$.
To conclude, the item has the property that it follows a distribution that when k-th items have been seen

$$p(a[i]) = \begin{cases} \frac{1}{2^{k-i+1}} & i \neq 1 \\ \frac{1}{2^{k-1}} & i = 1 \end{cases}$$

so the statement is also true in case k+1.
Conclusion: it follows that the statement is true for $k \geq 2$.
Running Time:
The algorithm is very similar to the one in (a), so the running complexity is O(Length(a)).
Space:
The space complexity is O(1).
Pseudocode:

replace(a)

$item = a[1]$
for $k = 2$ to Length(a) **do**

$$item = \begin{cases} a[i] & \text{with probability } \frac{1}{2} \\ item & \text{with probability } \frac{1}{2} \end{cases}$$

end for

Problem 3

(a)

If n is old, we set $k = \lfloor n/2 \rfloor + 1$ and use k-th order statistic (S,k) to find the median. If n is even, we set $k = n/2$ and $k = n/2 + 1$ seperately and use k-th order statistic (S,k) to find the two items, then calculate the mean of the two items, which is the median of S.

(b)

Upper bound the expected time of a subproblem of type j:

The expected running time of a subproblem of type j excluding the time on recurives calls is $O(Length(S))$, here $n * (\frac{3}{4})^j \leq S \leq n * (\frac{3}{4})^{j+1}$.

Upper bound the expected time until a good item a_i is selected:

If we want to select an item a_i that can help us throw out at least $\frac{1}{4}$ of the input, we can judge it by looking at the length of $|S^-|$ to assure the size of the part we can throw up, which takes $O(Length(S))$ time. Also we have a probability of $\frac{1}{2}$ to pick an item a_i that we want, so we select a good a_i by spending $O(Length(S))$ time in total.

Upper bound the expected time of the entire algorithm:

For the entire algorithm, the depth of j is $\log_{\frac{3}{4}} \frac{1}{n}$, so we have

$$O(\frac{3}{4}n) + O(\frac{3}{4}n) + O((\frac{3}{4})^2 n) + O((\frac{3}{4})^2 n) + ... + O((\frac{3}{4})^{\log_{\frac{3}{4}} \frac{1}{n}} n) + O((\frac{3}{4})^{\log_{\frac{3}{4}} \frac{1}{n}} n) = O(n)$$

Also we can derive the answer according to master theorem,

$$T(n) = T(\frac{3}{4}n) + O(n)$$

The expected running time is $O(n)$.

Problem 4

(a)

Suppose we have k balls, for each ball the probability that it will be discarded is

$$p = (1 - \frac{1}{n})^{k-1}$$

so the expected number of balls that should be discarded in this round is

$$E(D_k) = k * p = k * (1 - \frac{1}{n})^{k-1}$$

Now we have $k = \epsilon n$, so the expected balls that we have for next round is

$$\begin{aligned} E = k - E(D_k) &= \epsilon n(1 - (1 - \frac{1}{n})^{\epsilon n - 1}) \\ &\leq \epsilon n(1 - e^{-\frac{1}{n} * (\epsilon n - 1)}) \\ &< \epsilon n(1 - e^{-\epsilon}) \\ &\leq \epsilon n(1 - (1 - \epsilon)) \\ &= \epsilon^2 n = k^2/n \end{aligned}$$

We expect to have at most $\epsilon^2 n$ balls.

(b)

From part (a), we know that if we have k balls in some round, the expected number of balls left will be k^2/n, so suppose we have a_i balls after i-th round we can induct the equation

$$a_k = a_{k-1}^2/n = a_{k-2}^4/n^3 = ...$$

Also from part (a), we have $a_k < \epsilon n(1 - e^{-\epsilon})$. Inducted by this equation, we can get that after the first round we have at most $n(1 - e^{-1})$ balls and after the second round we have at most $n(1 - e^{-1})^2 < 1/2$ balls, which means $a_2 < 1/2$. Back to the previous equation, we can now finish it

$$\begin{aligned} a_k &= a_2^{2^{k-2}}/n^{1+2+...+2^{k-3}} \\ &< (n/2)^{2^{k-2}}/n^{2^{k-2}-1} \\ &= \frac{1}{2^{2^{k-2}} n} \end{aligned}$$

If we want the experiment ends at round k, we need $a_k < 1$, which means

$$\frac{1}{2^{2^{k-2}} n} < 1$$

$$k > 2 + \log \log n$$

So as long as $k > 2 + \log \log n$, the experiment must end no later than k-th round. Equivalently, we would discard all the balls within $O(\log \log n)$ rounds.

Problem 5

(a)

If the other $n-1$ people are not trying to access the computer and person i is trying to access, then the person i could succeed. The probability is $p*(1-p)^{n-1}$, which we will use P_1 to denote.

(b)

We can calculate the first order derivative and makes it equal to 0.

$$\frac{\partial P_1}{\partial p} = (1-p)^{n-1} - p*(n-1)*(1-p)^{n-2} = 0$$

$$p = \frac{1}{n}$$

so we need to set $p = \frac{1}{n}$ and probability P_1 will be $\frac{1}{n} * (1-\frac{1}{n})^{n-1}$.

(c)

The probability that person i fails in one step is $1 - \frac{1}{n} * (1-\frac{1}{n})^{n-1}$. If the person i did not succeed to access in any of the first $t = en$ steps, the probability will be $(1-\frac{1}{n}*(1-\frac{1}{n})^{n-1})^{en}$, which we will use P_2 to denote.

$$P_2 = (1-\frac{1}{n}*(1-\frac{1}{n})^{n-1})^{en}$$

$$\leq (1-\frac{1}{n}*\frac{n}{n-1}*e^{-1}(1-\frac{1}{n}))^{en}$$

$$= (1-\frac{1}{en})^{en}$$

$$\leq e^{-1}$$

so the upper bound is e^{-1}.

(d)

From part (c), we have $(1-\frac{1}{en})^t$ is the approximation of P_2, and if $t = en$, P_2 should be upper bounded by e^{-1}.
Now we make $t = ken\log n$, then we have

$$P_2 \leq e^{-k\ln n} \leq \frac{1}{n^k}$$

so t should be more than $ken\ln n$.

(e)

We know $1 - P_2$ is the probability that one person will succeed to access the computer in the first t steps, so if all people want to succeed, the the probability is $(1 - P_2)^n$. We need the probability greater than $1 - \frac{1}{n}$ then we have

$$(1 - P_2)^n \geq 1 - \frac{1}{n}$$

We make $t = -en\ln(1 - e^{-\frac{1}{n^2}})$, so we have

$$P_2 \leq e^{ln(1-e^{-\frac{1}{n^2}})} = 1 - e^{-\frac{1}{n^2}}$$

$$1 - p_2 \geq e^{-\frac{1}{n^2}}$$

since $e^{-1} \geq (1 - \frac{1}{n})^n$, $e^{-\frac{1}{n}} \geq 1 - \frac{1}{n}$

$$(1 - P_2)^n \geq e^{-\frac{1}{n}} \geq 1 - \frac{1}{n}$$

so t should be more than $-en\ln(1 - e^{-\frac{1}{n^2}})$, which is roughly $\ln\frac{3}{2}en \approx 1.102n$ when n is large enough.