CSE523: Randomized Algorithms

Assignment 1 Solutions

Divyajeet Singh (2021529)

Solution 1.

Let N be the number of people who receive the card meant for them. Let X_i be a Bernoulli random variable governing that the i^{th} person receives the card meant for them, i.e.

$$X_i = \begin{cases} 1 & \text{the } i^{th} \text{ person receives the } i^{th} \text{ card} \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

Note that X_i is 1 irrespective of the positions of other cards. We are interested in finding the expectation of N, i.e. $\mathbb{E}[N]$. We see that the random variable N can be written as

$$N = X_1 + X_2 + \cdots + X_n = \sum_{i=1}^{n} X_i \tag{2}$$

$$\implies \mathbb{E}[N] = \sum_{i=1}^{n} \mathbb{E}[X_i] = \sum_{i=1}^{n} \mathbb{P}[X_i = 1] \qquad \text{by linearity of expectation} \tag{3}$$

We find the probability that the i^{th} person recieves the correct card. Out of the $n!$ total permutations, if we fix one card at its correct envelope, the remaining cards can be arranged in exactly $(n-1)!$ ways (keeping the correctly placed i^{th} card fixed).

$$\mathbb{P}[X_i = 1] = \frac{(n-1)!}{n!} = \frac{(n-1)!}{n \cdot (n-1)!} = \frac{1}{n} \tag{4}$$

So, each $X_i \sim \textsc{Bernoulli}\left(\frac{1}{n}\right)$. Hence, by (3) and (4), we get

$$\mathbb{E}[N] = \sum_{i=1}^{n} \mathbb{P}[X_i = 1] = \sum_{i=1}^{n} \frac{1}{n} = \frac{1}{n} \cdot n = 1 \tag{5}$$

Hence, in expectation, 1 person receives the card meant for them. Section 1 in the Appendix expands on the intuition behind the solution.

Solution 2.

We are now interested in the probability[1] $\mathbb{P}[N = r]$. For this, just like above, we fix any r cards. We want only and exactly these cards to go in their correct envelopes. Once these r cards are fixed, out of the $n!$ permutations, the remaining cards can be arranged in exactly $(n-r)!$ ways (keeping the correctly placed r cards fixed). Let us denote the indices of fixed cards by $p_1, p_2, \ldots, p_r$. Then[2], the total probability that these cards are in their correct envelopes is

$$\mathbb{P}\left[\bigcap_{i=1}^{r} X_{p_i}\right] = \frac{(n-r)!}{n!} \tag{6}$$

However, we want to now rule out the permutations where any *other* cards appear on their correct position. Let us denote the indices of the non-fixed cards by $q_1, q_2, \ldots, q_m$ (where $m = n - r$). We first find the probability that

[1] Notice how $N \nsim \textsc{Binomial}(n, \mathbb{P}[X_i = 1])$ since it is a sum of n dependent Bernoulli random variables.

[2] **Abuse of notation:** I use the random variables X_i and the event that the i^{th} card is placed in the correct envelope interchangeably. So, the union of X_1 to X_n represents the the event $(X_1 = 1) \cup (X_2 = 1) \cup \cdots \cup (X_n = 1)$. The same goes for intersections.

(among the non-fixed cards) at least 1 card is in its correct envelope. This can be found using the Inclusion-Exclusion principle.

$$\mathbb{P}\left[\bigcup_{i=1}^{m} X_{q_i}\right] = \sum_{i=1}^{m} \mathbb{P}[X_{q_i}] - \sum_{1\le i<j\le m} \mathbb{P}[X_{q_i} \cap X_{q_j}] + \sum_{1\le i<j<k\le m} \mathbb{P}[X_{q_i} \cap X_{q_j} \cap X_{q_k}] - \cdots + (-1)^m \mathbb{P}\left[\bigcap_{i=1}^{m} X_{q_i}\right] \quad (7)$$

The terms in this large sum can be easily found by substituting different values for r in (6) and multiplying by the size of the summation. It is interesting that the probability in (6) is irrespective of the choice of the r cards. So, the j^{th} term of the summation is simply the product of the number of ways to choose j out of m cards and the corresponding probability that the j chosen cards are in their correct envelopes.
Out of $m = n - r$, j cards can be selected in exactly $\binom{m}{j}$ ways. Hence, we get

$$\begin{aligned}
\mathbb{P}\left[\bigcup_{i=1}^{m} X_{q_i}\right] &= \binom{m}{1}\frac{(m-1)!}{m!} - \binom{m}{2}\frac{(m-2)!}{m!} + \cdots + (-1)^m \binom{m}{m}\frac{(m-m)!}{m!} \\
&= \frac{m!}{1!\cdot(m-1)!}\cdot\frac{(m-1)!}{m!} - \frac{m!}{2!\cdot(m-2)!}\cdot\frac{(m-2)!}{m!} + \cdots + (-1)^m \frac{m!}{m!\cdot(m-m)!}\cdot\frac{(m-m)!}{m!} \quad (8) \\
&= \sum_{i=1}^{m} \frac{(-1)^{i+1}}{i!}
\end{aligned}$$

We are interested to find the probability that none of the non-fixed cards land on their correct positions. This can be found using (8)

$$1 - \mathbb{P}\left[\bigcup_{i=1}^{m} X_{q_i}\right] = 1 - \sum_{i=1}^{m} \frac{(-1)^{i+1}}{i!} = \frac{(-1)^0}{0!} + \sum_{i=1}^{m} \frac{(-1)^i}{i!} = \sum_{i=0}^{n-r} \frac{(-1)^i}{i!} \quad (9)$$

Finally, we find the probability that exactly r cards are in their correct envelopes

$$\begin{aligned}
\mathbb{P}[N = r] &= \binom{n}{r}\ \mathbb{P}\left[\bigcap_{i=1}^{r} X_{p_i}\ \cap\ \overline{\bigcup_{i=1}^{n-r} X_{q_i}}\right] \\
&= \binom{n}{r}\ \mathbb{P}\left[\bigcap_{i=1}^{r} X_{p_i}\right]\left(1 - \mathbb{P}\left[\bigcup_{i=1}^{n-r} X_{q_i}\right]\right) \quad (10) \\
&= \frac{n!}{r!\cdot(n-r)!}\cdot\frac{(n-r)!}{n!}\ \sum_{i=0}^{n-r}\frac{(-1)^i}{i!} = \frac{1}{r!}\sum_{i=0}^{n-r}\frac{(-1)^i}{i!} = \mathbf{N_r} \quad \text{(say)}
\end{aligned}$$

Note that the above probabilities can now be multiplied because X_{p_i}s are independent of X_{q_i}s, since the cards indexed with q_is can only be shuffled amongst themselves. So, we can write the probability distribution of N as follows

$$P_N(r) = \mathbb{P}[N = r] = \begin{cases} \mathbf{N_r} & \text{if } r = 0, 1, \ldots, n \\ 0 & \text{otherwise} \end{cases} \quad (11)$$

We can verify that (10) gives the correct answer by calculating the expectation of N, which is given in Section 2 of the Appendix to this submission. We now find the probability that none of the cards are in the correct envelope in the limit as $n \to \infty$. This is simply

$$\lim_{n\to\infty} P_N(0) = \lim_{n\to\infty} \frac{1}{0!}\sum_{i=0}^{n}\frac{(-1)^i}{i!} = \sum_{i=0}^{\infty}\frac{(-1)^i}{i!} = e^{-1} = \frac{1}{e} \quad \text{using Taylor's expansion} \quad (12)$$

Solution 3.

We present a modified version of the popular *Quickselect* algorithm. The Quickselect algorithm is a modification of Quicksort - we recurse only to one side of the array where we are sure to find the k^{th} smallest element[3]. We present a randomized version of the quickselect algorithm in Algorithm (1), where the partitioning pivot is chosen

Algorithm 1 Randomized Algorithm to find the k^{th} smallest element

```
procedure RAND-QUICKSELECT(A[1 : n], k):
    i ∼ 1, . . . , n uniformly at random
    j ← PARTITION(A[1 : n], i)
    if k < j then
        return RAND-QUICKSELECT(A[1 : j − 1], k)
    else if k > j then
        return RAND-QUICKSELECT(A[j + 1 : n], k − j)
    else
        return A[j]
    end if
end procedure
```

uniformly at randomly from the array at each step.
Leveraging randomization, this algorithm runs in linear time in expectation. The partitioning algorithm, given in Algorithm (2), is borrowed from Quicksort. It also returns the index at which the partition ends.

Algorithm 2 Partitioning algorithm around a given pivot index

```
procedure PARTITION(A[1 : n], i):
    x ← A[i]
    j ← 1
    for l ← 1 to n do
        if A[l] ≤ x then
            SWAP(A[j], A[l])
            j ← j + 1
        end if
    end for
    SWAP(A[j], A[n])
    return j
end procedure
```

When we pick a pivot in an array of size n, we count the number of elements which land to the right of the pivot. Let R denote the number of such elements. Since we pick the pivot randomly, it is equally likely that any number of elements land to the right of the pivot after partitioning, each with a probability of $\frac{1}{n}$. So, for any $0 \leq i \leq n$,

$$\mathbb{P}[R = i] = \frac{1}{n} \tag{13}$$

$$\mathbb{E}[R] = \sum_{i=0}^{n} i \, \mathbb{P}[R = i] = \sum_{i=0}^{n} \frac{i}{n} = \frac{1}{n} \cdot \frac{n \cdot (n-1)}{2} = \frac{n-1}{2} \approx \frac{n}{2} \tag{14}$$

We see that in expectation, we will get (approximately) half the elements to the right of the pivot. Similarly, approximately half the elements land to the left of the pivot in expectation. So, the recurrence can be defined as

$$\begin{aligned}
\mathbb{E}[T(n)] &= \mathbb{E}_{k \sim 1,\ldots,n}[T(k)] + cn = \mathbb{E}\left[T\left(\frac{n}{2}\right)\right] + cn \\
&= \mathbb{E}_{k \sim 1,\ldots,\frac{n}{2}}[T(k)] + c \cdot \frac{n}{2} + cn = \mathbb{E}\left[T\left(\frac{n}{4}\right)\right] + c \cdot \frac{n}{2} + cn \\
&\vdots \\
&\leq c \sum_{i=0}^{\infty} \frac{n}{2^i} = cn \cdot \left(1 + \frac{1}{2} + \frac{1}{4} + \cdots\right) = 2cn = O(n)
\end{aligned} \tag{15}$$

The worst case time complexity of this algorithm is still $O(n^2)$ since with at least $\frac{1}{n!}$ probability, the algorithm selects the worst pivots and repeatedly partitions around them (in the reversed sorted order).

[3]The worst case time complexity of quickselect is $\Theta(n^2)$. In the classical setting, we assume a distribution over all possible permutations of the array and say that *on average* (over the input distribution), quickselect runs in linear time.

Solution 4.

[**Note:** Hashing has not been covered in class - this is just an attempt.]
For simplicity, we assume that the arrays in consideration are permutations of $[n]$. Let us assume that the given *sorting* algorithm is called $\textsc{Sort}(A[1:n])$.
We build our solution stepwise. Algorithm (3) gives an algorithm to use the prime number generator to create hash functions. Let $\mathbb{W}_{<p}$ denote the set of whole numbers smaller than p, i.e. $\{0, 1, \ldots, p-1\}$.

Algorithm 3 Returns a random hash function

procedure GENERATE-HASH-FUNCTION(n, ϵ):
  $p \leftarrow$ prime greater than $\frac{n}{\epsilon}$
  $A \leftarrow \{x = (x_1, x_2, \ldots, x_r) \mid x \text{ is base-}p\ (x_i \in \mathbb{W}_{<p}\ \forall\ 1 \leq i \leq r)\}$
  $a \sim A$ uniformly at random
  return h_a defined by $h_a(x) \triangleq (\sum_{i=1}^{r} a_i x_i) \bmod p$ ▷ $h_a : A \to \mathbb{W}_{<p}$
end procedure

Courtesy for the hash function generation algorithm to [2]. I attempted to use this to first create a *simple* randomized algorithm $\mathcal{A}$ that checks the existence of only 1 element of A in $\textsc{Sort}(A)$. This could have been extended to an algorithm $\mathcal{A}_\epsilon$ that runs $\mathcal{A}$ some sublinear of n number of times to get the desired probability of success. However, all attempts were in vain.
A trivial $O(n)$ algorithm can be as follows. Get a hash map H of size p, where p is a prime greater than $\frac{n}{\epsilon}$. Let $B \leftarrow \textsc{Sort}(A)$. Mark every element of A as TRUE in H. If we find some $b \in B$ with $H[b] = \textsc{False}$, we report that the output is not a permutation. However, its analysis might be complicated due to the involved hashing.

Solution 5.

The given algorithm can be described by Algorithm (4).

Algorithm 4 Modified Quicksort Algorithm

procedure MOD-QUICKSORT($A[l:r]$):
  if $l \geq r$ **then**
    return
  end if
  while TRUE **do**
    $i \sim l, \ldots, r$ uniformly at random
    $j \leftarrow \textsc{Partition}(A[l:r], i)$
    if $j - l + 1 \leq \frac{2}{3}(r - l + 1)$ **and** $r - j + 1 \leq \frac{2}{3}(r - l + 1)$ **then**
      break
    end if
  end while
  MOD-QUICKSORT($A[l : j-1]$)
  MOD-QUICKSORT($A[j+1 : r]$)
end procedure

To analyze the expected running time of this algorithm, we first bound the probability of successfully finding a *good pivot*[4]. Let's find the number of good pivots in an array of size n. It is easy to see that if the sorted index j of the pivot is less than $\frac{1}{3}n$, then more than $\frac{2}{3}n$ elements land to its right. Similarly, if $j > \frac{2}{3}n$, then more than $\frac{2}{3}n$ elements land to its left. So, we see that the number of good pivots in an array of size n is given by

$$\left(\frac{2}{3} - \frac{1}{3}\right) n = \frac{n}{3} \tag{16}$$

Since the pivot is picked uniformly at random, the probability of picking a good pivot in one trial is

$$p = \frac{n}{3n} = \frac{1}{3} \tag{17}$$

[4]**Definition.** ***Good Pivot*:** A pivot having at most $\frac{2}{3}$ elements to either side.

It is not hard to notice that the number of trials required for successfully picking a pivot is distributed as a random variable $X \sim \textsc{Geometric}(p)$ since we repeat the trials (which are independent) until a good pivot is found. We find the expected number of trials required to get a successful trial, i.e. find a good pivot.

$$\mathbb{E}[X] = p^{-1} = \left(\frac{1}{3}\right)^{-1} = 3 \tag{18}$$

Note that each time a pivot is picked and tested for *goodness*, it takes $O(n)$ time for an array of size n since the partition is computed for that pivot. Finally, we calculate the number of elements that can land to the right (or left) of a good pivot. It follows from **Problem 3.** that all good pivots are equally likely (any pivot can be selected with equal probability). So, for any $\frac{1}{3}n \leq i \leq \frac{2}{3}n$,

$$\mathbb{P}[R = i] = \frac{1}{\frac{2}{3}n - \frac{1}{3}n} = \frac{3}{n} \tag{19}$$

$$\mathbb{E}[R] = \sum_{i=\frac{1}{3}n}^{\frac{2}{3}n} i\ \mathbb{P}[R = i] = \sum_{i=\frac{1}{3}n}^{\frac{2}{3}n} \frac{3i}{n} = \frac{3}{n} \sum_{i=\frac{1}{3}n}^{\frac{2}{3}n} i = \frac{3}{n}\left[\frac{n}{2 \cdot 3}\left(\frac{2n}{3} + \frac{n}{3} - 1\right)\right] = \frac{n-1}{2} \approx \frac{n}{2} \tag{20}$$

Similarly, approximately $\frac{n}{2}$ elements land to the left of a good pivot. Let $T(n)$ denote the time taken by Algorithm (4) on an input of size n. The recurrence relation can be defined as follows (the second term represents the expected work done to find the pivot)

$$\begin{aligned}
\mathbb{E}[T(n)] &= 2\mathbb{E}_{k \sim \frac{1}{3}n, \ldots, \frac{2}{3}n}[T(k)] + 3cn = 2\mathbb{E}\left[T\left(\frac{n}{2}\right)\right] + 3cn \\
&= 2\left(2\mathbb{E}_{k \sim \frac{1}{3}\cdot\frac{n}{2}, \ldots, \frac{2}{3}\cdot\frac{n}{2}}[T(k)] + 3c \cdot \frac{n}{2}\right) + 3cn = 2\left(2\mathbb{E}\left[T\left(\frac{n}{4}\right)\right] + 3c \cdot \frac{n}{2}\right) + 3cn \\
&= 2\left\{2\left(2\mathbb{E}\left[T\left(\frac{n}{8}\right)\right] + 3c \cdot \frac{n}{4}\right) + 3c \cdot \frac{n}{2}\right\} + 3cn \\
&= 2^3\ \mathbb{E}\left[T\left(\frac{n}{2^3}\right)\right] + 2^2 \cdot 3c \cdot \frac{n}{2^2} + 2^1 \cdot 3c \cdot \frac{n}{2^1} + 2^0 \cdot 3c \cdot \frac{n}{2^0} \\
&\ \ \vdots \\
&\leq 2^k\ \mathbb{E}\left[T\left(\frac{n}{2^k}\right)\right] + (k+1) \cdot 3cn
\end{aligned} \tag{21}$$

The recurrence ends when

$$\frac{n}{2^k} = 1 \implies 2^k = n \implies k = \log_2 n \tag{22}$$

Substituting (22) in (21) gives the expected time complexity of the modified quicksort algorithm where we only partition around good pivots.

$$\begin{aligned}
\mathbb{E}[T(n)] &\leq 2^{\log_2 n}\mathbb{E}[T(1)] + 3cn \cdot (\log_2 n + 1) \\
&= 3cn \log_2 n + (3c+1) \cdot n \qquad \text{assuming } T(1) = 1 \\
&\leq c^2 \cdot n \log_2 n \quad \forall\ n \geq 2 \\
\therefore \mathbb{E}[T(n)] &= O(n \log_2 n)
\end{aligned} \tag{23}$$

Hence, the modified quicksort algorithm still runs in $O(n \log_2 n)$ poly-log time in expectation. It is interesting that the overall expected running time of this algorithm matches the expected running time of random quicksort where the pivots are chosen uniformly at random.

References

1. Counting Derangements, Wikipedia
2. Randomized Algorithms by Klienberg-Tardos, Princeton University

Appendix

Section 1

One might *think* that linearity of expectation in (3) does not hold since X_is are dependent - if a card is already placed in an envelope, no other card can go in that envelope, and no other envelope can contain this card. This hunch of dependence is true; however, the linearity holds regardless of independence.
One might also be worried about *over-counting* or *double-counting* the cards, since a variable X_i can be 1 in cases when some j^{th} card is also correctly placed. However, even in that case, X_i still equals 1 (and so, we count it as 1 card being on its correct position). The j^{th} card being in the correct place will be accounted for by X_j, which will also be set to 1 in this case. Hence, we surely do not double count.
We can try to get the intuition for this by taking an example, $n = 3$. Then, we get the following permutations of the cards in envelopes numbered 1 to 3.

$$\begin{array}{lll} 1 & 2 & 3 \\ 1 & 3 & 2 \end{array} \qquad \begin{array}{lll} 2 & 1 & 3 \\ 2 & 3 & 1 \end{array} \qquad \begin{array}{lll} 3 & 1 & 2 \\ 3 & 2 & 1 \end{array}$$

The joint distribution table for X_1, X_2, and X_3 is given below.

$$\begin{aligned} \mathbb{P}[X_1 = 0, X_2 = 0, X_3 = 0] &= \frac{2}{6} & \mathbb{P}[X_1 = 1, X_2 = 0, X_3 = 0] &= \frac{1}{6} \\ \mathbb{P}[X_1 = 0, X_2 = 0, X_3 = 1] &= \frac{1}{6} & \mathbb{P}[X_1 = 1, X_2 = 0, X_3 = 1] &= 0 \\ \mathbb{P}[X_1 = 0, X_2 = 1, X_3 = 0] &= \frac{1}{6} & \mathbb{P}[X_1 = 1, X_2 = 1, X_3 = 0] &= 0 \\ \mathbb{P}[X_1 = 0, X_2 = 1, X_3 = 1] &= 0 & \mathbb{P}[X_1 = 1, X_2 = 1, X_3 = 1] &= \frac{1}{6} \end{aligned}$$

The important parts of the table are where exactly two out of the three variables are 1. Such events have a 0 probability, because once two variables are fixed, the third random variable gets fixed due to the dependence. In this case, the marginal PMFs of X_1 becomes

$$P_{X_1}(x_1) = \sum_{(x_2,x_3)\in\{0,1\}^2} \mathbb{P}[X_1 = x_1, X_2 = x_2, X_3 = x_3] = \begin{cases} \frac{1}{6} + \frac{1}{6} = \frac{1}{3} & x_1 = 1 \\ \frac{2}{3} & x_1 = 0 \\ 0 & \text{otherwise} \end{cases}$$

Similarly, the marginals of X_2 and X_3 can be written. Now, if we find the expected number of correctly placed cards using the formula in (3), we get

$$\mathbb{E}[N] = \mathbb{P}[X_1 = 1] + \mathbb{P}[X_2 = 1] + \mathbb{P}[X_3 = 1] = \frac{1}{3} + \frac{1}{3} + \frac{1}{3} = 1$$

The surprising fact is that this notion generalizes to n dependent variables.

Section 2

We find the expectation of N using the newly found PMF, $P_N(r)$, given in (11).

$$\mathbb{E}[N] = \sum_{r=0}^{n} r\ \mathbb{P}[N = r] = \sum_{r=1}^{n} \frac{1}{(r-1)!} \sum_{i=0}^{n-r} \frac{(-1)^i}{i!} = \sum_{r=0}^{n-1} \frac{1}{r!} \sum_{i=0}^{n-r} \frac{(-1)^i}{i!}$$

Using the Taylor Series expansion for the exponential,

$$e^x = \sum_{i=0}^{\infty} \frac{x^i}{i!} = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots$$

It is not hard to notice that the first summation resembles the Taylor's expansion for e^1 and the second resembles the same for e^{-1}. So, a reasonable attempt is to assume extremely large n.

$$\lim_{n\to\infty} \mathbb{E}[N] = \lim_{n\to\infty} \sum_{r=0}^{n-1} \frac{1}{r!} \sum_{i=0}^{n-r} \frac{(-1)^i}{i!} = \sum_{r=0}^{\infty} \frac{1}{r!} \sum_{i=0}^{\infty} \frac{(-1)^i}{i!} = e \cdot \frac{1}{e} = 1$$

For smaller n, let us try to use this observation.

$$\begin{aligned}\mathbb{E}[N] &= \sum_{r=0}^{n-1} \frac{1}{r!} \sum_{i=0}^{n-r} \frac{(-1)^i}{i!} = \sum_{r=0}^{n-1} \sum_{i=0}^{n-r} \frac{1}{r!} \cdot \frac{(-1)^i}{i!} \\ &= \sum_{r=0}^{n-1} \sum_{j=r}^{n} \frac{1}{r!} \cdot \frac{(-1)^{j-r}}{(j-r)!} \qquad \text{change of variables } (j = r + i)\end{aligned}$$

I'm pretty sure this sum is 1; I just can't prove it (yet).