601.433/633 Introduction to Algorithms **Fall 2024**
Homework #2 **Due:** September 12, 2024, 9am

Bella Lu
(Partners: Jecia Mao, Luna Liu)

1 Reversals (33 pts)

Given an array $[a_1, a_2, \ldots, a_n]$, a reversal is a pair (i, j) such that $i < j$ but $a_i > a_j$. For example, in the array $[5, 3, 2, 10]$ there are are three reversals $((1, 2), (1, 3), (2, 3))$. Note that the array has no reversals if and only if it is sorted, so the number of reversals can be thought of as a measure of how well-sorted an array is.

(a) (11 points) What is the expected number of reversals in a random array? More formally, consider a uniformly random permutation of n distinct elements $a_1, \ldots, a_n$: what is the expected number of reversals? Give your answer precisely, without asymptotic notation. Prove your answer.

The first element in the array can be reversed with $(n-1)$ other elements. The second element in the array can be reversed with $(n-2)$ other elements. $(n-1)+(n-2)+...+1$ equals to, by the arithmetic sum formula, $\frac{n(n-1)}{2}$. This is the same as $\binom{n}{2}$, because we are finding all possible combinations of reversals (which is a pair) where order doesn't matters since the pair (i, j) is the same as as (j, i). In other words, a_i being reversed with a_j is the same as a_j being reversed with a_i.

So, the total number of reversals an array can have is $\binom{n}{2}$.

Let k represent the kth (i, j) pair.

Let I_k be the integer variable that can only take on values of 0 or 1, where $I_k\begin{cases}0\text{: is not reversed}\\1\text{: is reversed}\end{cases}$

The expected number of reversals is $E(\sum_{k=1}^{\binom{n}{2}} I_k)$. By Linearity of Expectation, the expectation of the number of reversals equals to the sum of all the expectations, which is $\sum_{k=1}^{\binom{n}{2}} E(I_k)$. $E(I_k) = \frac{1}{2}$ because there are two outcomes, 0 and 1, where both are equally likely (i.e. either reversed or not). Thus, $\sum_{k=1}^{\binom{n}{2}} \frac{1}{2} = \binom{n}{2} \cdot \frac{1}{2}$.

$\therefore$ The expected number of reversals is $\frac{1}{2}\binom{n}{2}$. ■

(b) (11 points) Recall the insertion sort algorithm:

```
for i = 1 to n
```

```
    j = i
    while j > 0 and A[j-1] > A[j]
        swap A[j] and A[j-1]
        j = j - 1
```

Suppose that our array has d reversals. Prove that the (worst-case) running time of insertion sort is $\Theta(n + d)$.

For insertion sort, we need to traverse every element in A. In the outer for loop, it goes from 1 to n, which has $\Theta(n - 1 + 1) = \Theta(n)$ runtime. The inner while loop swaps every reversal pair back.

It is $n + d$ because there are d reversals in total, which means that every reversal we fix for each element, there is one less reversal left. In other words, the algorithm does d swaps in total after traversing the entire array.

For example, say the first iteration needs d_1 swaps, the second iteration needs d_2 swaps, ..., the last iteration needs d_n swaps. There are a total of n iterations, according to the outer for loop. Also, since the total reversals is d, $d_1 + d_2 + ... + d_n = d$.

$\therefore$ The total run time of the algorithm is $\Theta((1 + d_1) + (1 + d_2) + ... + (1 + d_n))$. There are n ones, and $d_1 + d_2 + ... + d_n = d$, so the running time of insertion sort is $\Theta(n + d)$. ■

(c) (11 points) What does this imply about the *average-case* running time of insertion sort as a function only of n? That is, if we draw a permutation uniformly at random, what is the expected running time of insertion sort (in asymptotic notation)? Note that this is *not* a randomized algorithm; this is a deterministic algorithm on a random input.

We proved in (b) that the running time of insertion sort is $\Theta(n + d)$. Also, we proved in (a) that the expected number of reversals is $\frac{1}{2}\binom{n}{2}$.

Let T be the running time. Let x be a random variable for the number of reversals.
Then $T = \Theta(n + x)$, which means that $T \leq c_1(n + x)$ and $T \geq c_2(n + x)$.

$E(T) \leq c_1(n + \text{expected number of reversals}) = c_1(n + \frac{1}{2}\binom{n}{2})$.
$E(T) \geq c_2(n + \text{expected number of reversals}) = c_2(n + \frac{1}{2}\binom{n}{2})$.

We can expand $n + \frac{1}{2}\binom{n}{2}$:

$$n + \frac{1}{2}\binom{n}{2}$$

$$= n + \frac{n^2 - n}{4}$$

$$= \frac{n^2}{4} + \frac{3n}{4}$$

Prove that $c_1(\frac{n^2}{4} + \frac{3n}{4}) = O(n^2)$:
$c_1(\frac{n^2}{4} + \frac{3n}{4}) \leq c_3 \cdot n^2$
Let $c = c_3/c_1$
$\frac{n^2}{4} + \frac{3n}{4} \leq c \cdot n^2$
We can choose $c = 1, n_0 = 1$
The statement holds.

Prove that $c_2(\frac{n^2}{4} + \frac{3n}{4}) = \Omega(n^2)$:
$c_2(\frac{n^2}{4} + \frac{3n}{4}) \geq c_3 \cdot n^2$
Let $c = c_3/c_2$
$\frac{n^2}{4} + \frac{3n}{4} \geq c \cdot n^2$
We can choose $c = 1/4, n_0 = 1$
The statement holds.

$\therefore$ Since $E(T) \leq c_1(\frac{n^2}{4} + \frac{3n}{4}) = O(n^2)$ and $E(T) \geq c_2(\frac{n^2}{4} + \frac{3n}{4}) = \Omega(n^2)$, the expected running time of insertion sort as a function of n is $\Theta(n^2)$. ■

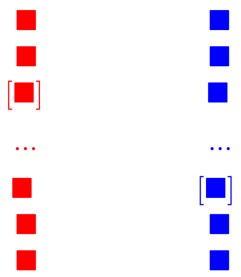
2 Pitcher Matching (34 points)

Suppose that you are given n red and n blue water pitchers, all of different shapes and sizes. All red pitchers hold different amounts of water, as do all the blue ones. Moreover, for every red pitcher there is a blue pitcher that holds exactly the same amount of water (and vice versa).

Your job is to find a matching between red pitchers and blue pitchers that hold the same amount of water. In other words, you want to identify for every red pitcher which blue pitcher has the same volume, and similarly for every blue pitcher which red pitcher has the same volume. To do this, you are only allowed to use the following operation: pick a red pitcher and a blue pitcher, fill the red pitcher, and pour it into the blue pitcher. This will tell you whether the volume of the red pitcher is less than, equal to, or greater than the volume of the blue pitcher. In other words, you can compare any red pitcher and any blue pitcher. But you *cannot* compare two red pitchers or two blue pitchers.

(a) (16 points) Give a randomized algorithm that uses $O(n \log n)$ comparisons in expectation.

Let the squares represent pitchers (assume they have different sizes):



1. Choose a random pivot from the red pitchers (i.e. [🟥]).

2. Compare the red pivot to every blue pitcher, and partition the blue pitchers by testing if the blue pitcher holds more or less water than the red pivot. Partition by putting the blue pitchers less than the red pivot to the left and the blue pitchers more than the red pivot to the right.

3. The blue pitcher that has the same volume as the red pivot (i.e. [🟦]) is now in the correct spot, relative to the blue pitchers.

4. Use the blue pitcher that was just placed into the correct spot, as the new pivot.

5. Compare the blue pivot to every red pitcher, and partition the red pitchers (essentially the same process as step 2).

6. Recursively sort the left and right side of the pivot, repeating the previous 5 steps.

$\therefore$ This process uses a method similar to randomized QuickSort, which has $O(n \log n)$ comparisons in expectation. Since we are doing it back and forth, the $n \log n$ will be multiplied by some constant, but it is still $O(n \log n)$. After sorting, since there is a blue pitcher that holds exactly the same amount of water of every red pitcher, the pitchers will be correctly matched (i.e. 1st in red column matches 1st in blue, 2nd in red matches 2nd in blue, etc.

(b) (9 points) Prove that your algorithm is correct (i.e., that it always returns the correct matching).

We can do a proof of strong induction.

Base case:
Let $n = 1$ (i.e. there is one red pitcher and one blue pitcher).
Since there are only 1 red and 1 blue pitcher to begin with, the algorithm with output the exact same input (no swapping is needed since there's only 1).
Since the original statement says for every red pitcher there will be one blue that matches, then therefore the algorithm at $n = 1$ must be correct because the two pitchers will have same volumes. The base case holds.

For the sake of induction, let's assume that for all length $n \leq k$, the algorithm is correct (i.e. there is a correct matching between red and blue pitchers).

For any chosen pivot in the $(k + 1)$ pitchers, after partitioning, it will split into two sections consisting of pitchers smaller and greater than the pivot. Each group (left and right), will have length $n \leq k$. By our inductive assumption, any length $n \leq k$ correctly matches the pitchers, and since the left and right sides' length $n \leq k$, the algorithm also holds for lengths $k + 1$.

$\therefore$ The algorithm is correct. ■

(c) (9 points) Prove that your algorithm uses $O(n \log n)$ comparisons in expectation.

Let the random pivot chosen be the kth smallest element. This means that:
1. The left side of the pivot: $|L| = k - 1$.
2. The right side of the pivot: $|G| = n - k$.

We recurse on each side, so the left side does $T(k-1)$ and the right side does $T(n-k)$. Each iteration does $2n-1$ work because one round of comparisons need to be made for a red pivot, the another round for the blue pivot, and one of them has the same volume.

Thus, the recursion can be written as $T(n) = T(k-1) + T(n-k) + (2n-1)$.

Let T be number of comparisons.
The probability of picking a pivot in an array of length n (i.e. $Pr(k)$) is $\frac{1}{n}$.

The expected number of comparisons is:

$$E(T) = \sum_{k=1}^{n} Pr(k) \cdot \text{number of comparisons}$$
$$= \sum_{k=1}^{n} \frac{1}{n} \cdot [T(k-1) + T(n-k) + (2n-1)]$$
$$= \sum_{k=1}^{n} \frac{1}{n} \cdot [T(k-1) + T(n-k)] + \sum_{k=1}^{n} \frac{1}{n} \cdot (2n-1)$$
$$= \tfrac{1}{n} \cdot \sum_{k=1}^{n} [T(k-1) + T(n-k)] + n \cdot \frac{(2n-1)}{n}$$
$$= \tfrac{2}{n} \cdot \sum_{k=1}^{n-1} T(k) + (2n-1)$$

We can use the guess and check method. Let's guess that $T(n) \leq cn \ln n$.

$$T(n) = \tfrac{2}{n} \cdot \sum_{k=1}^{n-1} T(k) + (2n-1)$$
$$\leq \tfrac{2}{n} \cdot \sum_{k=1}^{n-1} ck \ln k + (2n-1)$$
$$\leq \tfrac{2c}{n} \cdot \textstyle\int_1^n (x \ln x) dx + (2n-1)$$
$$= \tfrac{2c}{n} \cdot (\tfrac{1}{2}x^2 \ln x - \tfrac{1}{4}x^2)_{x=1}^{n} + (2n-1)$$
$$= \tfrac{2c}{n} \cdot (\tfrac{1}{2}n^2 \ln n - \tfrac{1}{4}n^2 + \tfrac{1}{4}) + (2n-1)$$

$$\leq cn \ln n - \tfrac{c}{2}n + \tfrac{c}{2n} + 2n - 1$$

$$\leq cn \ln n \text{ (for } c \geq 2)$$

$\therefore$ The algorithm uses $O(n \log n)$ comparisons in expectation. ■

3 Searching an Array Randomly (33 points)

Let A be an unsorted array of length n, where each entry of A is an integer. Suppose that we are looking for some integer x in A, i.e., we want to find an index i such that $A[i] = x$ if such an index exists. If no such index exists, we should return False. Consider the following randomized algorithm.

- Initially, all indices are unmarked.
- While not all indices are marked:
  - Pick an index $i \in [n]$ uniformly at random.
  - If $A[i] = x$ return i.
  - Else mark index i.
- Return false

Note that in each iteration we pick an index i uniformly at random from $[n]$, not from the set of unmarked indices. So we might examine a given index more than once.

(a) (16 points) Suppose that x appears in $k \geq 1$ places in A, i.e., $|\{i : A[i] = x\}| = k$. What is the expected running time of this algorithm, as a function of n and k?

Hint: if you are unfamiliar with Bernoulli and Geometric distributions, consider reading CLRS (3rd edition) Appendix C.4.

If A has n elements, and there are k elements that contain the value x, then the probability of choosing an element (i.e. success) equaling to x is $\frac{k}{n}$.

Thus, the probability of not choosing the correct index (i.e. failure) is $1 - \frac{k}{n}$.

Let X be the number of trails needed to obtain a success, which in this case would be the the number of trails needed to pick a correct index. In other words, the running time of this algorithm.

Let the zth trial be the first success.

Let p = probability of success = $\frac{k}{n}$. Let q = probability of failure = $1 - \frac{k}{n}$.

$$E[X] = \sum_{z=1}^{\infty} z q^{z-1} p$$

$$= \tfrac{p}{q}\sum_{z=1}^{\infty} kq^k$$

$$= \tfrac{p}{q} \cdot \tfrac{q}{(1-q)^2}$$

$$= \tfrac{p}{q} \cdot \tfrac{q}{p^2}$$

$$= \tfrac{1}{p}$$

$\therefore E[X] =$ expected running time $= \frac{1}{p} = \frac{1}{\frac{k}{n}} = \frac{n}{k}$. ■

(b) (17 points) Suppose that x does not appear in A. What is the expected running time of the algorithm?

When we first pick an element, the probability of selecting an unmarked element is $\frac{n}{n}$ because all of them start unmarked. This means that it will take an expected $\frac{1}{p} = 1$, with $p = \frac{n}{n}$, tries to pick an unmarked element.

After picking the first element, there are now $n-1$ unmarked elements. Thus, the probability of picking an unmarked element is now $\frac{n-1}{n}$. Again, using the Geometric distribution, the expected number of tries until we get another unmarked element is $\frac{1}{p} = \frac{n}{n-1}$.

Same pattern follows as we mark 3, 4, 5, ... elements in the array, with the terminating case when all elements have been marked (i.e. x does not appear in A). So, to find the expected running time, we can add up all the expected number of tries as we mark each element:

$$\tfrac{n}{n} + \tfrac{n}{n-1} + \tfrac{n}{n-2} + ... + \tfrac{n}{1}$$

$$= n(\tfrac{1}{n} + \tfrac{1}{n-1} + \tfrac{1}{n-2} + ... + \tfrac{1}{1})$$

$$= n\sum_{k=1}^{n} \frac{1}{k}$$

$$\leq nH_n \text{ (Harmonic Series)}$$

$$\leq n \log n$$

$$= O(n \log n)$$

$\therefore$ The expected running time of the algorithm is $O(n \log n)$. ■