**Design and Analysis of Algorithms, Fall 2014**
**Exercise V: Solutions**

---

**V-1** (**CLRS 5.1-3**$\star$) Suppose that you want to output 0 with probability $1/2$ and 1 with probability $1/2$. At your disposal is a procedure BIASED-RANDOM, that outputs either 0 or 1. It outputs 1 with some probability $p$ and 0 with probability $1-p$, where $0 < p < 1$, but you do not know what $p$ is. Give an algorithm that uses BIASED-RANDOM as a subroutine, and returns 0 with probability $1/2$ and 1 with probability $1/2$. What is the expected running time of your algorithm as a function of $p$.

Our algorithm is as follows:

```
UNBIASED-RANDOM()
   Output: 0 with probability 1/2 and 1 with probability 1/2
1  while true do
2      $a \leftarrow$ BIASED-RANDOM()
3      $b \leftarrow$ BIASED-RANDOM()
4      if $a < b$ then return 0
5      if $a > b$ then return 1
```

The algorithm calls BIASED-RANDOM twice to get two random numbers $A$ and $B$. It repeats this until $A \neq B$. Then, depending on whether $A < B$ (that is, $A = 0$ and $B = 1$) or $A > B$ (that is, $A = 1$ and $B = 0$) it returns 0 or 1 respectively.

In any iteration, we have $\Pr(A < B) = p(1-p) = \Pr(B < A)$, that is, the probability that the algorithm returns 0 in that iteration equals to the probability that it returns 1 in that iteration. Since with probability 1 we return something at some point (and not repeat the loop endlessly) and the probabilities of returning 0 and 1 are equal in each iteration, the total probabilities of returning 0 and 1 must be $1/2$ and $1/2$ respectively.

The algorithms stops, if it either returns 0 or 1. In every iteration, the probability of this is $\Pr(A \neq B) = \Pr(A < B) + \Pr(B < A) = 2p(1-p)$. Thus, we have a sequence of independent Bernoulli trials, each with probability $2p(1-p)$ of success. Therefore, the number of iterations required before the algorithm stops is geometrically distributed with parameter $2p(1-p)$, and the expected number of iterations is $1/(2p(1-p))$. As each iteration takes constant time (assuming that BIASED-RANDOM takes constant time), the expected running time of the algorithm is $\Theta(1/(p(1-p)))$.

---

**V-2** (**CLRS 5.2-5**) Let $A[1..n]$ be an array of $n$ distinct numbers. If $i < j$ and $A[i] > A[j]$, then the pair $(i, j)$ is called an *inversion* of $A$. Suppose that the elements of $A$ form a uniform random permutation of $(1, 2, \ldots, n)$. Use indicator random variables to compute the expected number of inversions.

For $i, j$ such that $1 \leq i < j \leq n$, let $I_{ij}$ be an indicator variable for the event that the pair $(i, j)$ is an inversion. The total number of inversions is then $\sum_{i<j} I_{ij}$. For each pair $i < j$ the probability of inversion is $1/2$ and thus we have $\mathbf{E}[I_{ij}] = 1/2$. As there are $n(n-1)/2$ such pairs, the expected number of inversions is

$$\mathbf{E}\left[\sum_{i<j} I_{ij}\right] = \sum_{i<j} \mathbf{E}[I_{ij}] = \sum_{i<j} 1/2 = \frac{n(n-1)}{4}.$$

**V-3** (**CLRS 5.3-3**) Suppose that instead of swapping element $A[i]$ with a random element from the subarray $A[i..n]$, we swapped it with a random element from anywhere in the array:

```
PERMUTE-WITH-ALL($A$, $n$)
1  for $i$ = 1 to $n$
2      swap $A[i]$ with $A[\text{RANDOM}(1,n)]$
```

Does this code produce a uniform random permutation? Why or why not?

No. In each of $n$ iterations the algorithm chooses the index $i$ independently and uniformly at random from set $\{1,\dots,n\}$. This means that there are $n^n$ different possible sequences each has probability $1/n^n$. On the other hand, there are $n!$ distinct permutations, and to get a uniform distribution over permutations, the probability of each should be $1/n!$. Thus, we should have $k/n^n = 1/n! \Leftrightarrow n^n = kn!$, where $k$ is an integer. But this is not possible in general, since in general $n^n$ is not divisible by $n!$ (consider a prime $n > 2$ for example).

(In fact, $n^n$ is not divisible by $n!$ unless $n = 1$ or $n = 2$. To see this, assume that $n > 2$ and let $p > 1$ be any prime that divides $n-1$. (If $n-1$ is a prime, then $p = n-1$, otherwise $p < n-1$.) Now, if $p$ divides also $n$, it should also divide the difference $n-(n-1) = 1$. However, this cannot be true, since we assumed that $p > 1$. Therefore, $n$ is not divisible by $p$. But this means, that $n^n$ is not divisible by $p$ (as the prime decomposition of $n^n$ contains only primes that appear in the decomposition of $n$). On the other hand, because $p$ divides $n-1$ which divides $n!$, $p$ also divides $n!$. Similarly, if $n!$ divides $n^n$, $p$ would also divide $n^n$. Since this is not the case, $n^n$ cannot be divisible by $n!$.)

**V-4** (**CLRS 5.3-5**$\star$) Prove that in the array $P$ in procedure PERMUTE-BY-SORTING, the probability that all elements are unique is at least $1 - 1/n$.

For $i, j$ such that $1 \le i < j \le n$, let $E_{ij}$ denote the event that elements $P[i]$ and $P[j]$ are identical. Since the elements in $P$ are chosen independently and uniformly at random from values 1 to $n^3$, we have $\Pr(E_{ij}) = 1/n^3$ for all pairs $i, j$. The event that not all elements are unique, that is, there is at least one pair of identical elements, is $\bigcup_{i<j} E_{ij}$. Therefore, the probability that all elements are unique is

$$\begin{aligned}
\Pr\left(\left(\bigcup_{i<j} E_{ij}\right)^c\right) &= 1 - \Pr\left(\bigcup_{i<j} E_{ij}\right) \\
&\ge 1 - \sum_{i<j} \Pr(E_{ij}) \\
&= 1 - \frac{n(n-1)}{2} \cdot \frac{1}{n^3} \\
&= 1 - \frac{1}{2n} + \frac{1}{2n^2} \\
&\ge 1 - 1/n.
\end{aligned}$$

The first inequality follows from direct usage of the union bound (also known as Boole's inequality).

**V-5** (**CLRS 8-4 Water jugs**) Suppose that you are given $n$ red and $n$ blue water jugs, all of different shapes and sizes. All red jugs hold different amounts of water, as do the blue ones. Moreover, for every red jug, there is a blue jug that holds the same amount of water, and vice versa.
It is your task to find a grouping of the jugs into pairs of red and blue jugs that hold the same amount of water. To do so, you may perform the following operation: pick a pair of jugs in which one is red and one is blue, fill the red jug with water, and then pour the water into the blue jug. This operation will tell you whether the red or the blue jug can hold more water, or if they are of the same volume. Assume that such a comparison takes one time unit. Your goal is to find an algorithm that makes a minimum number of comparisons to determine the grouping. Remember that you may not directly compare two red jugs or two blue jugs.

**a.** Describe a deterministic algorithm that uses $O(n^2)$ comparisons to group the jugs into pairs.

Pick a red jug and compare it against all blue jugs until the matching jug is found, and then pair these two jugs. Repeat this for all red jugs.

There are $n$ red jugs and each needs at most $n$ comparisons to find the matching blue jug. Thus, the algorithm uses $O(n^2)$ comparisons.

**c.** Give a randomized algorithm whose expected number of comparisons is $O(n \log n)$, and prove that this bound is correct. What is the worst-case number of comparisons for your algorithm?

This problem can be solved by emulating the quicksort algorithm with small modification. In particular, we perform quicksort on both sets simultaneously, and since we cannot do comparisons between jugs of the same color, we use a jug of the other color as the pivot element.

In the following, we will use the same notation to refer to a jug and its size. That is, if $r$ is a jug then its size is also denoted by $r$. The algorithm is as follows:

```
ORGANIZE($R$, $B$)
    Input: a set of red jugs $R$ and a set of blue jugs $B$
 1  if $|R| = |B| = 1$ then
 2      pair the remaining two jugs
 3  else
 4      pick $r \in R$ uniformly at random
 5      compare $r$ against all elements of $B$ and set
 6          $b \leftarrow$ blue jug with the same size as $r$
 7          $B_< \leftarrow \{b \in B \mid b < r\}$
 8          $B_> \leftarrow \{b \in B \mid b > r\}$
 9      compare $b$ against all elements of $R$ and set
10          $R_< \leftarrow \{r \in R \mid r < b\}$
11          $R_> \leftarrow \{r \in R \mid r > b\}$
12      pair jugs $r$ and $b$
13      ORGANIZE($R_<$, $B_<$)
14      ORGANIZE($R_>$, $B_>$)
```

It remains to analyze the expected number of comparisons done by this algorithm. As with the algorithm itself, we mimic the analysis of randomized quicksort. For the purposes of our analysis, we assume that the set of red jugs is

$$R = \{r_1, r_2, \dots, r_n\},$$

where $r_i < r_j$ for all $i < j$, and similarly, the set of blue jugs is

$$B = \{b_1, b_2, \dots, b_n\},$$

where $b_i < b_j$ for all $i < j$. Thus the algorithm will pair $r_i$ with $b_i$ for all $i$.

Let $X$ be a random variable that counts the number of comparison done by our algorithm. Denote by $X_{ij}$ the indicator random variable for the event that $r_i$ and $b_j$ are compared, that is,

$$X_{ij} = \begin{cases} 1 & \text{if } r_i \text{ and } b_j \text{ are compared} \\ 0 & \text{otherwise.} \end{cases}$$

We have that $X = \sum_{i=1}^{n} \sum_{j=1}^{n} X_{ij}$. Note that this is slightly different from the quicksort analysis, as here we have to consider all pairs $i, j$, not just pairs with $i < j$. By linearity of expectation, we get

$$\mathbf{E}[X] = \mathbf{E}\left[\sum_{i=1}^{n}\sum_{j=1}^{n} X_{ij}\right] = \sum_{i=1}^{n}\sum_{j=1}^{n} \mathbf{E}[X_{ij}].$$

Since $X_{ij}$ is an indicator random variable, we have $\mathbf{E}[X_{ij}] = \Pr(r_i \text{ and } r_j \text{ are compared})$. Assume that $i \neq j$. Due to symmetry we may assume $i < j$. Let $R_{ij} = \{r_i, r_{i+1}, \ldots, r_j\}$ and $B_{ij} = \{b_i, b_{i+1}, \ldots, b_j\}$. It is easy to verify that our algorithm compares $r_i$ and $b_j$ if and only if either $r_i$ or $r_j$ is the first element from $R_{ij}$ that is selected as pivot by the algorithm. To see this, observe that if $r_i$ is the first pivot from $R_{ij}$, it will be compared against all jugs in $B_{ij}$, including $b_j$. On the other hand, if $r_j$ is the first pivot, then $b_j$ will be compared against all jugs in $R_{ij}$, including $r_i$. In all other cases $r_i$ ends up in $R_<$ while $b_j$ ends up in $B_>$ and thus they are never compared. Observing that each element of $R_{ij}$ is equally likely to picked as the pivot first, we get that

$$\begin{aligned}
\Pr(r_i \text{ and } r_j \text{ are compared}) &= \Pr(r_i \text{ is picked first}) + \Pr(r_j \text{ is picked first}) \\
&= \frac{1}{j-i+1} + \frac{1}{j-i+1} \\
&= \frac{2}{j-i+1}.
\end{aligned}$$

Thus, we have that $\mathbf{E}[X_{ij}] = \mathbf{E}[X_{ji}] = \frac{2}{|j-i|+1}$ for all $i \neq j$. For the case $i = j$, observe that $r_i$ is always eventually compared with $b_i$. Thus, $\mathbf{E}[X_{ii}] = 1$ for all $i$. Combining this with our earlier analysis, we get

$$\begin{aligned}
\mathbf{E}[X] &= \sum_{i=1}^{n}\sum_{j=1}^{n} \mathbf{E}[X_{ij}] \\
&= \sum_{i=1}^{n} \mathbf{E}[X_{ii}] + \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} (\mathbf{E}[X_{ij}] + \mathbf{E}[X_{ji}]) && \text{(regroup)} \\
&= n + \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} 2\mathbf{E}[X_{ij}] && \text{(symmetry)} \\
&= n + \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} \frac{4}{j-i+1} \\
&= n + \sum_{i=1}^{n-1}\sum_{k=1}^{n-i} \frac{4}{k+1} && \text{(change of variables)} \\
&< n + 4\sum_{i=1}^{n-1}\sum_{k=1}^{n-i} \frac{1}{k} && \left(\frac{1}{k} > \frac{1}{k+1} \text{ for all } k\right) \\
&< n + 4\sum_{i=1}^{n-1} \ln(n-i+1) && \left(\sum_{k=1}^{m} \frac{1}{k} < \ln(m+1)\right) \\
&= O(n) + O(n\log n) \\
&= O(n\log n).
\end{aligned}$$