Chapter 5. Probabilistic Analysis and Randomized Algorithms

5.1-1.
Candidate comparison is reflexive, antisymmetric and transitive, so it introduces a total order.

5.1-2.

Algorithm 1: RANDOM(a, b)

```
if a = b then
  | return a
end
m = a + (b − a)/2;
r = RANDOM(0, 1);
if r = 0 then
  | return RANDOM(a, ⌊m⌋);
else
  | return RANDOM(⌈m⌉, b);
end
```

The expected running time is $\Theta(\lg(b-a))$.

5.1-3.
If we call $\text{BIASED}-\text{RANDOM}$ twice, then the probability of getting $0-1$ and $1-0$ is same, so make use of them; that is, if we get $0-1$, return 0, and if we get $1-0$, return 1. Otherwise, call $\text{BIASED}-\text{RANDOM}$ once again. This procedure returns 0 and 1 evenly, described as below:

Algorithm 2: UNBIASED-RANDOM

```
while true do
    a = BIASED − RANDOM;
    b = BIASED − RANDOM;
    if a ≠ b then
        return a;
    end
end
```

The expected running time is $\Theta(1/(2p(1-p)))$.

5.2-1.
The probability that you hire exactly one time occurs if the first candidate is indeed the best, which has a probability of $1/n$.
The probability that you hire exactly n time occurs if the candidates arrives at increasing order of quality, which has a probability of $1/n!$.

5.2-2.
To hire exactly twice, exactly one candidate after the first candidate should be hired. That is, the quality of candidates should be in decreasing order with one increase. To hire k-th candidate as the second and the final candidate, we first pick k candidates, sort in decreasing order, and sort another $n-k$ candidates, sort in decreasing order and concatenate them. The only wrong case is that every elements are in decreasing order after concatenation. The number of cases is $\binom{n}{k}-1$, so the probability is

$$\frac{\sum_{k=2}^{n}(\binom{n}{k}-1)}{n!} = \frac{2^n-n-1}{n!}$$

5.2-3.

$$\mathbb{E}[X] = \mathbb{E}[\sum_i X_i] = \sum_i \mathbb{E}[X_i] = \sum_i \sum_{j=1}^{6} \frac{j}{6} = 3.5 \cdot n.$$

5.2-4.
Let X be the number of customers who get back their hat and X_i be the indicator random variable that the customer i gets the hats back. $P(X_i) = 1/n$, so $\mathbb{E}[X] = \mathbb{E}[\sum_i X_i] = \sum_i \mathbb{E}[X_i] = \sum_i \frac{1}{n} = 1$.

5.2-5.
Let $X_{i,j}$ for $i < j$ be the indicator random variable that $A[i] > A[j]$.

$$\mathbb{E}[\sum_{i<j} X_{i,j}] = \sum_{i<j} \mathbb{E}[X_{i,j}] = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} P(A[i] > A[j]) = \sum_{i=1}^{n-1} \frac{1}{2} \cdot (n-i) = \frac{n(n-1)}{4}.$$

5.3-1.
This modification ensures the initialization of the loop invariant.

Algorithm 3: RANDOMIZE-IN-PLACE(A)

```
n = A.length;
swap A[1] with A[RANDOM(1, n)];
for i = 2 to n do
 | swap A[i] with A[RANDOM(i, n)];
end
```

5.3-2.
No. This algorithm can never produce $[3, 2, 1]$. For the first iteration of the loop, $A[1]$ should be swapped with $A[3]$, but at the next iteration $A[2]$ is swapped with $A[3]$, resulting $[3, 1, 2]$.

5.3-3.
No. Consider the case of $n = 3$, then the possible ending result would be 27. However, there can be $3! = 6$ orderings, so they cannot be appear equally likely.

5.3-4.
The probability that $A[i]$ winds up in any position j is equal to $i + n \equiv j (mod n)$, which is $1/n$.
The algorithm doesn't actually generate uniformly random permutations, it just rotates the array.

5.3-5.
The probability that all elements are unique is

$$1(1 - \frac{1}{n^3}) \cdots (1 - \frac{n}{n^3}) \geq (1 - \frac{n}{n^3})^n \geq 1 - \frac{1}{n}.$$

5.3-6.

Algorithm 4: PERMUTE-BY-SORTING(A)

let $P[1, ...n]$ be a new array **for** $i = 1$ *to* n **do**
  $P[i] = 1;$

for $i = 1$ *to* n **do**
  swap $P[i]$ with $P[\text{RANDOM}(i, n)]$;

5.3-7.
For $m = 0$, it is obviously true. Suppose S is a uniform $m-1$ subset of $n-1$. For $j \in \{1, \cdots, n-1\}$,

$$P(j \in S') = P(j \in S) + P(j \notin S \cup i = j) = \frac{m-1}{n-1} + (1 - \frac{m-1}{n-1})\frac{1}{n} = \frac{m}{n}.$$

This probability is equally distributed for $j \in \{1, \cdots, n-1\}$, so it should be also equally distributed for $j \in \{1, \cdots, n\}$.

5.4-1.

$$1 - (\frac{364}{365})^k \geq \frac{1}{2} \Rightarrow k \geq 263.$$
$$1 - k \cdot \frac{1}{365}(\frac{364}{365})^{k-1} \geq \frac{1}{2} \Rightarrow k \geq 115.$$

5.4-2.
This is a restatement of the birthday problem. The answer is;

$$1 + \sum_{k=1}^{b} \frac{b!}{(b-k)!\, b^k}$$

5.4-3.
The all comparison in the analysis is pairwise, so pairwise independence is enough.

5.4-4.

$$\mathbb{E}[X] = \sum_{i=1}^{n} \sum_{j=i+1}^{n} \sum_{k=j+1} n\frac{1}{365^2} = \binom{n}{3}\frac{1}{365^2} = \frac{n(n-1)(n-2)}{6 * 365^2} > 1 \Rightarrow n = 94.$$

5.4-5.

$$1 \cdot \frac{n-1}{n} \cdots \frac{n-k+1}{n} = \frac{(n-1)!}{(n-k)!\, n^k}.$$

5.4-6.
Let X_i be the indicator random variable that bin i is empty after all balls are tossed, and X be the sum.

$$\mathbb{E}[X] = \sum_{i=1}^{n} \mathbb{E}[X_i] = \sum_{i=1}^{n} (\frac{n-1}{n})^n = n(\frac{n-1}{n})^n.$$

Let Y_i be the indicator random variable that bin i contains exactly one ball after all balls are tossed, and Y be the sum.

$$\mathbb{E}[Y] = \sum_{i=1}^{n} \mathbb{E}[Y_i] = \sum_{i=1}^{n} \binom{n}{1} \frac{1}{n} (\frac{n-1}{n})^{n-1} = n(\frac{n-1}{n})^{n-1}.$$

5.4-7.
Let $s = \lg n - 2 \lg \lg n$ and split n flips to n/s groups. The probability that one group comes up with all heads is

$$\frac{1}{2^s} = \frac{\lg^2 n}{n}.$$

The probability that no one of groups comes up with all heads is

$$(1 - \frac{\lg n^2}{n})^{\frac{n}{s}} \le e^{-\frac{\lg^2 n}{\lg n - 2 \lg \lg n}} = n^{-1-\frac{2 \lg \lg n}{\lg n - 2 \lg \lg n}} < n^{-1}.$$

5-1.
a. For each INCREMENT operation, $n_{i+1} - n_i$ is increased with the probability of $1/(n_{i+1} - n_i)$, so the expected value of increase is 1, therefore, the expected increase after n INCREMENT operations is n.
b. In this case, for each INCREMENT operation, the probability that the

value is changed is $1/100$, so this can be regarded as a binomial distribution with $p = 0.01$, but increasing 100 for each step. The variance is $100n \cdot 0.01 \cdot 0.99 = 0.99n$.

5-2.
a.

Algorithm 5: RANDOM-SEARCH(x, A)

```
n = A.length;
S = ∅;
while —S— != n do
    i = RANDOM(1, n);
    if A[i] = x then
        return i;
    else
        S = S ∪ {i};
    end
end
```

---

b. The expected number of picks is n, since the probability that index i is selected is $1/n$.
c. Similar with b., the expected number of picks is n/k.
d. This is equivalent to the balls and bins problem, the answer is $n(\ln n + O(1))$.
e. The worst case running time is n, the average case running time is $(n+1)/2$.
f. The worst case running time is $n - k + 1$.
For analyzing the average case running time, let X be a set of matches and Y be a set of non-matches. An element in X is read iff it comes before every other element of X, which has probability $1/k$.
An element in Y is read iff it comes before every other element of X, which has probability $1/(k+1)$. Hence, the expected running time is

$$\frac{n-k}{k+1} + \frac{k}{k} = \frac{n+1}{k+1}.$$

g. Both the average case/worst case running time is n.
h. Same with DETERMINISTIC-SEARCH.
i. DETERMINISTIC-SEARCH. SCRAMBLE-SEARCH is useless for improving time complexity.