Introduction To Algorithm
Third Edition
Answer

Xia Ding

January 5, 2016

5.1

5.1–1

Partial order: A relation which is reflexive, anti-symmetric and transitive.

Total order: A total order that is a partial order which has *totality* property ($\forall a\ b \in S$: aRb or bRa).

First, we prove the partial order.

- *Reflexive*: Because everybody is as good or better as themselves.
- *Transitive*: If A is better than B and B is better than C, then A is better than C.
- *Anti-symmetric*: If A is better than B, than B is not better than A.

Now, the set S is partial order, because any two candidates are comparable, so the set S is total order.

5.1–2

Let $n = b - a$. The algorithm is as follows:

1. find the smallest integer c such that $2^c \geq n$, that is $c = \lceil \lg n \rceil$.
2. call RANDOM (0, 1) c times to get a c-digit binary number r.
3. if $r > n$ we go back to the step 2.
4. return $a + r$.

This produces a uniformly random number in that range. However, there is a possibility to have to repeat step 2. There is $p = \frac{n}{2c}$ chance of not having to repeat step two. The geometric distribution suggests that on average it takes $\frac{1}{p}$ trials before we get such a number, that is $\frac{2c}{n}$ trials. Since we perform c calls to RANDOM (0, 1) on each trial, the expected running time is

$$O(\frac{2^c}{n}c) = O(\frac{\lg(b-2)2^{\lg(b-a)}}{b-a}) = O(\lg(b-a))$$

5.1–3

1. Generate two random numbers x and y
2. If they are not same, return x
3. Otherwise, repeat from step 1

Because $P[0,1] = P[1,0] = 2p(1-p)$, so number of trails is $\frac{1}{2p(1-p)}$. So running time is $O(\frac{1}{p(1-p)})$.

5.2

5.2–1

When the best candidate is the first one fired, then you hire exactly one time, $P = \frac{(n-1)!}{n!} = \frac{1}{n}$. When the order of candidates is increasing, you will hire n times, $P = \frac{1}{n!}$.

5.2–2

You hire twice when your first fired candidate is rank $i(i < n)$ and all candidates with rank k such that $i < k < n$ come after candidate with rank n. Let X_i be the case that first candidate is rank i and you hire exactly twice. Because you first hire candidate ranked i has probability $\frac{1}{n}$, and the best candidate comes first in rest $n - i$ better candidate has probability $\frac{1}{n-i}$. So,

$$P(X_i) = \frac{1}{n}\frac{1}{n-1}$$

So the probability of hiring exactly twice is:

$$P(X) = \sum_{i=1}^{n-1} P(X_i) = \frac{1}{n}\sum_{i=1}^{n-1}\frac{1}{i} = \frac{1}{n}(\lg(n-1) + O(1))$$

5.2–3

First, we analysis one dice. Let X_i be the indicator such that $X_i = Ithisdiceshowsnumberi$. Let Y_j be the random variable denoting the number got from the $j-th$ dice. Then $Y_j = \sum_{i=1}^{6} X_i$, because $E[X_A] = \Pr\{A\}$,

$$\begin{aligned} E[Y_j] &= E[\sum_{i=1}^{6} X_i] \\ &= \sum_{i=1}^{6} E[X_i] \\ &= \sum_{i=1}^{6} i \Pr\{i\} \\ &= \sum_{i=1}^{6} i\frac{1}{6} \\ &= 3.5 \end{aligned}$$

So the expect value of sum of n dice is:

$$E[Y] = E[\sum_{j=1}^{n} Y_j] = \sum_{j=1}^{n} 3.5 = 3.5n$$

5.2–4

Define a random variable X that equals the number of customers that get back their own hat, so that we want to compute $E[X]$. For $i = 1, 2, 3 \dots n$, define the indicator random variable
$X_i = I\{customerigetsbackhisownhat\}$.
Then $X = X_1 + X_2 + \cdots + X_n$.
Since the ordering of hats is random, each customer has a probability of $1/n$ of getting back his hat. In other word, $\Pr\{X_i = 1\} = 1/n$, which, implies that $E[X_i] = 1/n$. Thus:

$$E[X] = E[\sum_{i=1}^{n} X_i] = \sum_{i=1}^{n} \frac{1}{n} = 1$$

5.2–5

Let X_{ij} be an indicator random variable for the event where the pair $A[i], A[j]$ for $i < j$ is inverted, i.e., $A[i] > A[j]$. More precisely, we define $X_{ij} = IA[i] > A[j]$ for $1 \leq i < j \leq n$. We have $\Pr X_{ij} = 1 = 1/2$, because given two distinct random numbers, the probability that the first is bigger than the second is 1/2, By Lemma 5.1, $E[X_{ij} = 1/2]$.

Let X be the random variable denoting the total number of inverted pairs in the array, so that

$$X = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} X_{ij}$$

We want the expected number of inverted pairs, so we take the expectation of both sides of the above equation to obtain

$$\begin{aligned} E[X] &= E[\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} X_{ij}] \\ &= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{1}{2} \\ &= \binom{n}{2} \frac{1}{2} \\ &= \frac{n(n-1)}{4} \end{aligned}$$

5.3

5.3–1

Before enter the loop, you can pick a random number in A and swap it with A[1]. Then start the loop with $i = 2$. And modify the Proof's **Initialization:** to "Consider the situation just before the first loop iteration, so that $i = 2 \quad \cdots$."

5.3–2

Although it will not produce the identity permutation there are other permutations that it fails to produce. For example, consider its operation when $n = 3$, when it should be able to produce the $n! - 1 = 5$ non-identity permutations. The for loop iterates for $i = 1$ and $i = 2$. When $i = 1$, the call to RANDOM returns one of two possible values (either 2 or 3), and when $i = 2$, the call to RANDOM returns just one value (3). Thus, this procedure can produce only 2 possible permutations, rather than the 5 that are required.

5.3–3

No. Because it produce n^n different permutations, not $n!$.

5.3–4

Because once *offset* is determined, the entire permutation is determined. And B is the result of shifting A, each value of *offset* occurs with probability $1/n$, so $A[i]$ has a $1/n$ probability of winding up in any particular position in B.

However, there are only n different permutations, not $n!$, so the result is not uniformly random.

5.3–5

$$\begin{aligned} P &= \prod_{i=0}^{n-1} \frac{n^3 - i}{n^3} \\ &\geq (1 - \frac{1}{n^2})^n \\ &\geq 1 - \frac{1}{n} \qquad (Bernoulli's \quad inequality) \end{aligned}$$

5.3–6

Generate new properties and retry.

5.3–7

We use a loop invariant:

> RANDOM-SAMPLE($i, n - m + i$) produce a random i-subset S of $\{$1,2,3,... $n - m + i\}$ for $i = 0, 1, 2, \ldots m$

Initialization: When $i = 0$, RANDOM($0, n - m$) return $\emptyset$ with probability 1, which is $\binom{n-1}{0}$.

Maintenance: We assume RANDOM($i - 1, n - m - 1 + i$) produce a random ($i - 1$)-subset of $\{$1,2,3,... $n - m - 1 + i\}$, with each possible result have probability $1/\binom{n-m-1+i}{i-1}$. Then after call RANDOM($i, n - m + i$), the probability of i-subset containing $n - m + i$ is

$$\frac{i}{n - m + i} \frac{1}{\binom{n-m-1+i}{i-1}} = \frac{1}{\binom{n-m+i}{i}}$$

If it doesn't contain $n - m + i$, it may contain one of $n - m$ numbers, probability of each is:

$$\frac{n - m}{n - m + i} \frac{1}{\binom{n-m-1+i}{i}} = \frac{1}{\binom{n-m+i}{i}}$$

So, RANDOM($i, n - m + i$) produce a random random i-subset S of $\{$1,2,3,... $n - m + i\}$

Termination: At termination, $i = m$, and we conclude that RANDOM(m, n) produce a random m-subset of $\{$1,2,3,... $n\}$.

5.4

5.4–1

Problems

5.1 *Probabilistic counting*

a. Suppose at the start of the $m-th$ INCREMENT OPERATION, the value of counter is i, if i increase, the number represented by i will increase by $n_{i+i} - n_i$. We use X_m be the value of increment of number represented by i.

$$\begin{aligned}
E[X_m] &= 0 \Pr\{i\ don't\ increase\} + (n_{i+1} - n_i) \Pr\{i\ increase\} \\
&= 0 \cdot (1 - \frac{1}{n_{i+1} - n_i}) + (n_{i+1} - n_i) \cdot \frac{1}{n_{i+1} - n_i} \\
&= 1
\end{aligned}$$

The expected value of increment of number represented by i is 1 during each INCREMENT operation, so after n INCREMENT OPERATION, expected value represented by counter is exactly n.

b.

$$\begin{aligned}
Var[X_m] &= E[X_m^2] - E^2[X_m] \\
&= 0^2 \cdot \frac{99}{100} + 100^2 \cdot \frac{1}{100} - 1^2 \\
&= 99
\end{aligned}$$

So variance of single INCREMENT operation is 99, after n INCREMENT operation,

$$Var[X] = Var[\sum_{i=1}^{n} X_i] = \sum_{i=1}^{n} Var[X_i] = 99n$$

5.2 *Searching an unsorted array*

a. Algorithm 1.

b. Because probability of getting i is $1/n$, so the expected value of trails is n.

c. Because probability of getting i is k/n, so the expected value of trails is n/k.

d. According to balls and bins model in 5.4.2, expected value is $n(\ln n + O(1))$.

e. The worst-case running time is n. The average-case is $(n+1)/2$.

Algorithm 1 RANDOM-SEARCH(A, x)

```
S = ∅ {S is a set}
while |S| ≠ n do
    i = RANDOM(1, n)
    if A[i] ≠ x then
        S = S ∩ i
    else
        return i
    end if
end while
return None
```

f. The worst-case running time is $n - k + 1$. Let X_i be an indicator random variable representing $A[i] = x$. $\Pr\{X_i\} = \frac{1}{k+1}$. Let Y be the indicator random variable representing after $n - k$ search, we find $A[i] = x$. $\Pr Y = 1$.

$$E[X] = E[X_1 + X_2 + X_3 + \cdots + X_{n-k} + Y] = 1 + \frac{n-k}{k+1} = \frac{n+1}{k+1}$$

g. Both of them are n.

h. The same as solution of part (**f**), replacing "average-case" with "expected".

i. DETERMINISTIC-SEARCH. Because SCRAMBLE-SEARCH will first permute the array randomly, which will cost $O(n)$ time.