Algorithms HW4

Pg 122

5.2-3)

$$X = \begin{cases} 1 & 1/6 \\ 2 & 1/6 \\ 3 & 1/6 \\ 4 & 1/6 \\ 5 & 1/6 \\ 6 & 1/6 \end{cases}$$

$$E[X] = (6+5+4+3+2+1)/6 = \frac{7}{2}$$

$$\therefore \sum_{i=1}^{n} E[X] = n \cdot \frac{7}{2} = \boxed{\frac{7n}{2}}$$

5.2-5)

$$X_{ij} = \begin{cases} 1 & \text{pair } (i,j) \text{ are inversion of } A \\ 0 & \text{pair } (i,j) \text{ not inversion of } A \end{cases}$$

Given 2 random numbers, probability that the first is greater than the second is $1/2$.

$$\therefore \Pr(X_{ij} = 1) = 1/2$$

Hence

$$E[X_{ij}] = 1 \cdot \tfrac{1}{2} + 0 \cdot \tfrac{1}{2} = 1/2$$

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

$$= (n-1) \cdot [(n-1) + (n-2) + \dots + 1] \cdot E[X_{ij}]$$

$$= \frac{(n-1) \cdot (n-1+1)}{2} \cdot E[X_{ij}]$$

$$= \frac{n \cdot (n-1)}{2} \cdot \frac{1}{2}$$

$$= \boxed{\frac{n \cdot (n-1)}{4}}$$

13	19	9	5	12	8	7	4	21	2	6	11
13	19	9	5	12	8	7	4	21	2	6	11
13	19	9	5	12	8	7	4	21	2	6	11
9	19	13	5	12	8	7	4	21	2	6	11
9	5	13	19	12	8	7	4	21	2	6	11
9	5	13	19	12	8	7	4	21	2	6	11
9	5	8	19	12	13	7	4	21	2	6	11
9	5	8	7	12	13	19	4	21	2	6	11
9	5	8	7	4	13	19	12	21	2	6	11
9	5	8	7	4	13	19	12	21	2	6	11
9	5	8	7	4	2	19	12	21	13	6	11
9	5	8	7	4	2	6	12	21	13	19	11
9	5	8	7	4	2	6	11	21	13	19	12

5-2.c) Similar to problem 5-2.b, but now we have k valid elements.

Hence

$$E[N] = \sum_{i \geq 1} i \cdot \left(1 - \frac{k}{n}\right)^{i-1} \cdot \left(\frac{k}{n}\right)$$

$$= \frac{k}{n} \cdot \frac{1}{\left(1 - \left(1 - \frac{k}{n}\right)\right)^2}$$

$$= \frac{k}{n} \cdot \frac{1}{\left(\frac{k}{n}\right)^2}$$

$$= \boxed{\frac{n}{k}}$$

5-2.d) $N :=$ random variable denoting the number of searches required to find n numbers from A

The i^{th} stage consist of the searches after the $(i-1)$st find until the ith find.

Let $N_i :=$ # of searches at i^{th} stage

$$E[N_i] = \frac{n}{n - i + 1}$$

Hence,

$$E[N] = \sum_{i=1}^{n} E[N_i]$$

$$= \sum_{i=1}^{n} \frac{n}{n - i + 1}$$

$$= n \cdot \sum_{i=1}^{n} \frac{1}{i}$$

$$= n \cdot (\ln(n) + O(1))$$

$$\approx n \cdot \ln(n)$$

5-2.e) Average RT: $\frac{(n+1)}{2}$

Worst RT: n

Pg 143

5-2. a)

```
count = 0
S = array filled with zeros of size A.length
while count != A.length:
    i = randomIndex(A)
    if A[i] == x:
        return i
    if S[i] == 0:
        S[i] = 1
        count ++
return (x not in A)
```

5-2. b) $N :=$ random variable denoting the number of searches required

$$E[N] = \sum_{i \geq 1} i \cdot \Pr(i \text{ iterations required})$$

$$= \sum_{i \geq 1} i \cdot \left(1 - \frac{1}{n}\right)^{i-1} \cdot \left(\frac{1}{n}\right)^{1}$$

$$= \frac{1}{n} \cdot \frac{1}{\left(1 - \left(1 - \frac{1}{n}\right)\right)^2}$$

$$= \frac{1}{n} \cdot n^2$$

$$= \boxed{n}$$

Pg 204

8.4-4) To obtain $\Theta(n)$ performance, bucket-sort must be applied.

Let:

$$r_i = \sqrt{\frac{i}{n}}$$

$$c_i = \{(x, y) \mid r_{i-1} \leq x^2 + y^2 \leq r_i\},$$

for $i = 1, 2, \ldots, n$

The c_i regions partition the circle into n parts of equal area, being each a bucket. Since the points are uniformly distributed and each region has equal area, bucket-sort will run on $\Theta(n)$

```
BucketSort(A)
  n = A.length
  B = [0]*n
  for i=0 to n-1:
     B[i] = []            #Create lists
  for i=1 to n:
     B[j].append(A[i]), where
     (j-1)/n <= A[i].x** + A[i].y** < j/n
  for i=0 to n-1:
     sort(B[i])
  return B[0]+B[1]+...+B[n-1]
```