ECE374 Fall2020

Homework6

Name: Zhang Yichi 3180111309

Nov. 1^{st} 2020

5.2-1

I hire exactly one time only when the first candidate is the best candidate. Therefore, the probability is $p = \frac{1}{n}$. I hire exactly n times only when every candidate I interview is better than the previous one, i.e. the rank of the candidates is increasing. Therefore, the probability is $p = \frac{1}{n!}$

5.2-2

The first candidate is always hired. Thus, if I want to hire exactly twice, the first candidate cannot be the best one and the second candidate I hire must be the best one. Assume the candidate with rank i ($1 \leq i \leq n-1$) is the first candidate, $p_i = \frac{1}{n}$. Then, the best candidate with rank n must appear before the candidates with rank $i+1, i+2, \cdots, n-1$, $p_n = \frac{1}{n-i}$. Thus, the probability of hiring exactly twice is $p = \sum_{i=1}^{n-1} p_i p_n = \sum_{i=1}^{n-1} \frac{1}{n}\frac{1}{n-i} = \frac{1}{n}\sum_{i=1}^{n-1}\frac{1}{i}$

7.1-1

	array
i=0, j=1	13 19 9 5 12 8 7 4 21 2 6 11
i=0, j=2	13 19 9 5 12 8 7 4 21 2 6 11
i=1, j=3	9 19 13 5 12 8 7 4 21 2 6 11
i=2, j=4	9 5 13 19 12 8 7 4 21 2 6 11
i=2, j=5	9 5 13 19 12 8 7 4 21 2 6 11
i=3, j=6	9 5 8 19 12 13 7 4 21 2 6 11
i=4, j=7	9 5 8 7 12 13 19 4 21 2 6 11
i=5, j=8	9 5 8 7 4 13 19 12 21 2 6 11
i=5, j=9	9 5 8 7 4 13 19 12 21 2 6 11
i=6, j=10	9 5 8 7 4 2 19 12 21 13 6 11
i=7, j=11	9 5 8 7 4 2 6 12 21 13 19 11
swap A[i+1] and A[r]	9 5 8 7 4 2 6 11 21 13 19 12

7.1-4

```
def partition(A,p,r):
    x = A[r]
    i = p-1
    for j in range(p,r):
        if A[j] >= x:
            i += 1
            A[i],A[j] = A[j],A[i]
    A[i+1],A[r] = A[r],A[i+1]
    return i+1
```

Change the line in partition from A[j] <= x to A[j] >= x.

7.2-1

For $T(n) = T(n-1) + \Theta(n)$, we first guess $T(n) = \Theta(n^2)$. Then, we have $c_1 n^2 \leq c_1 n^2 + (1 - c_1 n) = c_1(n-1)^2 + c_1 n \leq T(n) = T(n-1) + \Theta(n) \leq c_2(n-1)^2 + c_2 n = c_2 n^2 - (c_2 n - 1) \leq c_2 n^2$ when $1 - c_1 n > 0$ and $c_2 n - 1 > 0$. Therefore, $T(n) = \Theta(n^2)$.

7.2-2

The running time of QUICKSORT when all elements of array A have the same value is $\Theta(n^2)$ because in this case the partition will always return the last position, so $T(n) = T(n-1) + \Theta(n)$. As proved above, it is $\Theta(n^2)$.

7.1

a.

i	j	array
0	13	13 19 9 5 12 8 7 4 11 2 6 21
1	11	6 19 9 5 12 8 7 4 11 2 13 21
2	10	6 2 9 5 12 8 7 4 11 19 13 21
10	9	6 2 9 5 12 8 7 4 11 19 13 21

b. In the first loop, x=A[p] is between the i and j, so there must exists $m, n \in [p, r)$ s.t. $A[m] \leq x$ and $A[n] \geq x$ and after the loop, j=m, i=n. If m=n=p, then the loop terminates. Otherwise, after the first loop, $A[m] \geq x$ and $A[n] \leq x$. At every beginning of a while loop, n will be the bound of j since $n < j$ and $A[n] \leq x$, m will be the bound of i since $m > i$ and $A[m] \geq x$. Once i reaches m or j reaches n, i must be larger than j and the loop ought to terminate. Thus, after the first loop, i and j will never be out of m and n, therefore, p and r.

c. As described above, if m=n=p, this partition will return j=p. Otherwise, $j \in [m, n] \subseteq (p, r)$ so it will return $j \in (p, r)$. Therefore, when HOARE-PARTITION terminates, it returns a value j

such that $p \leq j < r$.

d. When i goes from i_1 to i_2, the elements in $A[i_1..i_2 - 1]$ are smaller than x. Also, when j goes from j_1 to j_2, the elements in $A[j_2 + 1..j_1]$ are larger than x. Therefore, when it terminates, the elements in $A[p..i - 1]$ are smaller than x and the elements in $A[j + 1..r]$ are larger than x. If i=j, then A[i]=A[j]=x, so every element of A[p..j] is less than or equal to every element of A[j+1..r]. If i>j, then $A[p..j] \subseteq A[p..i - 1]$, so every element of A[p..j] is less than or equal to every element of A[j+1..r].

e.

```
def Hoar_partition(A,p,r):
    x = A[p]
    i = p-1
    j = r+1
    while True:
        while True:
            j -= 1
            if A[j] <= x:
                break
        while True:
            i += 1
            if A[i] >= x:
                break
        if i < j:
            A[i],A[j] = A[j],A[i]
        else:
            return j

def quicksort(A,p,r):
    if p < r:
        q = Hoar_partition(A,p,r)
        quicksort(A,p,q)
        quicksort(A,q+1,r)
```

In the previous quicksort, we call quicksort twice as quicksort(A,p,q-1) and quicksort(A,q+1,r), because elements before q are all smaller than A[q]. However, using Hoar partition, elements before q are not always smaller than A[q]. Thus, the q-th element still need to be sorted so we call quicksort twice as quicksort(A,p,q) and quicksort(A,q+1,r).