CSE 321
Homework 3

ANSWERS

Q1)

I will solve this question using the master theorem.

Master Theorem

If $a \geq 1$ and $b > 1$ are constants and $f(n)$ is an asymptotically positive function, then t he time complexity of a recursive relation is given by $T(n) = aT(n/b) + f(n)$

where, $T(n)$ has the following asymptotic bounds:

1. If $f(n) = O(n^{\log_b a-\epsilon})$, then $T(n) = \Theta(n^{\log_b a})$.
2. If $f(n) = \Theta(n^{\log_b a})$, then $T(n) = \Theta(n^{\log_b a} * \log n)$.
3. If $f(n) = \Omega(n^{\log_b a+\epsilon})$, then $T(n) = \Theta(f(n))$.

$\epsilon > 0$ is a constant.

a) **$T(n)=27\,T(n/3) +n^2$**

$a = 27$; $b = 3$; $f(n) =n^2$ then

$\log_3 27 = 3$ and $n^3 >n^2$ then case 1 is valid here.

Result = **$T(n) = \Theta(n^{\log_3 27}) = \Theta(n^3)$**

b) **$T(n) = 9T(n/4)+n$**

$a = 9$; $b = 4$; $f(n) = n$ then

$\log_4 9= 1.58$ (approximately) and $n^{1.58} > n^1$ then case 1 is valid here.

Result = **$T(n) = \Theta(n^{1.58})$**

c) **$T(n) = 2T(n/4)+ \sqrt{n}$**

$a = 2$; $b = 4$; $f(n) = n^{1/2}$

$\log_4 2=1/2$ and $n^{1/2}==f(n)$ then case 2 is valid here.

Result = **$T(n) = \Theta(n^{1/2}\log n)$**

d) $T(n) = 2T(\sqrt{n})+1$

Lets suppose; $m = \log_2^n$ then $n = 2^m$
$T(2^m) = 2T(2^{m/2})+1$

Suppose $T(2^m) = S(m)$
$S(m) = 2S(m/2)+1$ then use master theorem
$a = 2$; $b = 2$; $f(m) = 1$ (constant)
$\log_2^2=1$ and $n^1>f(m)$ (because $f(m)$ constant) then case 1 is valid here.

$S(m)= \Theta(m^{\log_2 2})= \Theta(m)$
but $S(m) = T(2^m)$ and $m = \log_2^n$ so;
$T(2^m) = \Theta(m)$
Then putting values and;

Result = **$T(n) = \Theta(\log_2^n)$**

e) $T(n) = 2T(n-2)$, $T(0)=1$, $T(1)=1$

$T(n)=2T(n-2)$
$T(n)=2[2T(n-4)] = 2^2T(n-2-2)$
$T(n)=2^3T(n-2-2-2)$
$T(n) = 2^kT(n-2k)$ Assume $n-2k = 0$ then $n=2k$ and $k = n/2$
$T(n)=2^{n/2}T(0)$ => $T(0) = 1$ and $T(n) =2^{n/2}$

Result = **$T(n) = \Theta(2^{n/2})$**

f) $T(n)=4T(n/2)+n$, $T(1)=1$

$a = 4$; $b = 2$; $f(n) = n$

$\log_2^4 = 2$ and $n^2 > n^1$ then case 1 is valid here.

Result = **$T(n) = \Theta(n^2)$**

g) $T(n)= 2\,T(\sqrt[3]{n})+1$, $T(3)=1$;

Lets suppose; $m = \log_3^n$ then $n = 3^m$
$T(3^m) = 2T(3^{m/3})+1$

Suppose $T(3^m) = S(m)$
$S(m) = 2S(m/3)+1$ then use master theorem
$a = 2$; $b = 3$; $f(m) = 1$ (constant)
$\log_3^2=0.63$ and $n^{0.63}>f(n)$ (because $f(n)$ constant) then case 1 is valid here.

$S(m)= \Theta (m^{\log_3 2})= \Theta (m^{0.63})$
but $S(m) = T(3^m)$ and $m = \log_3{}^n$ so;
$T(3^m) = \Theta (m^{0.63})$
Then putting values and;

Result = **$T(n) = \Theta ((\log_3{}^n)^{0.63})$**

Q2)

When;

n=1; print count = (1-0) =1

n=2; print count = (2-1).T(1) = 1

n=4; print count = (4-1)T(2) = 3

n=8; print count = (8-1)T(4)=21

n=16; print count = (16-1)T(8)=315

n=32; print count =(32-1)T(16)= 9765

.

.

.

n = n print count (n-1)T(n/2) so;

T(n) = (n-1)T(n/2)

Using Backward substitution method;

T(n) = T(n/2)(n-1)

T(n/2) = T(n/4)(n/2 -1)

T(n/4) = T(n/8)(n/4 -1)

T(n/8) = T(n/16)(n/8 -1)

Then we can say ;

T(n) = T(n/2)(n-1)+1

T(n) = T(n/4)(n/2 -1)(n-1)

T(n) = T(n/8)(n/4 -1)(n/2 -1)(n-1)

.

.

.

T(n) = T(n/n)(n/(n/2) -1).... (n/8 -1)(n/4 -1)(n/2-1)(n-1)

T(n/n) = T(1) =1 (base case)

T(n) = T(1) $\prod_{x=0}^{\log n/2} \frac{n}{2^x} - 1$ we can ignore constans;

Result = **T(n) =O($\frac{n^{\log n}}{2^{\log^2 n}}$)** (logs are base 2)

Q3) 3 Recursive calls of size 2n/3.

The base case if statement takes constant time (1).Then recurrence relation;

T(n)= 3T(2n/3) + 1

According to the master theorem;

a = 3; b = 3/2; f(n) = 1

$\log_{3/2}{}^{3}$ = 2.7 and $n^{2.7}$ > f(n) (because f(n) constant) then case 1 is valid here.

Result = **T(n) = Θ($n^{2.7}$)**

Q4)

Insertion Sort Analysis ;

Let's assume all the permutations are equally likely.

Let's restate that (i; j) is an inversion if ai < aj and j > i. With a swap you only

change one inversion. Swapping adjacent elements changes the total inversion by one.

Number of inversion in insertion sort is equal to the swap count. Probability for any i

and j being an inversion is 1/2 in every permutation. So:

E[$I_{i,j}$] = ($P_{i,j}$ = 1) = 1/2

E[I] =$\sum_{i<j} E$[$I_{i,j}$] = $\sum_{i<j} 1/2$

Since all possible pairs is $\binom{n}{2}$ we have ;

E[I] = $\sum_{i<j} 1/2 = \frac{1}{2}\binom{n}{2}$

$I_{avg} = \frac{1}{2}\binom{n}{2} = \theta(n^2)$

Quick Sort Analysis ;

Let's start by how the Hoare's partition works. Even if the swap indices coincide the swap number will be n + 1 since it checks if right > left not right >= left.

Partition(N) = n+1

Since the pivot is selected randomly we can say it has 1/N probability of being anywhere then multiply this by all the possibilities where the pivot can be so:

C_N = Partition(N) +($\sum_{1\le k\le N} C_{k-1} + C_{N-k})\frac{1}{N}$

Note that since the indices will be symmetric:

$\sum_{1\le k\le N} C_{k-1} + C_{N-k} = 2\sum_{1\le k\le N} C_{k-1}$

Then ;

C_N=N+1+(2 $\sum_{1\le k\le N} C_{k-1})\frac{1}{N}$

C_N=N+1+((2 $\sum_{1\le k\le N-1} C_{k-1}$)+2$C_{N-1}$)$\frac{1}{N}$

C_{N-1}=N+$\frac{1}{N-1}$(2 $\sum_{1\le k\le N-1} C_{k-1}$)

$\sum_{1\le k\le N-1} C_{k-1}$ =((C_{N-1}-N)(N-1))/2

C_N=N+1+((C_{N-1}-N)(N-1)+2C_{N-1}) $\frac{1}{N}$

C_N=N+1+(C_{N-1}N-C_{N-1}-N^2+N+2C_{N-1}) $\frac{1}{N}$

C_N=N+1+(C_{N-1}N-N^2+N+C_{N-1}) $\frac{1}{N}$

We multiply both parts with N.

C_N=N^2+N+C_{N-1}N-N^2+N+C_{N-1}

C_N=(N+1)C_{N-1}+2N

When we re-arrange the terms.

$$\frac{c_N}{N+1} = \frac{c_{N-1}}{N} + \frac{2}{N+1}$$

$$\frac{c_N}{N+1} = \frac{c_{N-2}}{N-1} + \frac{2}{N} + \frac{2}{N+1}$$

.

.

.

$$\frac{c_N}{N+1} = \frac{c_2}{3} + \sum_{3\le k\le N-1} \frac{2}{k+1}$$

By integration technique;

$$C_2/3 + \int_2^{N-1} \frac{2}{x+1}dx \ge \frac{c_N}{N+1} \ge C2/3 + \int_4^{N} \frac{2}{x+1}dx$$

$C_2/3+2\ln N-2\ln 3 \geq \frac{C_N}{N+1} \geq C_2/3 + 2\ln(N+1) - 2\ln 5$

Let c_1, c_2 be constants.

$2N\ln N + c_1N + 2\ln N + c_1 \geq C_N \geq 2N\ln(N + 1) + c_2N + 2\ln(N + 1) + c_2$:

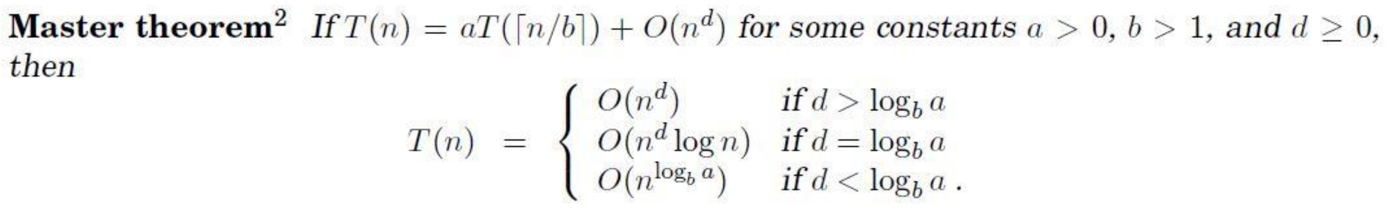
Therefore for N>10 you can see that this is true:

$6N\ln N \geq C_N \geq N\ln N$:

Therefore it is obvious that $C_n \in \theta(n \log n)$.

Insertion sort has n^2 average time and Quick sort has nlogn average time complexity theoretically Quick sort is better when we look at the python code for counting swaps for the same arrays with length of 10,20,30,40,50....190,200 the swaps count of quick sort less than insertion sort.Both by theoretical and experimental results Quick sort is better.

Q5)



Master theorem[2] *If* $T(n) = aT(\lceil n/b\rceil) + O(n^d)$ *for some constants* $a > 0$, $b > 1$, *and* $d \geq 0$, *then*

$$T(n) = \begin{cases} O(n^d) & \text{if } d > \log_b a \\ O(n^d \log n) & \text{if } d = \log_b a \\ O(n^{\log_b a}) & \text{if } d < \log_b a\,. \end{cases}$$

a) **$T(n) = 5T(n/3)+O(n^2)$**

Using master theorem;

a = 5; b = 3; d = 2;

$2 > \log_3{}^5$ then ;

Result = **$T(n) = O(n^2)$**

b) **$T(n) = 2T(n/2)+O(n^2)$**

Using master theorem;

a = 2; b = 2; d = 2;

$2 > \log_2{}^2$ then ;

Result = **$T(n) = O(n^2)$**

c) **$T(n) = T(n-1)+O(n)$**

Using Backward substitution method;

$T(n) = T(n-1) + n$

$T(n-1) = T(n-2) + n-1$

$T(n-2) = T(n-3) + n-2$

We can say ;

$T(n) = T(n-2) + n-1 + n$

$T(n) = T(n-3) + n-2 + n-1 + n$

.

.

.

$T(n) = T(n-k) + n-k+1 + n-k+2 + ... + n-1 + n$

since $n-k = 1$ so $T(1) = 1$ (Assume initial condition)

$T(n) = T(1) + 2 + ... + n$

$T(n) = 1 + 2 + ... + n$

$T(n) = n(n-1)/2$

$T(n) = n^2/2 - n/2$

Result = **$T(n) = O(n^2)$**