**Assignment - Week 1 Day 3**

**Submitted by:**

**23. Sunil Thapa(612810)**

**14. Anish Maharjan (613446)**

**5. Kapil Dangol (613417)**

**Question 1**. Write an algorithm

beautiful(A, n)

Input : An integer array with n elements

such that the best case running time is equal to the worst case running time. Write the algorithm and give your analysis to justify your claim.

**Solution:** Selection Sort has the same best and worst case, that is O(n^2).

beautiful(A, n){

int min, temp;

for ( int i = 1 to n-1){

min = i;

for (int j= i+1 to n){

if (list[j] < list[min]) {

min = j'

}

}

if (indexMin != i) {

temp = list[min];

list[min] = list[i];

list[i] = temp;

}

}

}

**Question 2.** Order them based on their complexity.

2^n , 2^(2n), 2^(n + 1), 2^( 2^n ) (Note: ^ stands for exponent operation. Example: 2^n = 2^n )

**Solution:**

1. O(2^n)
2. O(2^(n + 1)),
3. O(2^(2n))
4. O(2^( 2^n ))

**Question 3.** Mention one algorithm you know for each of the time complexities listed.

O(1), O(log n), O(n), O(n log n), O(n^2 ), O(n^3 ), O(2^n )

Example. O(n log n) : Quicksort

**Solution:**

* O(1): Printing one element of an array.
* O(log n):Binary Search
* O(n): GCD with Euclidean Algorithm
* O(n log n):Merge Sort
* O(n^2 ): Bubble Sort
* O(n^3 ): T(*n*) = 3*n^3* + 2*n* + 7
* O(2^n ): recursive calculation of Fibonacci numbers

**Question 4.** Apply Master Theorem and determine the time complexity of

fib(n) shown in slide 48.

**Solution:**

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

= 2T(n-1) + c //from the approximation T(n-1) ~ T(n-2)

= 2\*(2T(n-2) + c) + c

= 4T(n-2) + 3c

= 8T(n-3) + 7c

= 2^k \* T(n - k) + (2^k - 1)\*c

Let's find the value of k for which: n - k = 0

k = n

T(n) = 2^n \* T(0) + (2^n - 1)\*c

= 2^n \* (1 + c) - c

i.e. T(n) ~ 2^n

So the time complexity of fib(n) is O(2^n)

**Question 5**

Solve the recurrence

T(1) = 1

T(n) = 2T(n/2) + c

without using the Master Theorem.

Hint: Assume n = 2^k.

Follow the steps I did just before lunch break. Only difference is I did for 32. You are doing it for any perfect power of 2.

**Solution:**

T(1) = 1

T(n) = 2T(n/2) + c

Assume n=2^k

Let k=7 then n=2^6 =64

T(2^6) =2T(2^5)+c (i)

2T(2^5) =2[2T(2^4)+c]=4T(2^4)+2c (ii)

4T(2^4) =4[2T(2^3)+c]=8T(2^3)+4c (iii)

8T(2^3) =8[2T(2^2)+c]=16T(2^3)+8c (iv)

16T(2^2) =16[2T(2)+c]=32T(2)+16c (v)

16T(2) =32[2T(1)+c]=64T(1)+32c (vi)

Adding all equation s from (i) to (vi)

T(2^6) =c+2c+4c+8c+16c+32c+64T(1)

=64T(1)+63c

=64+(64-1)c

=n+(n-1)c

= θ(n)

**Question 6.** Practice Master theorem. It is a very important result in Analysis of algorithms. There are many resources on the internet. Show three different examples covering three possible cases. Show your detailed work.

**Case a=b^k**

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

= 1\*T(n / (3/2) ) + n⁰

so a=1, b= (3/2) , k=0

b^k=(3/2)^0=1

As a=b^k, using the master theorem,

T(n) = θ(n^k logn)

=θ(n^0 logn)

=θ(logn)