Name: Gaurav Yadav

Roll number: 21075034

Dept: Computer Science and Engineering (B.tech)

Lab – Assignment – 4

Q1) Propose and implement an algorithm that adheres to the specified recurrence relation, and then use the established techniques to determine the algorithm's time complexity.

i) T(n,m)=T(n-1,m)+T(n,m-1)+O(1)

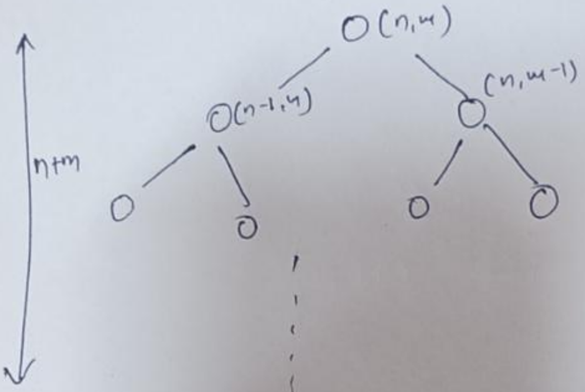
Ans->

```
#include<bits/stdc++.h>
using namespace std;
#define ll long long

ll solve(ll n,ll m){
    if(n==0) return 1;
    if(m==0) return 1;

    return solve(n-1,m)+solve(n,m-1);
}

int main(){
        ios::sync_with_stdio(0);
        cin.tie(0);
        cout.tie(0);
        ll n,m;
        cin>>n>>m;

        cout<<solve(n,m);


        return 0;
}
```

21075034
Gaurav Yadav

ⓐ $T(n,m) = T(n-1,m) + T(n,m-1) + O(1)$

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



Guess using recursion tree: $T(n,m) = O(2^{n+m})$

proof by induction: To prove: $T(n,m) \leq c \cdot 2^{n+m} - b$

Base case: we can easily see by recursion tree: $T(0,n) = T(n,0) = n$
$\leq c \cdot 2^{n} - b$

inductive step:

lets assume, the equation holds for $T(n,m-1)$, $T(n-1,m)$

$\Rightarrow T(n,m) \leq c \cdot 2^{n+m-1} + c \cdot 2^{n+m-1} - 2b + 1$

$\leq c \cdot 2^{n+m} - 2b + 1$

$\leq c \cdot 2^{n+m} - b$ for $b \geq 1$

hence, $T(n,m) = O(2^{n+m})$

21075034
Gaurav Yadav

ii) T(n)=3T(n/2)+O(n)

```cpp
#include<bits/stdc++.h>
using namespace std;
#define ll long long

ll solve(ll n){
    if(n==0) return 1;

    for(int i=0;i<n;i++){

    }

    return solve(n/2)+solve(n/2)+solve(n/2);
}

int main(){
        ios::sync_with_stdio(0);
        cin.tie(0);
        cout.tie(0);
        ll n;
        cin>>n;

        cout<<solve(n);


        return 0;
}
```

b) $T(n) = 3\,T(\frac{n}{2}) + O(n)$

from master's theorem

$T(n) = aT(\frac{n}{b}) + O(f(n))$

by comparing $a = 3, b = 2, f(n) = n$

$\log_b a > 1$

$\{\log_2 3\}$

$\therefore f(n) = n^{\log_b a - \epsilon}$, $\epsilon = \log_2 3 - 1 > 0$

$\Rightarrow T(n) = O(n^{\log_b a}) = O(n^{\log_2 3})$

$\approx O(n^{1.58})$

21075034
Gaurav Yadav

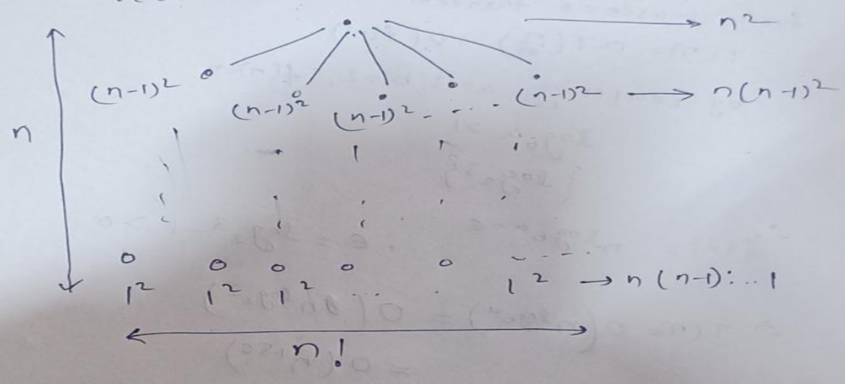
iii) **T(n)=nT(n-1)+n^2**

```
#include<bits/stdc++.h>
using namespace std;
#define ll long long

ll solve(ll n){
    if(n==0) return 1;
    for(int i=0;i<n;i++){
        for(int j=0;j<n;j++){

        }
    }

    ll x=0;
    for(int i=0;i<n;i++){
        x+=solve(n-1);
    }
    return x;
}

int main(){
     ios::sync_with_stdio(0);
     cin.tie(0);
     cout.tie(0);
     ll n;
     cin>>n;

     cout<<solve(n);


     return 0;
}
```

21075034
Gaurav Yadav

(c) $T(n) = nT(n-1) + n^2$

recursion tree



$T(n) = n^2 + n(n-1)^2 + \dots + n(n-1)\dots 1^2$

$= n! + (n^2 + n(n-1)^2 + \dots + 1^2)$

clearly $T(n) \geq n! \Rightarrow T(n) = \Omega(n!)$ (i)

Proving $T(n) = O(n!)$

$T(n) \leq cn! - 2n$

Base case: $T(4) \leq 24C - 8$ will be for some C

Inductive step:

assume $T(n-1) \leq c(n-1)! - 2(n-1)$

$T(n) \leq n[c(n-1)! - 2(n-1)] + n^2$

$\leq cn! + n^2 - 2n(n-1)$

$\leq cn! - 2n \quad \forall n \geq 4$

$\Rightarrow T(n) \leq cn! - 2 \leq cn! \quad \forall n \geq 4$ for some c

$\Rightarrow T(n) = O(n!)$ (ii)

using (i) & (ii) $T(n) = \Theta(n!)$

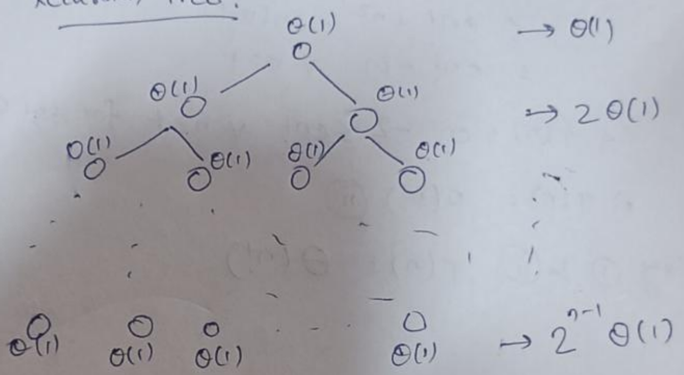
21075034

Gaurav Yadav

2) For the provided pseudo-code, define and solve the recurrence relation.

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

Recursion Tree:



$$T(n) = \Theta(1)\left[1 + 2 + \cdots + 2^{n-1}\right]$$

$$= (2^n - 1)\,\Theta(1)$$

$$= \Theta(2^n - 1)$$

$$\boxed{T(n) \approx \Theta(2^n)}$$