Assignment1: CS 610
Chanda Grover:17310004

Sol 1:

a) If N is a n bit number, then number of bits in N! $= \lfloor \log_2 N! \rfloor + 1 = \lfloor \log_2 (N^n) \rfloor + 1 = \lfloor n * \log_2 N \rfloor + 1$

b) Algorithm to count the number of bits in N!

```
//Factorial of N
        I]  Fact(N)
                {
        II}     If(N==1)
                        Return 1;
        III]    Else
        IV)             Return Fact(N*N-1)
                }
//Counting number of bits in N
        V]      count=0
        VI]  While(N)
                {
        VII]    N=>>1
        VIII]   Count=Count+1
                }
        IX]  Return Count
```

Analysis:
Fact(N) , time complexity is 0(N), since in every recursive function call N is decrementing by 1 and finally Factorial value is returned in N.
Next, Counting the number of bits takes 0(N') time complexity, where N' is the number of bits in factorial of N, since for every N, N' is shifted to right once.
Therefore over all time complexity is **0(N+N')** .

Sol 2.
Let e = (u,v) belongs to G(m,n), where m is the number of edges and n is the number of nodes of the graph. Let adjacent List data structure is used for graph representation. If we delete e from the graph G, and u and v are still present in the same connected component of graph , then definitely e belongs to the cycle of G. This can be checked by a DFS on this graph.

DFS(u):

1. Mark u as "Explored" and add u to R.
2. For each edge (u,v) incident to u
    a. If v is not marked "Explored" then
3. Recursively invoke DFS(u)
4. End If
5. End For

Analysis: Time taken by the above algorithm is **0(m+n)**

1. Time taken in deletion of e from the graph in the worst case is 0(n), since any vetex v can have at max n incident edges.
2. Next, the time taken by DFS is 0(m+n). Since at max we have n vertices in the graph. Therefore n lists are there and for every vertex there can be at max m edges. In the for loop we are calling DFS recursively each edge of vertex u which is equal to 0(2m), since every edge contributes to two vertices.

3. Therefore overall time complexity of the above algorithm is 0(m+n)+0(n)=0(m+n)

Sol 3 a)

Bound on the number of operations performed by the given algorithm is given by function f:

$0(n^3)$

i loop executes n times and for every value of i , j executes n-1 times, n-2times, n-3 times, …….. 2,1 times ans so on…..

Therefore number of operations=n{ (n-1) +(n-2) +(n-3) +…….3, 2, 1}

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

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

$<=\mathbf{0(n^3)}$

Sol 3 b)

Best case is also same here as worst case, since i loop executes n times and for every value of i , j executes n-1 times, n-2times, n-3 times, …….. 2,1 times ans so on…..

Therefore number of operations= n{(n-1) +(n-2) +(n-3) +…….3, 2, 1}

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

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

$=\mathbf{\Omega(n^3)}$

Sol 3 c)

Algo:

```
1.  initialize t=A[0]
2.  for(i=1,j=i+1; i<n , j<n; j++ )
3.          Add A[i] through A[j] and store in some temporary variable, say t.
4.          Store t  into B[i][j]
5.          If(j==n)
6.             Increase i by 1
7.             Update t with A[i]
8.             Set j=i
9.          EndIf
10.     EndFor
```

Analysis: This algo takes g(n)=$\mathbf{0(n^2)}$ **time**.

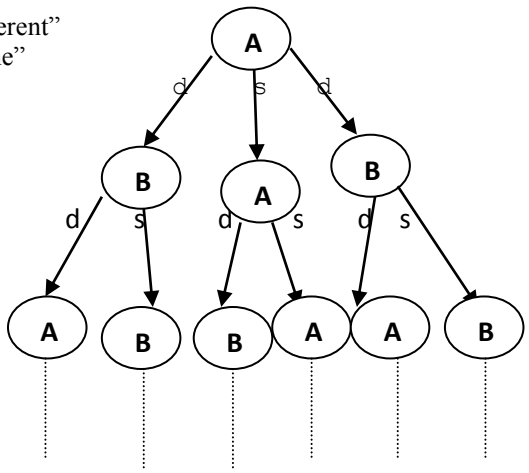
Number of operations:

i loop executes n times. Instead of computing A[i] through A[j] for every j value, this algorithm add only one more value to the temporary variable since it stores the results cumulatively. Therefore it performs n operations only.

$\lim_{n\to\infty} \frac{g(n)}{f(n)}$ = $\lim_{n\to\infty}(n^2/n^3)$= $lim_{n\to\infty}$ (1/n)= 0

Sol 4:

d= ”Different”
s= “Same”



Let A and B represents the two different classes of species. Nodes of the trees are represented by A and B. Edges are labeled as 's' or 'd', if they belong to same species or different species respectively. The problem of deciding whether m judgments are consistent or not can be determined by executing BFS on the given tree. Number of species= number of nodes=n and number of edges = number of judgments=m.

Algorithm:

1. Set discovered [s]=true and discovered[v]=false for all v belonging to species.
2. Initialize L[0] to consists of single element N.
3. Set the layer count i=0
4. Set the current BFS tree T=Ø
5. While L[i] is not empty
    a. Initialize an empty list L[i+1]
    b. For each node u belonging to L[i]
        i. Consider each edge (u,v) incident to u
        ii. If discovered[v] = false then
            1. Set discovered[v]=true
            2. Set spec[v]=A and label the edge as same
        iii. Else
            Set Spec[v]=B and label the edge as different

        End If.

    c. Add edge (u,v) to the tree T
    d. Add v to the list L[i+1]
    e. End For
    f. Increment the layer count I by 1.
6. End while.

Analysis: Time Complexity is simply O(m+n), since while loop executes for every L[i] and each L[i] can have maximum n nodes. Next, every node can have m edges, therefore sum of degree of all edges in the graph is 2m, since every edge contributes two vertices. After this we will scan all the edges of graph, and check if we have any edge having two vertices same or different. Hence we can decide whether the m judgments are consistent or not.

Sol 5:

Given Claim is false:

diam(G)=max{dist(v1,v2):v1,v2∈ V}

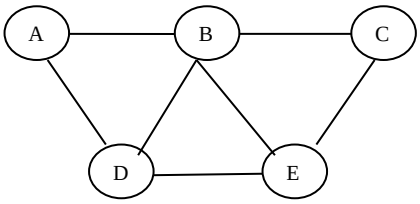
AvgPairDist(G), $\mu(G) = \frac{1}{\binom{n}{2}} \sum_{u,v \subset V} d(u,v).$ AvgParDist(G) is represented by $\mu(G)$, where d(u,v) is the distance between vertex u and v.

AvgPairDist(G) ≤ max{dist(v1,v2):v1,v2∈ V} = diam(G)

$$\frac{\text{diam}(G)}{\text{AvgPairDist}(G)} \geq \frac{\text{diam}(G)}{\text{diam}(G)} = 1$$

Therefore the claim is false.

For Example below is the one of the graph representation:



If we consider length of each edge as one, then roughly diameter of graph is 3.
And the average pairwise distance is (1+2+1+2+1+1+1+1+2+1)/10 = 13/10=1.3

Therefore apd(G)<=diam(G)

According to Plenik[4], he showed that apart from trivial inequality $\mu(G)$ <=diam(G) , no such relationship exists between diameter and average distance of a graph.

Few upper bounds were established independently:

- If G is a connected graph of order n, then $1 \leq (G) \leq (n + 1)/3$. Equality holds if and only if G is a complete graph or a path. [5, 6, 7]
- If G is a connected graph of n vertices and m edges, then $T(G) \geq 2 - 2mn(n-1)$. Equality holds if and only if diam(G) = 2

Sol 6:

Consider a directed graph G. We can check the consistency of given statements by verifying if G is acyclic or not. For each person P_i , let b_i and d_i denote the birth and deaths of a person P_i . (b_i , d_i) represents edge in the following two senses:

a) (d_i,b_j) edge is included when person i died before person j was born.
b) (b_i, d_j) and (d_i, b_j) edges are included when life spans of P_i and P_j are included at least partially.

Using DFS we can check if there is any cycle in the directed graph in 0(m+n) time. Cycle is present in any graph if any vertex has back edge to its parent or any ancestor of its parent. If we reach a vertex that is already in the recursion stack, then there is a cycle in the tree. Two cases happens:

Case [I]: If there is no cycle in the directed graph, it means it has topological ordering of the deaths and births of all people. That means this ordering is consistent with all the information that is given.

Case [II]: If there is cycle in the graph, it means events corresponding to the node precede the next, but then we cannot identify the first event that would happened, which is not consistent with the given information.

Collaborators:

Thanks to Anirban Dasgupta, G V Sumukha Bhardwaj, Chamancir Kaur, Aashish for helping me and discussing the problems.

References:

1] https://math.stackexchange.com

2] Algorithm Design by Jon Kleinberg and Eva Tardos

3] https://pdfs.semanticscholar.org/7b59/54df3f583ccf85b65c19d9aa7843cf1372a8.pdf : Wayne Goddard: Clemson University, Clemson SC USA,goddard@clemson.edu and Ortrud R. Oellermann University of Winnipeg, Winnipeg MN Canada, o.oellermann@uwinnipeg.ca

4] H. Wiener. Structural determination of paraffin boiling points.J. Amer. Chem. Soc. , 69(1):17–20, 1947

5] J.K. Doyle and J.E. Graver. Mean distance in a graph. Discrete Math. , 7(2):147– 154, 1977.

6] R.C. Entringer, D.E. Jackson, and D.A. Snyder. Distance in graphs. Czech. Math. J., 26:283–296, 1976.

7] L. Lov´asz. Combinatorial Problems and Exercise. Akad´emiai Kiad´o, Budapset, 1979.

8] http://www.geeksforgeeks.org/