Algorithms and Complexity Assignment 1

Student: Lixian Sun
Username: lixians
Student ID: 938295
Subject: COMP90038

Q1.

$f(n) = 4f\left(\frac{n}{2}\right) + g(n)$

$= 4(4f\left(\frac{n}{2}\right) + g(\frac{n}{2})) + g(n)$

$= 4(\dots\left(4f\left(\frac{n}{2^{\log_2 n}}\right) + g\left(\frac{n}{2^{\log_2 n-1}}\right)\right)\dots) + g(n)$

$= 4^{\log_2 n}f(1) + 4^{\log_2 n-1}g\left(\frac{n}{2^{\log_2 n-1}}\right) + \dots + 4^0 g(n)$

$f(n) \epsilon\, \theta(4^{\log_2 n}f(1) + \sum_{i=0}^{\log_2 n-1} 4^i g(\frac{n}{2^i}))$

$= \theta(n^2) + \theta(\sum_{i=0}^{\log_2 n-1} 4^i g(\frac{n}{2^i}))$

$= \theta(n^2) + \theta(n\sum_{i=0}^{\log_2 n-1} 2^i)$

$= \theta(n^2) + \theta(n^2 - n)$
$= \theta(n^2)$
The time complexity of the function $f(n)$ is $\theta(n^2)$.

Q2.
Function: Pick up (n-1) digits from the n digits array which can let the product of these (n-1) digits be the max, and then return the max product.
Input: The array A[.] which contains n digits.
Return: The max product of the (n-1) digits.

$function\ MAXPRODUCT(A[.],\ n)$
$\quad max \leftarrow -\infty$
$\quad min \leftarrow +\infty$
$\quad maxAnswer \leftarrow 1$ $\quad //create\ a\ variable\ to\ hold\ the\ value\ of\ max\ product$
$\quad for\ i \leftarrow 0\ to\ n$
$\qquad if\ A[i] \geq 0\ and\ A[i] < min$
$\qquad\quad min \leftarrow A[i]$
$\qquad\quad minPositive \leftarrow i$ $\quad //find\ the\ smallest\ number\ no\ smaller\ than\ 0$
$\qquad if\ A[i] < 0\ and\ A[i] > max$

$max \leftarrow A[i]$

$maxNegative \leftarrow i \quad //find\ the\ biggest\ number\ \ smaller\ than\ 0$

$for\ i \leftarrow 0\ to\ n$

$if\ i! = minPositive\ and\ i! = maxNegative$

$maxAnswer \leftarrow maxAnswer * A[i] \quad //the\ product\ of\ the\ other\ n-2\ digits$

$if\ maxAnswer \geq 0$

$maxAnswer \leftarrow maxAnswer * A[minPositive]$

$else$

$maxAnswer \leftarrow maxAnswer * A[maxNegative]$

$return\ maxAnswer$

Q3.

Function: Find the minimum number of cells that the contractor must pass through where the radiation level was above the acceptable threshold level, from position (0, 0) to position (n-1, n-1).

Input: The matrix A[.][.] which contains the information about each cell and size n.

Return: The minimum number of cells.

$function\ BestPath(A[.][.], n)$

$values\ in\ Value[.][.]\ are\ all\ \ 0 \quad //create\ the\ Value[.][.]\ of\ size\ n$

$Value[0][0] \leftarrow A[0][0] \quad //set\ the\ value\ of\ Value[0][0]\ because\ it's\ the\ start\ point$

$FindPath(1, n)$

$function\ FindPath(step, n)$

$if\ step \leq n-1 \quad //cells\ at\ the\ edges\ can\ only\ be\ reached\ from\ up\ or\ left\ cells$

$Value[step][0] \leftarrow A[step][0] + Value[step-1][0]$

$Value[0][step] \leftarrow A[0][step] + Value[0][step-1]$

$if\ step \geq 2\ and\ step \leq n-1$

$for\ i \leftarrow 1\ to\ step-1$

$j \leftarrow step - i$

$Value[i][j] \leftarrow A[i][j] + \min(Value[i-1][j], Value[i][j-1])$

$else\ if\ step > n-1\ and\ step \leq 2n-2$

$for\ i \leftarrow step - n + 1\ to\ n-1$

$j \leftarrow step - i$

$Value[i][j] \leftarrow A[i][j] + \min(Value[i-1][j], Value[i][j-1])$

$if\ step == 2n-2 \quad //reach\ the\ destination\ of\ A[n-1][n-1]$

$return\ Value[n-1][n-1]$

$FindPath(step+1, n)$

Q4.

(a)

Function: Return a list containing the neighborhood degree for each node v ∈ V.

Because the graph is represented using adjacency list, we suppose that for each v ∈ V, there is a list Adj[v] in adjacency list, and every vertex that is adjacent to v is stored in Adj[v].

Input: the graph, G, represented using an adjacency list.

Output: A list containing the neighborhood degree for each node.

$function\ NeighborhoodDegree(graph\ G)$

$\quad create\ degree[v] \leftarrow \{0\}$

$\quad create\ neighborhoodDegree[v] \leftarrow \{0\}$

$\quad for\ each\ vetex\ m\ \in V$

$\quad\quad count \leftarrow 0$

$\quad\quad for\ i \leftarrow 0\ to\ Adj[m].length$

$\quad\quad\quad count \leftarrow count + 1$

$\quad\quad degree[m] \leftarrow count$

$\quad for\ each\ vetex\ m\ \in V$

$\quad\quad for\ i \leftarrow 0\ to\ Adj[m].length$

$\quad\quad\quad n \leftarrow Adj[m].next$

$\quad\quad\quad neighborhoodDegree[m] \leftarrow neighborhoodDegree[m] + degree[n]$

$\quad return\ neighborhoodDegree[v]$

For each vertex in the graph G, we visit them and their adjacent twice, so the time complexity of the function is O(V+E).

(b)

If we use the adjacency matrix to represent the graph G, the Time complexity will become $O(|V^2|)$. That is because the function has to visit every cell in the matrix, even if there is no edge between the two nodes. In this case, we should use Adj[.][.] to represent the graph G. The changes to the function if we use the matrix are as follows:

$\quad [m, n] \leftarrow Adj[.][.].size$

$\quad for\ i \leftarrow 0\ to\ m$

$\quad\quad count \leftarrow 0$

$\quad\quad for\ j \leftarrow 0\ to\ n$

$\quad\quad\quad count \leftarrow count + 1$

$\quad\quad degree[i] \leftarrow count$

$\quad for\ i \leftarrow 0\ to\ m$

$\quad\quad for\ j \leftarrow 0\ to\ n$

$\quad\quad\quad neighborhoodDegree[i] \leftarrow neighborhoodDegree[i] + degree[j]$

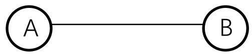
In the function, i equals to j equals to the number of vertex in the graph G, so the time complexity is $O(|V^2|)$.

Q5.

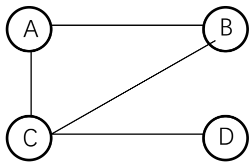
If there are $2n$ nodes in V, the function of connecting these $2n$ nodes with n^2 edges is as follows:

```
function ConstructGraph(V, n)
    mark each node in V with 0
    for i ← 1 to n do
        pick 2 nodes from V which are marked with 0
        mark this two nodes with 1
        chose a node from this two nodes randomly
        create edges between this node and all the other nodes in graph marked with 1
```

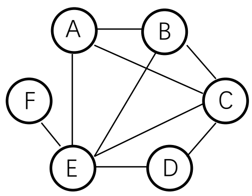
Graph Explanation:



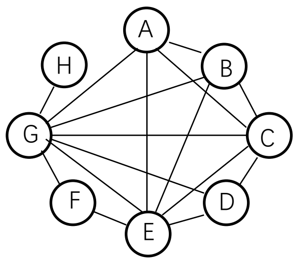
N=1, 2N=2, N^2=1



N=2, 2N=4, N^2=4



N=3, 2N=6, N^2=9



N=4, 2N=8, N^2=16

Mathematical Induction:

i. when n equals to 1, there are 2 nodes in V. Input V and n, there will be 1 edge in the graph which equals to n^2 and the graph has exactly one, unique, complete pairing.

ii. if the function is right when there are n nodes in V, there should be n^2 edges in the graph and the graph has exactly one, unique, complete pairing. When n turn in to $n+1$, according to the function, we should add 2 more nodes in to V, then we should mark them with 1, and then we should choose 1 nodes from this 2 nodes and connect this node with any other nodes in V which is marked with 1. So there will be $2(n+1)-1=2n+1$ edges to be added into the graph and the graph will has $n^2+2n+1=(n+1)^2$ edges. And the new added nodes can only pair with each other because the node which haven't been chosen only has one adjacent node, and the other nodes have already been paired, so the graph has exactly one, unique, complete pairing when there are $n+1$ nodes.

According to the two steps upon, the function is right.