IT300 Assignment 7

NAME: SUYASH CHINTAWAR
ROLL NO.: 191IT109
TOPIC: GREEDY ALGORITHMS - 2

Q1. Find the fewest number of steps required to transform word_i to word_j for any given input. Design and implement this as a shortest path problem using a greedy approach.

SOLUTION:

Code:

```
/*
This program computes the min no. of steps needed
to convert every word to every other word given in list
NOTE:
1) The matrix has rows and cols as per inputs words in order
2) Please do not give wrong inputs
*/
#include <bits/stdc++.h>
using namespace std;
#define INF 10000000

//Make adjacency list
void adjacency_list(int n, int k, int d, vector<string> wlist,
vector<int> adj_list[])
{
    for(int i=0;i<n;i++)
    {
        for(int j=i+1;j<n;j++)
        {
            int nd=0; //num of different letters
            for(int l=0;l<k;l++)
            {
                if(wlist[i][l]!=wlist[j][l]) nd++;
            }
            if(nd<=d)
            {
                adj_list[i].push_back(j);
                adj_list[j].push_back(i);
            }
        }
    }
}
```

```
//Perform BFS to get shortest path for one source and all
destinations (words)
void bfs(int src, vector<int> adj_list[], vector<int>& min_steps)
{
    queue<int> q;
    q.push(src);

    min_steps[src] = 0;
    while(!q.empty())
    {
        int x=q.front();
        q.pop();
        for(auto y:adj_list[x])
        {
            if(min_steps[y]>min_steps[x]+1)
            {
                min_steps[y]=min_steps[x]+1;
                q.push(y);
            }
        }
    }
}

int main()
{
    int n,k,d;
    cout<<"Enter n: ";
    cin>>n;
    cout<<"Enter k: ";
    cin>>k;
    cout<<"Enter d: ";
    cin>>d;

    vector<string> wlist(n);
    cout<<"Enter the words:\n";
    for(int i=0;i<n;i++) cin>>wlist[i];

    vector<int> adj_list[n];
    adjacency_list(n,k,d,wlist,adj_list);
```

```
    //Storing min_steps for each word
    vector<vector<int>> min_steps(n, vector<int>(n,INF));
    for(int i=0;i<n;i++)
    {
        bfs(i, adj_list, min_steps[i]);//bfs on each word
    }
    cout<<endl;

    cout<<"Minimum steps needed to convert every word to every other
word:\n";
    cout<<"NOTE: See code comments for matrix format\n";
    /*
    The rows and cols in the matrix will be
    the words given in input in the same order..
    for eg.
    matrix[i][j] says the min steps needed to convert word i to j
    */
    for(int i=0;i<n;i++)
    {
        for(int j=0;j<n;j++)
        {
            cout<<min_steps[i][j]<<" ";
        }
        cout<<endl;
    }
}
```

(continued...)

Output:

```
ubuntu@suyash-18-04:~/Desktop/sem5/it300/a7$ g++ q1.cpp
ubuntu@suyash-18-04:~/Desktop/sem5/it300/a7$ ./a.out
Enter n: 7
Enter k: 3
Enter d: 1
Enter the words:
hit
cog
hot
dot
dog
lot
log

Minimum steps needed to convert every word to every other word:
NOTE: See code comments for matrix format
0 4 1 2 3 2 3
4 0 3 2 1 2 1
1 3 0 1 2 1 2
2 2 1 0 1 1 2
3 1 2 1 0 2 1
2 2 1 1 2 0 1
3 1 2 2 1 1 0
ubuntu@suyash-18-04:~/Desktop/sem5/it300/a7$
```

Q2. As the last question of the successful interview, your boss gives you a piece of paper with numbers on it and asks you to compose a larger number from these numbers. The result is going to be your salary, so you are very much interested in maximising this number. Design and implement an algorithm using greedy technique to find the maximum salary from the randomly drawn n input numbers. The input can include single digit, double digit or many digits.

SOLUTION:

Code:

```
/*
This program takes in n numbers and
displays the largest number possible
NOTE:
1) Please do not give wrong inputs
*/
#include<bits/stdc++.h>
using namespace std;

//Using a custom comparator to sort in descending order
```

```
bool comparator(string n1, string n2)
{
   string n12 = n1 + n2;
   string n21 = n2 + n1;

   //after appending both strings check which is greater
   if(n12>n21) return true;
   return false;
}

int main()
{
   //no. of numbers
   int n;
   cout<<"Enter no. of numbers: ";
   cin>>n;

   //all numbers are taken in as strings
   vector<string> nums(n);
   cout<<"Enter the "<<n<<" numbers:\n";
   for(int i=0;i<n;i++) cin>>nums[i];

   //sort the array in descending order
   sort(nums.begin(),nums.end(),comparator);

   //append all the sorted string to form our result
   string res = "";
   for(int i=0;i<n;i++) res+=nums[i];
   cout<<"Maximum salary: "<<res<<endl;
}
```

Output:

```
ubuntu@suyash-18-04:~/Desktop/sem5/it300/a7$ g++ q2.cpp
ubuntu@suyash-18-04:~/Desktop/sem5/it300/a7$ ./a.out
Enter no. of numbers: 2
Enter the 2 numbers:
2 21
Maximum salary: 221
ubuntu@suyash-18-04:~/Desktop/sem5/it300/a7$ ./a.out
Enter no. of numbers: 3
Enter the 3 numbers:
1 10 9
Maximum salary: 9110
ubuntu@suyash-18-04:~/Desktop/sem5/it300/a7$ ./a.out
Enter no. of numbers: 3
Enter the 3 numbers:
23 39 92
Maximum salary: 923923
ubuntu@suyash-18-04:~/Desktop/sem5/it300/a7$
```

Q3. You are going to travel to another city that is located d miles away from your home city. Your car can travel at most m miles on a full tank and you start with a full tank. Along your way, there are gas stations at distances stop1, stop2, . . . , stop$n$ from your home city. What is the minimum number of refills needed? Design and implement an algorithm using a greedy approach.

SOLUTION:

Code:

```cpp
/*
This program takes gives the minimum no. of
refills needed to reach destination
NOTE:
1) Please do not give wrong inputs
*/
#include<bits/stdc++.h>
using namespace std;

//Function to find minimum refills needed
int min_refills(int d,int m, int n, vector<int> stops)
{
    //no. of refills
    int cnt = 0;

    //max distance that can be travelled currently
```

```
    int max=m;

    //index to represent current stop
    int i=0;

    while(max<d)
    {
        //no stops left or next stop unreachable
        if(i>=n or stops[i]>max)
        {
            return -1;
        }

        //Refilling at the farthest stop possible
        while(i<n-1 and stops[i+1]<=max)
        {
            i++;
        }
        max = stops[i]+m;
        i++;
        cnt++;
    }
    return cnt;
}

int main()
{
    //Take input
    int d,m,n;
    cout<<"Enter d: ";
    cin>>d;
    cout<<"Enter m: ";
    cin>>m;
    cout<<"Enter n: ";
    cin>>n;

    //storing stops(or gas stations)
    vector<int> stops(n+1);
    cout<<"Enter stops:\n";
```

```
    for(int i=0;i<n;i++) cin>>stops[i];

    //Counting minimum refills needed
    cout<<"Minimum refills needed: "<<min_refills(d,m,n,stops)<<endl;
}
```

Output:

```
ubuntu@suyash-18-04:~/Desktop/sem5/it300/a7$ g++ q3.cpp
ubuntu@suyash-18-04:~/Desktop/sem5/it300/a7$ ./a.out
Enter d: 950
Enter m: 400
Enter n: 4
Enter stops:
200 375 550 750
Minimum refills needed: 2
ubuntu@suyash-18-04:~/Desktop/sem5/it300/a7$ ./a.out
Enter d: 10
Enter m: 3
Enter n: 3
Enter stops:
1 5 9
Minimum refills needed: -1
ubuntu@suyash-18-04:~/Desktop/sem5/it300/a7$
```

THANK YOU