Lab Assignment 6

Prasanna Natarajan
1410110298

Problem Statement:

1. Suppose you were to drive from point A to point B. Your gas tank with a capacity C, when full, holds enough gas to travel m miles. You have a precise map that gives distances between gas stations along the route. Let d1 < d2 < …. < dn be the locations of all the gas stations along the route where di is the distance from point A to the gas station. You can assume that the distance between neighboring gas stations is at most m miles.
2. In the case that the rate at which you can fill your tank at a gas station is r (in liters/minute), so if you stop to fill your tank from 2 liters to 8 liters, you would have to stop for 6/r minutes. Give the most efficient greedy solution, where you need to minimize the total time you stop for gas filling?

Input: dist - Distance between A and B, m - the distance that can be crossed in full tank, c - capacity of gas that the car can hold, n - number of gas stations in between A and B, rate - rate at which the gas can be filled.

Output: A set of gas stations generated and all the relevant stations that the car should stop displayed.

Algorithm:

For finding the gas stations that are to be visited to minimize time:

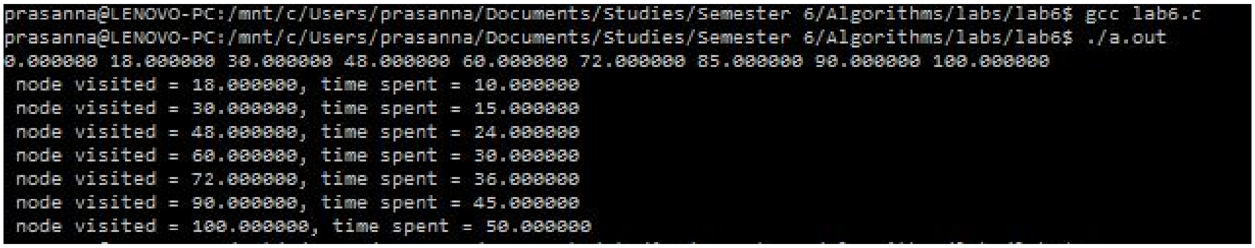
```
DistanceThatCanBeTravelled := m;
prev := 0
currentFuel := 10;
for i:=1 to n:
     while i<n and DistanceThatCanBeTravelled>a[i]: i++;
     i--;
    curDist := a[i];
    fuelReq := (curDist - prev) * (c/m)
    If fuelReq>currentFuel:
        time+:=fuelReq - currentFuel;
        currentFuel  = fuelReq
    currentFuel -:= fuelReq
    DistanceThatCanBeTravelled  := curDist+m;
    prev := a[i];
```

Code:

```c
/*
Name        :   Prasanna Natarajan
Roll Number :   1410110298
Inputs      :   dist - Distance between A and B,
                m - the distance that can be crossed in full tank,
                c - capacity of gas that the car can hold,
                n - number of gas stations in between A and B,
                rate - rate at which the gas can be filled.
Outputs     :   A set of gas stations generated and all the relevant stations that the car
should stop displayed.
*/

#include<stdio.h>
#include<stdlib.h>
#include<time.h>

// defining all the inputs
#define dist 100
#define m 20
#define c 10.0
#define n 9
#define rate 1

// function declarations
int * generate();

int main(){

    int *b = generate();
    double a[n] = {0,18,30,48,60,72,85,90,100};
    int i=0;
    for(i=0;i<n;i++){
        printf("%lf ",a[i]);
    }

    // Algorithm to find out the optimum gas stations

    double curDist = 0;
    double DistTravel = m;
    double prev = 0;
    double time = 10;
    double freq = 0;
    double curFuel = 10;
    for(i=1;i<n;i++){                  // outer for loop to travel through all gas stations
        while(i<n && DistTravel>a[i]){ // loop to find out if ith gas station can be skipped
        i++;
        }
        i--;
        curDist = a[i];                 // Travelling to the ith gas station
        freq = (curDist - prev) *(double)(c/m); // fuel required to go till the farthest gas
station within m
        //printf("freq = %lf cur fuel = %lf \n",freq , curFuel);
        if(freq > curFuel){
            time += (freq - curFuel)/r;
            curFuel = freq ;
        }
        curFuel -= freq;
        DistTravel = curDist+m;
        prev = a[i];
        printf("\n node visited = %lf, time spent = %lf",a[i],time);
    }
    printf("\n");


}

double * generate(){
    time_t t;
    srand((unsigned)time(&t));
    double*output = (double)malloc(sizeof(double)*n);
       double* arr = (double)malloc(sizeof(double)*n);
       double equalDiv = dist/(n);
       int i, node1, node2;
       int numRand = 10;
       int temp;

       for(i=0;i<num;i++){
               arr[i] = equalDiv;
       }

       for(i=0;i<((int)D-num*((int)D/num));i++){
               arr[i]++;
       }
       while(numRand!=0){
               node1 = rand()%n;
               node2 = rand()%n;
               while(node1 == node2)
                      node2 = rand()%n;
               temp = rand()%arr[node1];
               if(arr[node2]+temp <= m){
                      arr[node1]-=temp;
                      arr[node2]+=temp;
                      numRand--;
               }
       }
       double sum = 0;
       output[0] = 0;
       for(i=0;i<num;i++)
```

```
        {
                sum+=arr[i];
                output[i+1]+=sum;
        }
        return output;
}

void recurse(int *a,int DistTravel){

    static int curDist = 0,index = 1,time = 0;
    if(index>=n){
    exit(0);
    return;
    }
    if(DistTravel < a[index+1]){
        time += c - (-a[index] + DistTravel)*c/m;
        printf("\nRefilling in %d time = %d",a[index],time);
        index++;
        curDist = a[index-1];
        recurse(a,curDist+m/c*10);
        return;
    }
    else{
        time+=(c-((c/m)*(a[index+1] - a[index])))/rate;

        printf("\nGoing to %d time = %d",a[index],time);
        index++;
        recurse(a,(c/m*(a[index]-curDist))*c/m);
        return;

    }

    return;

}
```

Screenshots of solution:



```
prasanna@LENOVO-PC:/mnt/c/Users/prasanna/Documents/Studies/Semester 6/Algorithms/labs/lab6$ gcc lab6.c
prasanna@LENOVO-PC:/mnt/c/Users/prasanna/Documents/Studies/Semester 6/Algorithms/labs/lab6$ ./a.out
0.000000 18.000000 30.000000 48.000000 60.000000 72.000000 85.000000 90.000000 100.000000
 node visited = 18.000000, time spent = 10.000000
 node visited = 30.000000, time spent = 15.000000
 node visited = 48.000000, time spent = 24.000000
 node visited = 60.000000, time spent = 30.000000
 node visited = 72.000000, time spent = 36.000000
 node visited = 90.000000, time spent = 45.000000
 node visited = 100.000000, time spent = 50.000000
```