# Design and Analysis of Alogrithm

Zaeem Yousaf

To: Maryam Bashir (HomeWork # 7)

## Contents

# 1 Q: 1

## 1.1 Time complexity

$O(nlg(n))$

## 1.2 pseudocode

```
int maximizePoints(int points[], int time[], int maxTime){
  pointsPerUnit[points.length] = 0;
  for(int i=0; i< p.length; i++){
    pointsPerUnit[i] = p[i] / t[i]
  }

  //perform sort on points and respective changes will reflect in t
  int p = 0;
  t = 0;
  i = 0;
  while(i < no_t && t <= total_time){
    if(t+time[maxTime] < = maxTime){
      t = t+time[i];
      p = p + points[i];
    }
    i++;
  }
  return p;
}
```

# 2 Q: 2

## 2.1 time complexity

$O(n^2)$

## 2.2 Recurrence Relation

distance[i][j] = 1+ minimu(distance[i][j-1], distance[i-1][j], distance[i-1][j-1])

## 2.3 Pseudocode

```
distance[i][j] = 1+ minimu(distance[i][j-1], distance[i-1][j], distance[i-1][j-1])
```

```
  int makeString(string s1, string s2){
    for(int i=0; i< s1.length; i++){
      for(int j=0; j< s2.length; j++){
if(i == 0 || j == 0) d[i][j] = j;
else if (s1[i-1] == s2[j-1]){
  d[i-1][j-1]);
      }
      else{
d[i][j] = 1+ min(d[i][j-1],d[i-1][j],d[i-1][j-1];
      }
    }
      return d[m][n];
  }
```

# 3 Q: 4

## 3.1 part (a)

s[i][j] = minimum(c[i-1][j], 1+c[i][j-1]);

```
int substring(string x, string y){
  for(int i=0; i< x.length; i++){
    for(int j=0; j< y.length; j++){
      if(i==0) c[i][j] = j;
      else if(j == 0) c[i][j] = 1;
      else if(x[i-1] == y[j-1]) c[i][j] = 1+ c[i-1][j-1];
      else{
c[i][j] = minimum(c[i-1][j], 1+c[i][j-1]);
      }
    }
  }
}
```

# 4 Q: 5

## 4.1 using Dynamic programming

```
int maxDiscount(int F[], int D[], int n){
  int discount[n+1];
```

```
  discount[0] = 0;
  discount[1] = 0;
  int stores[n+1];
  store[0] = 0;
  if(d[0] < 0)d[0] = 0;
  else d[0] = 1;

  for(int i=2; i< n; i++){
    discount[i] = max(discount[i]-F[i-1]-1) + d[i-1],discount[i-1]);
  if(D[i] > D[i-1]
     store[i] = 1;
     else
       store[i] = 0;
     return (discount)

  }

}
```

# 5 Q: 6

## 5.1 part(a) is correct

### 5.1.1 proof

1. The problem is to find the smallest average
2. let there be 'n' number in set 'h' and 's'
3. there will be 'n' terms of differences let smallest height be $\mathrm{s_{height}}$ and smallest size be $\mathrm{s_s}$ smallest term be $\mathrm{s_t}$ smallest average $\mathrm{s_{avg}}$
4. $s_{avg} = \frac{s_{t1}+s_{t2}...s_{t_n}}{n}$
5. $\mathrm{s_t}$ is only when two numbers are very close to each other
6. $\mathrm{s_{avg}}$ is only when each term is the least one

Hence, this greedy algorithm will give the correct solution

## 5.2 part(b) incorrect

### 5.2.1 proof

let height = {87, 82, 80, 75} and size = {10,15,20,5}

1. smallest - smallest may not be the least term
2. e.g 75 - 10 = 65 and 75 - 20 = 55 which means that smalest-smallest != smalest difference
3. Hence, this is incorrect solution

# 6 Q: 7

## 6.1 Graph

Shortest Fair problem can be mapped on Graph. where **Vetices: airports** and **weighted Edges: fair** and therefore shortes distance between Vertex 'U' to Vertex 'V' will be the shortest fair. Dijkstra Algorithm can compute the shortest distance between from source to all Vertices.

## 6.2 Algorithm

```
function Dijkstra(Graph, source, destination):
      create vertex set Q
      for each vertex v in Graph:
          set last[v] to null
  set dist[v] to INFINITY

          add v to Q
      set dist[source] to 0

      while Q has any vertix:
         set u to vertex in Q with min dist[u]
          remove u from Q
          for each adjacent v of u:
              set comulative to dist[u] + weight(u, v)
              if comulative < dist[v]:
                  set dist[v] to comulative
                  set last[v] to u
return dist[destination]
```