COMP4500

Assignment 2 Report

Yangyang Xu S4344240

Question 1

(a)

Since we know the flags and their distances. There is a MyFlag class for constructing these flags. For ordering these flags by their distances, the priority queue will be used. There is a comparator (flagComp) of flag is used to determine and ensure the distance order of flags is from closest to farthest.

distance (final distance):

$$\sum_{i=1}^{n}\sum_{j=1}^{i} 2 * MyFlag_i distance + MyFlag_j\, distance$$

n: the total number of flags

old_distance: collect the distances of flags that have been fetched: $\sum_{j=1}^{i} distance(f_j)$

Code:

```
import java.util.*;

public class Main {
    public static void main(String[] args) {
        MyFlag flag1 = new MyFlag(1,5);
        MyFlag flag2 = new MyFlag(2, 3);
        MyFlag flag3 = new MyFlag(3, 11);
        PriorityQueue<MyFlag> flagList = new
PriorityQueue<>(flagComp);
        flagList.add(flag1);
        flagList.add(flag2);
        flagList.add(flag3);
        FetchFlag(flagList);
    }

    public static Comparator<MyFlag> flagComp = new
Comparator<MyFlag>(){
        @Override
        public int compare(MyFlag o1, MyFlag o2) {
            return (int) (o1.distance - o2.distance);
        }
    };


    public static void FetchFlag(PriorityQueue<MyFlag> flagList){
        int distance=0;
        List<Integer> old_distance = new ArrayList<>();
```

```
        while (!flagList.isEmpty()){
            for(int i = 0;i<old_distance.size();i++){
                distance += old_distance.get(i);
            }
            distance+= 2*flagList.peek().distance;
            old_distance.add(flagList.peek().distance);
            System.out.println("Flag "+flagList.poll().index);
        }
        System.out.println("Minimum Distance: "+ distance);
    }
}
class MyFlag{
    int index;
    int distance;
    public MyFlag(int index, int distance){
        this.index = index;
        this.distance = distance;
    }
}
```

(b)

It uses Greedy algorithm. To prove any optimal flag solution (S) to this problem, can be converted to a solution T, with the same final distance of n flags, and this is also the solution given by my algorithm of fetching the flag from the closest one to the farthest until all flags are fetched. My proof is by induction on the choice of flag. The d(n^{th} flag) denotes the distance of fetching the n^{th} flag, it includes the x units of n^{th} flag away from start line.

Base Case: If don't fetch any flag, the final distance will be 0;

Inductive Step: Make an assumption that both optimal solution and my algorithm have final distance d. When the optimal solution S, has the final distance: d + d(n+1^{th} flag) .The first flag fetched in optimal solution S must occur when the solution T also picked the first flag, and S has less or equal distance as T, because T only picks from closest flag. Replace the first choice of flag in S with the first choice of flag in T still gives the optimal solution, S', because

- S' has the same order of flag fetched as S,
- When the first flag with a distance d(1^{st} flag) can make it in T, then also can do it for the first flag to fetch in S', and

- When the first flag with the distance d(1st flag) and the second flag with the distance d(1st flag)+ d(2nd flag) can make it in S, the first choice of flag can make in S', which has less or equal distance than the S (when 1st flag fetched) and S' (when the 2nd flag fetched, same situation when 2nd flag also fetched in S).

Since there are n flags will be fetched eventually. When the 1st choice of flag is cancelled, there is an optimal solution (S'') to this problem, otherwise, a less final distance can be made to whole problem (**optimal solution of whole problem can have optimal solution for sub-problem**). For S'', it has n flag fetched with their related final distance, according to the induction hypothesis, my algorithm also is an optimal solution T'', which has same flags fetched and a final distance related to. If the choice of first flag fetched in S' and also in T is added to T'', there is a solution T'. To whole problem, T' has same number of flags fetched as optimal solution S', hence T' is also optimal. According to above analysation of T, T' has same solution as T. Therefore, T, my algorithm solution is also optimal.

Question 2

(b) There are always 9 unique different position combinations to start 3 actions, when Q1, Q2, Q3 have the front position [0,0,0]. Each unique position combination generates different next-step position according to if action can be applied, these next-step positions with minimum cost keep generating next-step position according to action, repeat until the front position becomes [X,Y,Z]. In the end, there are 9 action lists, and choose the one with minimum cost. T(X,0,0), T(0,Y,0), T(0,0,Z) can be used to do both Rejection and Sequential Pair(>= 1 people), If X,Y,Z are empty T(X=0,Y=0,Z=0), T(X,Y,Z) is constant θ(1). For worst case, T(X,Y,Z) = T(X-1,Y,Z)+T(X,Y-1,Z)+T(X,Y,Z-1)+T(X-2,Y,Z)+T(X,Y-2,Z)+T(X,Y,Z-2)+T(X-1,Y-1,Z)+T(X,Y-1,Z-1)+T(X-1,Y,Z-1)+θ(1) To get the lower bound, the new function will be smaller than current one. Since, each X, Y, Z will be done at least X-1, Y-1, Z-1, hence, T(X,Y,Z) = 3T(X-1,Y-1, Z-1)+θ(1) When X, Y, Z are not empty:

- ⇨ 3(3T(X-2,Y-2,Z-2)+θ(1))+θ(1), when X,Y, Z >=2
- ⇨ 3^2(3T(X-3,Y-3,Z-3)+θ(1))+3θ(1)+θ(1), when X,Y, Z >=3 ➔ Repeat …untill X,Y,Z >= k

$3^k T(X-k, Y-k, Z-k) + \sum_{a,b,c=0}^{k-1} C_{a,b,c}$ C: constant term for X, Y, Z.

=> X-k = Y-k = Z-k =0, X=Y=Z=k, $T(X,Y,Z) = 3^k T(0,0,0) + \sum_{a,b,c=0}^{k-1} C_{a,b,c} = \theta(3^k)$,Thus for lower bound the T(X,Y,Z) can be exponential.

(d)

There are 5 loops. And always 3 queues. The time complexity is:

$\sum_{a=0}^{X}\sum_{b=0}^{Y}\sum_{c=0}^{Z}\sum_{i=0}^{3}\sum_{j=0}^{i+1}\theta(C) = 9\sum_{a=0}^{X}\sum_{b=0}^{Y}\sum_{c=0}^{Z}\theta(C) = \theta(XYZ \bullet C)$.

space uses a 3D array which uses the X, Y, Z to be the size of each dimension of array, in the end, the space complexity is θ(XYZ•C)