Lab5 Q2 & Lab6 Q1

YAO ZHAO

Lab5 Question 2

- 1.Binary search from 1 to $\lfloor(Time\ range\ /\ N)\rfloor$
- 2.For each searched number t, check if all the tricks can be arranged t unit time.

Lab5 Question 2

Binary Search:

```
Left ← 1
Right ← ⌊(Time range / N)⌋
While (left ≤ right){
    mid ← (left + right)/2
    if (check mid if all the tricks can be arranged ){
        left←mid+1
    }else{
        right← mid-1
    }
}
Return right
```

Lab5 Question 2

How to check:

```
Sort the tricks by finish time so that b1<b2<b3<b4…<bn
For i = 1 to n{
      count←0
      For t= ai to bi{
          If flag[t] == false {
                flag[t] = true
                count ← count+1
                If (count == The target value)  break
           }
      }
      If count < The target value
            Return false
}
Return true
```

Lab5 Question 2 Problem Analysis

- Wrong choices of greedy template
- Try this simple test case:

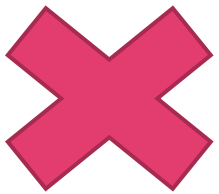
自测 #1

```
2
1 6
2 5
```

```
3
```

```
bool start_cmp(schedual a, schedual b)
{
    return a.start < b.start;
}

  sort(guy, guy + n, start_cmp);
```

```
Queue<time>h=new PriorityQueue<time>(new Comparator<time>() {
    public int compare(time a,time b) {
        return a.start-b.start;
    }
});
```

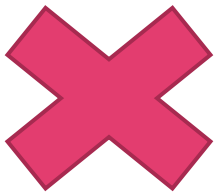


```
bool cmp(const Node& a,const Node& b){
        return a.en < b.en;
}

   sort(arr,arr+n,cmp);
```

Lab5 Question 2 Problem Analysis

- Correct greedy template but wrong time assignment strategy

```
static int bt,et,aans;
static time[] tl;
public static int doit(int min,int max){
    int ans = (min+max)/2;
    bt = tl[0].begin;
    et = tl[0].begin+ans;
    for (int i=1; i<tl.length; i++){
        if ((bt-tl[i].begin)>=ans){
            bt = bt-ans;
        }
        else
        if((tl[i].end-et)>=ans){
            et = et+ans;
        }
        else{
            if (min==max)
                return 0;
            return doit(min,ans);
        }
    }
    aans=ans;
    if (min==max)
        return 0;
    return doit(ans+1,max);

}
```



```
private static boolean judge(Pair[] intervals, int time) {
    boolean[] usedTime = new boolean[10010];
    for (Pair interval : intervals) {
        int obtainedTime = 0;
        for (int i = interval.x; i <= interval.y; i++) {
            if (!usedTime[i]) {
                usedTime[i] = true;
                if (++obtainedTime == time) {
                    break;
                }
            }
        }
        if (obtainedTime != time) {
            return false;
        }
    }
    return true;
}
```

Lab5 Question 2 Problem Analysis

- Is the following algorithm right or no?

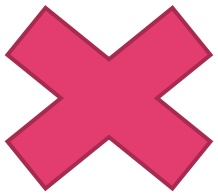
```
bool cmp(gf x, gf y)
{
    if (x.a != y.a)
        return x.a > y.a;
```

```
int cnt = 0;
for (int j = g[i].b; j >= g[i].a; --j)
    if (!h[j])
    {
        h[j] = 1;
        cnt++;
        if (cnt == x)
            break;
    }
if (cnt < x)
    return false;
```

Lab5 Question 2 Problem Analysis

- Wrong binary search implementation

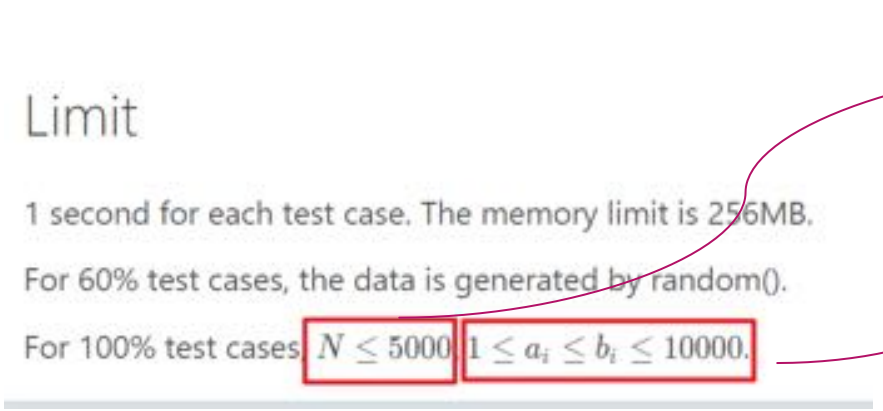
```
int left=0;
int right=t;
int mid=0;
while(left<=right){
    mid=(left+right)/2;
    if(check(mid))
        left=mid+1;
    else
        right=mid-1;
}
cout<<mid<<endl;
```



```
int left = 0, right = min_val, mid = 0;
while(left <= right){
    mid = (left + right) / 2;
    if(check(mid)){
        left = mid + 1;
    }else{
        right = mid - 1;
    }
}


printf("%d\n",right);
```

Lab5 Question 2 Problem Analysis

- Calculate the data range carefully to match your array size and search boundary.




```
#define maxn 5010

Friend friends[maxn];
```

```
int left = 0;
int right = 10010;
while (left < right){
    int mid = (left + right + 1) / 2;
    if (arrange(mid))
        left = mid;
    else
        right = mid - 1;
}
```

Case 1

Case 3

Case 2

Case 4

Lab6 Question 1

Dijkstra's Algorithm

For each unexplored node, explicitly maintain $\pi(v) = \min_{e=(u,v):\, u \in S} d(u) + \ell_e$

- Next node to explore = node with minimum $\pi(v)$.
- When exploring v, for each incident edge e = (v, w), update

$$\pi(w) = \min \{ \pi(w), \pi(v) + \ell_e \}.$$

Efficient implementation. Maintain a priority queue of unexplored nodes, prioritized by $\pi(v)$.

Lab6 Question 1

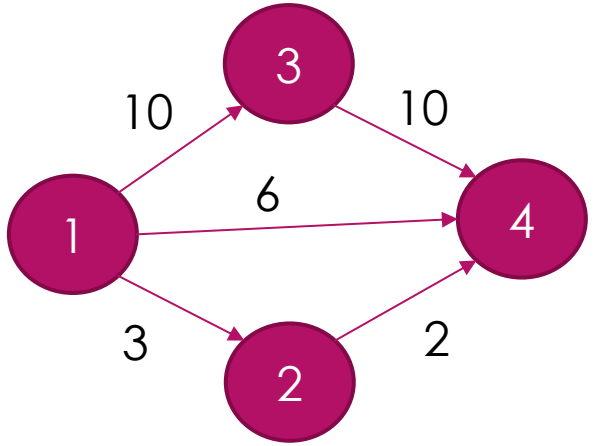
For the Question1, when you calculate the distance, use $\log(l_e)$ instead of l_e

Log(ab) = loga + logb

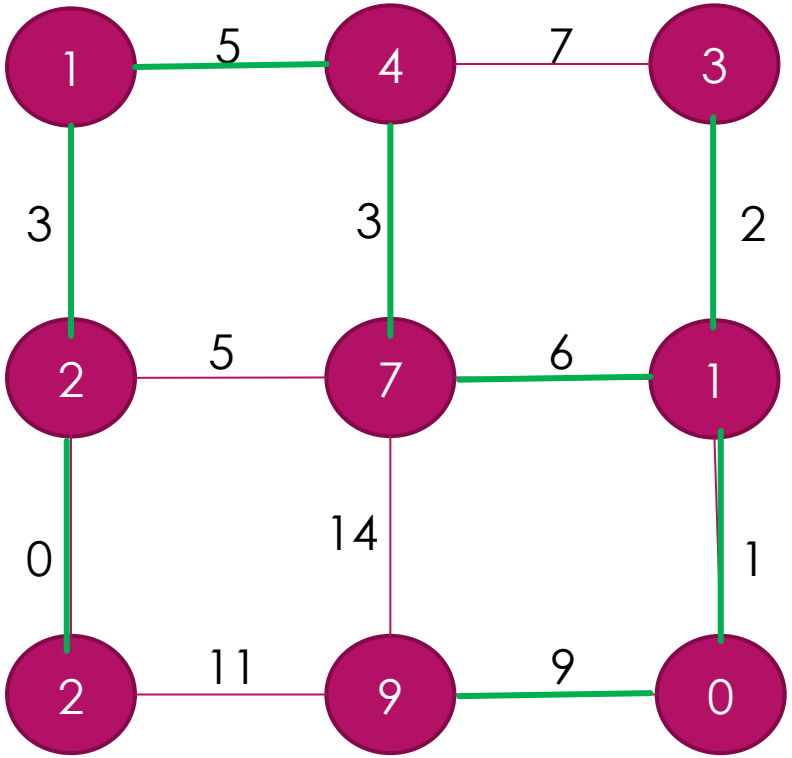
After finding the shortest path, you can calculate the answer using the original weight, remember mod 19260817.

Lab6 Question 2

How to calculate the actural distance?
Vertex A to B
Assume you spent x seconds to reach A, and the weight between A and B is w
The time interval of B is (a, b)
$diff = x+w \ \% \ (a+b)$
$If \ \ diff < a \qquad actural\ distance \mathrel{+}= a - diff$



Lab7 Question 1



0 1 2 3 3 5 6 9 → 29