



快快编程2360



状态

请同学定义状态

f[i]表示恰好跳到i号格时最小费用



| i= | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|-------|---|----------|---|---|---|---|---|---|---|---|----|
| x[i]= | 0 | 2 | 1 | 3 | 2 | 1 | 2 | 3 | 1 | 2 | 1 |
| f[i]= | 0 | ∞ | 1 | 3 | 2 | 2 | 3 | 5 | 3 | 4 | 4 |

$$f[10]=x[10]+min\{f[6],f[7],f[8]\}$$



算法1: 朴素DP 枚举决策j:之前位置在哪一格

状态转移的时间复杂度O(b-a+1)

算法2: 单调队列 维护滑动窗口 最小值 单调队列中只保留有望成为最值的编号 当前窗口最小值编号在单调队列左侧 状态转移的均摊时间复杂度**O(1)**



```
49 pint main(){
        freopen("dog.in","r",stdin);
50
        freopen("dog.out", "w", stdout);
51
52
        input();
                                      如何对拍?
53
        if(n*(b-a+1)<=1e7)
                                    保留第54,56行
54
            solveBF(); ←
                                   注释掉第53,55行
55
        else
56
            solve(); <</pre>
                                   正解没写完怎么办?
57
        return 0;
                                   注释掉53,55,56
58<sup>1</sup>}
                                       保留54
```

哪怕正解有错 其他部分可以得分

[L,R]是被i号依赖的格子编号

```
19 void solveBF(){
        f[0]=0;
20
        for(ll i=1;i<=n;++i){
21 †
22
            ll L=i-b-1;
                              请补充修改L
23
            ll R=i-a-1;
24
            f[i]=
            for(ll j=L;j<=R;++j)if(f[j]!=INF)</pre>
25
                 f[i]=min(f[i],f[j]+x[i]);
26
27
        if(f[n]==INF)cout<<-1<<endl;</pre>
28
        else cout<<f[n]<<endl;</pre>
29
30 ¹ }
```

```
单调队列优化决策
                         q[]储存有望成为最值的格子编号
31 p void solve(){
        f[0]=0;
32
        11 1=0, r=0;
33
        for(ll i=1;i<=n;++i){</pre>
34阜
35
            11 L=i-b-1; L=max(L,0LL);
36
            ll R=i-a-1;
            f[i]=INF;
37
            if(R<0)continue;</pre>
38
            while(1<r&&
                                )1++;
39
            while(1<r&&
40
            q[r++]=
41
            if(f[q[1]]==INF)continue;
42
            f[i]=
43
44
45
        if(f[n]==INF)cout<<-1<<endl;</pre>
        else cout<<f[n]<<endl;</pre>
46
47
```



快快编程2361



理解数据规模 请同学制定得分策略

对于10%的数据,a=b

对于30%的数据, n <= 10000

对于全部的数据, n <= 10^9, 1<=a<=b<=10, m<=100



```
53 pint main(){
        freopen("jump.in","r",stdin);
54
55
        freopen("jump.out","w",stdout);
56
        input();
57
        sort(pos+1,pos+1+m);
        if(a==b)
58
59
             solveEq();
        else if(n*(b-a+1)<=1e7)</pre>
60
             solveDP();
61
62
        else
63
             solveDPopt();
64
        return 0;
65<sup>1</sup>}
```

哪怕正解有错 其他部分可以得分



```
17 void input(){
        cin>>n>>a>>b>>m;
18
       for(ll i=1;i<=m;++i)cin>>pos[i];
19
20
21 void solveEq(){
22
        11 ans=0;
23
        for(ll i=1;i<=m;++i)
24
            if(pos[i]%a==0)++ans;
25
        cout<<ans<<endl;
26<sup>1</sup>}
```



```
1 /*姓名XXX

2 f[i]恰好跳到i号格时最小费用

3 n=10, a=2, b=3, m=5

4 i=0,1,2,3,4,5,6,7,8,9,10,11,5

x[i] 0 0 1 1 0 1 1 1 0 0 0 0 0

6 f[i]=0 - 1 1 1 2 2 2 7

*/
```

[L,R]是被i号依赖的格子编号



```
27 pvoid solveDP(){
28
        f[0]=0;
        for(ll i=1;i<=m;++i)x[pos[i]]=1;
29
        for(ll i=1;i<=n+a-1;++i){
30∮
            11 L=i-b; L=max(L,0LL);
31
32
            ll R=i-a;
33
            f[i]=INF;
            for(ll j=L;j<=R;++j)if(</pre>
34
                 f[i]=min(f[i],f[j]+x[i]);
35
36
37
        ll ans=INF;
38
39
        cout<<ans<<end1;</pre>
40
```



```
10 typedef long long ll;
11 const ll N=
12 const ll M=
13 const ll INF=1e9;
14 ll n,m,a,b;
15 ll f[N],x[N],pos[M];
```



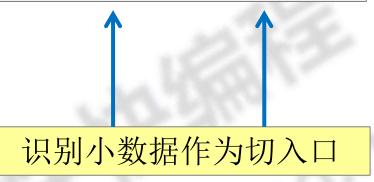
理解数据规模请同学思考满分算法

朴素的O(n) 算法不能满分

对于10%的数据,a=b 对于30%的数据,n <= 10000 对于全部的数据,n <= 10^9, 1<=a<=b<=10, m<=100

过路费收费点非常分散存在大量免费区域

跳跃步伐非常小





用小数据启发思路

$$a=1,b=2$$

| i= | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|-------|---|---|---|---|---|---|---|---|---|---|----|
| x[i]= | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| f[i]= | 0 | 1 | 1 | 2 | 1 | 1 | 1 | 1 | 1 | 2 | 1 |

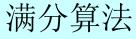
| 免费区域出现大量相同f[i] |
|----------------|
|----------------|

免费区域适当缩短不会影响结果

缩短到多少比较合适?

若相邻2个收费格间隔太远 可以适当缩短间隔

| i= | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|-------|---|---|---|---|---|---|---|---|
| x[i]= | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 |
| f[i]= | 0 | 1 | 1 | 2 | 1 | 1 | 2 | 1 |



坐标收缩



$$x[1]=5,x[2]=1000$$

$$a=9,b=10$$

{6,7,8,...,999}都免费

Sylvester定理

ab-a-b为71

8号一定可以免费到{80,81,...,999}中每一格

9,10,..,15号免费到{87,...,999}中每一格

$$x[1]=5,x[2]=96$$

[L,R]是被i号依赖的格子编号

sM[i]表示i号累积左移几格



```
41 pvoid solveDPopt(){
                              收缩到间隔200
                                             改成10000
42
        11 GAP=200;
                                             也不会超时
                              一定不影响结果
43
        pos[0]=0;
        for(ll i=1;i<=m;++i)</pre>
44
45
            if(pos[i]-pos[i-1]>GAP)
                 move[i]=pos[i]-(pos[i-1]+GAP);
46
        for(ll i=1;i<=m;++i)sM[i]=sM[i-1]+move[i];</pre>
47
        for(ll i=1;i<=m;++i)pos[i]-=sM[i];</pre>
48
49
        n≓
50
51<sup>1</sup>}
                              move[i]表示i号相对i-1号左移几格
```



快快编程820



理解数据规模 请同学制定得分策略

对于10%的数据,n≤10,m=1,0≤ti≤100

对于30%的数据,n≤20,m≤2,0≤ti≤100

对于50%的数据,n≤500,m≤100,0≤ti≤10⁴

另有20%的数据,n≤500,m≤10,0≤ti≤4×10⁶

对于100%的数据,n≤500,m≤100,0≤ti≤4×10⁶

m=1,2为得分点/突破口

ti=0为易错点

ti=4*10⁶为难点

部分分



It is the state of the state of

m=1



```
1 /*姓名XXX

2 f[T]恰好T时刻开出一辆车时此时最少共等待几分钟

3 n=5,m=5

4 T=0,1,2,3,4,5,6,7,8,9,10,11,12,13,14

5 人 ** ** *

6 f[T]=0 0 0 1 2 3 4 2 4 6 8

7 */
```

动态规划



$$n=5, m=5$$

| T= | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|-------|---|---|-----|---|---|---|---|---|---|---|----|----|----|----|------|
| 学生 | | | *** | | | | | | | | | | 43 | | \$\$ |
| f[T]= | 0 | 0 | 0 | 1 | 2 | 3 | 4 | 2 | 4 | 6 | 8 | 4 | 2 | 3 | 4 |

状态转移 决策

计算f[T]时要决策:前一次发车是哪个时刻p

枚举p的范围{0,1,2,...,T-m}

优化:减小枚举范围 枚举p的范围{T-2m,...,T-m}

若p<T-2m,一定可以在T-2m到T-m时刻间多发一次车不会使解答变差,只会变更好

动态规划



$$n=5, m=5$$

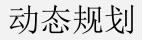
状态转移 决策

计算f[T]时要决策:前一次发车是哪个时刻p

费用:时刻{p+1,p+2,...,T}的所有人等待总时间

例如T=8,p=1,时刻{2,3,...,8}共3人

T*人数 - (到达时刻总和)





$$n=5, m=5$$

| T= [| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|-------|---|---|----------|---|--|---|---|---|---|---|----|----|--------------|----|----------|
| 学生 | | | % | | \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\ | | | | | | | | } | | % |
| f[T]= | 0 | 0 | 0 | 1 | 2 | 3 | 4 | 2 | 4 | 6 | 8 | 4 | 2 | 3 | 4 |

状态转移 决策

前缀和

计算f[T]时要决策:前一次发车是哪个时刻p

费用:时刻{p+1,p+2,...,T}的所有人等待总时间

需要知道 $\{p+1,p+2,\ldots,T\}$ 时刻共来了几人

cnt[x]表示时刻x共几人

sC[x]表示x时刻及之前共几人达到

sT[x]表示x时刻及之前达到所有人的到达时刻总和

```
21 proid solveDP(){
        for(ll i=1;i<=n;++i)cnt[t[i]]++;</pre>
22
        11 END=t[n]+m;
23
        for(ll T=1;T<=END;++T){</pre>
24₽
             sC[T]=sC[T-1]+cnt[T];
25
             sT[T]=sT[T-1]+cnt[T]*T;
26
27
        f[0]=0;
28
        for(ll T=1;T<=END;++T){</pre>
29 
             ll L=max(T-2*m,OLL);
30
             11 R=max(T-m, 0LL);
31
32
             f[T]=INF;
             for(11 p=L;p<=R;++p){</pre>
33 申
                  11 cost=T*(sC[T]-sC[p])-(sT[T]-sT[p]);
34
                  f[T]=min(f[T],f[p]+cost);
35
36
37
38
        11 ans=*min_element(f+t[n],f+END+1);
        cout<<ans<<endl;</pre>
39
```



```
50 pint main(){
        freopen("bus.in","r",stdin);
51
52
        freopen("bus.out","w",stdout);
53
        input();
54
        for(ll i=1;i<=n;++i)++t[i];
55
        sort(t+1,t+1+n);
        11 maxt=t[n];
56
57
        if(m==1)
58
            cout<<0<<endl;
        else if(m*maxt<=1e7)</pre>
59
60
            solveDP();
61
        else
62
            solveDPopt();
63
        return 0;
64
```

哪怕正解有错 其他部分可以得分

坐标收缩



间隔缩小到2m

状状划形形.net KKCoding.net

快快编程作业

2360

2361

820