

# C++算法



快快编程2360

快快编程  
kkcoding.net



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
---------------------------

快快编程  
kkcoding.net



$i=$	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	$\infty$	1	3	2	2	3	5	3	4	4

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



$i=$	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	$\infty$	1	3	2	2	3	5	3	4	4

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 int main(){
50     freopen("dog.in", "r", stdin);
51     freopen("dog.out", "w", stdout);
52     input();
53     if(n*(b-a+1)<=1e7)
54         solveBF(); ←
55     else
56         solve(); ←
57     return 0;
58 }
    
```

如何对拍?

保留第54,56行

注释掉第53,55行

正解没写完怎么办?

注释掉53,55,56

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(){
20     f[0]=0;
21     for(int i=1;i<=n;++i){
22         int L=i-b-1;
23         int R=i-a-1;
24         f[i]=
25         for(int j=L;j<=R;++j)if(f[j]!=INF)
26             f[i]=min(f[i],f[j]+x[i]);
27     }
28     if(f[n]==INF)cout<<-1<<endl;
29     else cout<<f[n]<<endl;
30 }
```

请补充修改L

```
31 void solve(){
32     f[0]=0;
33     ll l=0,r=0;
34     for(ll i=1;i<=n;++i){
35         ll L=i-b-1; L=max(L,0LL);
36         ll R=i-a-1;
37         f[i]=INF;
38         if(R<0)continue;
39         while(l<r&&[redacted])l++;
40         while(l<r&&[redacted])r--;
41         q[r++]=[redacted];
42         if(f[q[l]]==INF)continue;
43         f[i]=[redacted];
44     }
45     if(f[n]==INF)cout<<-1<<endl;
46     else cout<<f[n]<<endl;
47 }
```





快快编程2361

快快编程  
kkcoding.net



# 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， $n \leq 10000$

对于全部的数据， $n \leq 10^9$ ,  $1 \leq a \leq b \leq 10$ ,  $m \leq 100$

快快编程  
kkcoding.net



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

哪怕正解有错  
其他部分可以得分



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



```
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
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 void solveDP(){
28     f[0]=0;
29     for(ll i=1;i<=m;++i)x[pos[i]]=1;
30     for(ll i=1;i<=n+a-1;++i){
31         ll L=i-b;  L=max(L,0LL);
32         ll R=i-a;
33         f[i]=INF;
34         for(ll j=L;j<=R;++j)if( )
35             f[i]=min(f[i],f[j]+x[i]);
36     }
37     ll ans=INF;
38     ;
39     cout<<ans<<endl;
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,  $n \leq 10000$

对于全部的数据,  $n \leq 10^9, 1 \leq a \leq b \leq 10, m \leq 100$

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

识别小数据作为切入口







## 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



满分算法	坐标收缩
	$x[1]=5, x[2]=1000$
	$a=9, b=10$
	$\{6, 7, 8, \dots, 999\}$ 都免费
Sylvester定理	
$ab-a-b$ 为 71	
	6号一定可以免费到 $\{78, 79, \dots, 999\}$ 中每一格
	7号一定可以免费到 $\{79, 80, \dots, 999\}$ 中每一格
	8号一定可以免费到 $\{80, 81, \dots, 999\}$ 中每一格
	9, 10, ..., 15号免费到 $\{87, \dots, 999\}$ 中每一格
	$f[87]$ 取 $\min\{f[6], f[7], \dots, f[15]\}$
	$f[88], f[89], \dots$ 都取 $\min\{f[6], f[7], \dots, f[15]\}$
优化修改	$x[1]=5, x[2]=96$



[L,R]是被i号依赖的格子编号

```

41 void solveDPopt(){
42     ll GAP=200;
43     pos[0]=0;
44     for(ll i=1;i<=m;++i)
45         if(pos[i]-pos[i-1]>GAP)
46             move[i]=pos[i]-(pos[i-1]+GAP);
47     for(ll i=1;i<=m;++i) sM[i]=sM[i-1]+move[i];
48     for(ll i=1;i<=m;++i) pos[i]-=sM[i];
49     n=
50
51 }
    
```

收缩到间隔200 一定不影响结果	改成10000 也不会超时
---------------------	------------------

move[i]表示i号相对i-1号左移几格
sM[i]表示i号累积左移几格



# 快快编程820

快快编程  
kkcoding.net



# 理解数据规模

## 请同学制定得分策略

对于10%的数据,  $n \leq 10, m = 1, 0 \leq t_i \leq 100$

对于30%的数据,  $n \leq 20, m \leq 2, 0 \leq t_i \leq 100$

对于50%的数据,  $n \leq 500, m \leq 100, 0 \leq t_i \leq 10^4$

另有20%的数据,  $n \leq 500, m \leq 10, 0 \leq t_i \leq 4 \times 10^6$

对于100%的数据,  $n \leq 500, m \leq 100, 0 \leq t_i \leq 4 \times 10^6$

$m=1,2$ 为得分点/突破口

$t_i=0$ 为易错点

$t_i=4 \times 10^6$ 为难点

快快编程  
kkcoding.net



$m=1$








```
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					 										
$f[T]=$	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 $\{0, 1, 2, \dots, T-m\}$

优化：减小枚举范围

枚举 $p$ 的范围 $\{T-2m, \dots, T-m\}$






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

$n=5, m=5$

$T=$	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

计算 $f[T]$ 时要决策：前一次发车是哪个时刻 $p$

费用：时刻 $\{p+1, p+2, \dots, T\}$ 的所有人等待总时间






例如 $T=8, p=1$ , 时刻 $\{2, 3, \dots, 8\}$ 共3人

$$\begin{aligned} \text{等待总费用} \text{cost} &= (8-4) + (8-4) + (8-2) \\ &= 8 * \mathbf{3} - (\mathbf{4+4+2}) \end{aligned}$$

$$T * \text{人数} - (\text{到达时刻总和})$$



$n=5, m=5$

$T=$	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

计算 $f[T]$ 时要决策：前一次发车是哪个时刻 $p$

费用：时刻 $\{p+1, p+2, \dots, T\}$ 的所有人等待总时间

需要知道 $\{p+1, p+2, \dots, T\}$ 时刻共来了几人

$cnt[x]$ 表示时刻 $x$ 共几人

前缀和

$sC[x]$ 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



[L,R]是被T号依赖的格子编号

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



```

50 int main(){
51     freopen("bus.in", "r", stdin);
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);
56     ll maxt=t[n];
57     if(m==1)
58         cout<<0<<endl;
59     else if(m*maxt<=1e7)
60         solveDP();
61     else
62         solveDPopt();
63     return 0;
64 }
    
```

哪怕正解有错  
其他部分可以得分



```
41 void solveDPopt(){
42     move[1]=t[1]-1;
43     for(ll i=2;i<=n;++i)
44         if(t[i]-t[i-1]>2*m)
45             move[i]=
46     for(ll i=1;i<=n;++i) sM[i]=sM[i-1]+move[i];
47     for(ll i=1;i<=n;++i) t[i]-=
48     
49 }
```

# 快快编程作业

2360

2361

820