

1.成绩

算法分析

据说这题当年官方评测时也闹出过乌龙。小数会产生异变，比如以下代码：

```
int  a = 80 * 0.2
```

赋值号右侧是实数，最后结果可能会为16.000001或15.999999，赋值给a的话，a的值可能是16或15，产生错误。解决方法：化乘为除（A,B,C 都是 10的整数倍），或定义为double，最后结果保留0位输出。

```
1  #include <iostream>
2  #include <cstdio>
3  #include <cstring>
4  #define ll long long
5  using namespace std;
6  int main()
7  {
8      int a, b, c;
9      cin>>a>>b>>c;
10     int ans = a * 20 / 100 + b * 30 / 100 + c * 50 / 100;
11     cout<<ans<<endl;
12     return 0;
13 }
```

2.图书管理员

算法分析

n和q是小于1000的，直接枚举就可以。需要预处理10的次方。

```
1  #include <iostream>
2  #include <cstdio>
3  #include <cstring>
4  #include <cmath>
5  #include <algorithm>
6  #define ll long long
7  using namespace std;
8  int x[10], sbook[1010];
9  int main()
10 {
11     x[0] = 1;
12     for (int i = 1; i <= 8; ++i) x[i] = x[i-1] * 10;
13     int n, q;
14     scanf("%d%d", &n, &q);
15     for (int i = 1; i <= n; ++i) scanf("%d", &sbook[i]);
16     sort(sbook + 1, sbook + n + 1);
17     int len, num;
18     for (int i = 1; i <= q; ++i)
19     {
20         scanf("%d%d", &len, &num);
21         int ans = -1;
22         for (int j = 1; j <= n; ++j)
23         {
24             if (sbook[j] % x[len] == num)
25             {
26                 ans = sbook[j];
27                 break;
28             }
29         }
30         printf("%d\n", ans);
31     }
32     return 0;
33 }
```

3.棋盘

算法分析

一个很容易想到的思路是：将无颜色的点拆点，然后转图论求最短路。红色用1，黄色用2，无色用0。坐标(i,j)对应的点编号为 $i * (m - 1) + j$ ，无色的点比如点(x,y)是无色的，拆点后编号 $x * (m - 1) + y$ 的点是红色， $x * (m - 1) + y + m * m$ 的点是黄色。相反地，给定一个点的编号，也可以求出它的对应的坐标。有几个细节：

- 1.变色是一种花费，走过去是否要花费还要看两者的颜色是否相同，颜色不相同的话，也得要有花费。
- 2.无色的点，只能被变色，变色后可以连接其相邻的红色或黄色的点，不能连接无色的了。因此，对应变色的无色的点，可以用vis数组标记下。
- 3.建图过程。逐个枚举点u，v是和其相连的点：
  - u是红色，v是红色，则边权为0；v是黄色，则边权为1；v是无色，对v拆点，u到v边权是0，u到 $v + m * m$ 边权为1，标记无色的点v。
  - u是黄色，v是红色，则边权为1；v是黄色，则边权为0；v是无色，对v拆点，u到v边权是1，u到 $v + m * m$ 边权为0，标记无色的点v。
  - u是无色，并且该点被标记过，说明该点被变色过，可以向四周扩展。v是红色，则u到v边权为0， $u + m * m$ 到v边权为1；v是黄色，则u到v边权为1， $u + m * m$ 到v边权为0。
- 4.最短路过程中，如果访问到一个点是无色的，则最短路的代价要加2。以下用的是dij+堆优化。

```
1  #include <iostream>
2  #include <cstdio>
3  #include <cstring>
4  #include <cmath>
5  #include <algorithm>
6  #include <queue>
7  #include <utility>
8  #define ll long long
9  using namespace std;
10 int a[110][110], m, n, vis[110][110];
11 int dx[4] = {-1, 0, 1, 0},
12     dy[4] = {0, 1, 0, -1};
13 struct node
14 {
15     int next, to, val;
16 }edg[100010];
17 int h[20010], cnt, v[20010];
18 int dis[20010], pre[20010];
19 priority_queue< pair<int, int> > q;
20 void sadd(int u, int v, int val)
21 {
22     ++cnt;
23     edg[cnt].next = h[u];
24     edg[cnt].to = v;
25     edg[cnt].val = val;
26     h[u] = cnt;
27 }
28 void dij()
29 {
30     memset(dis, 0x3f, sizeof(dis));
31     q.push(make_pair(0, 1));
32     dis[1] = 0;
33     pre[1] = 0;
34     while (q.size())
35     {
36         int u = q.top().second; q.pop();
37         if (v[u]) continue;
38         v[u] = 1;
39         for (int i = h[u]; i; i = edg[i].next)
40         {
41             int t = edg[i].to, w;
42             if (v[t]) continue; // 如果t是白点，则跳过
43         }
```

```

44         if (t > m * m) w = t - m * m;
45         else w = t;
46         int x, y;
47         if (w % m == 0)
48         {
49             x = w / m; y = m;
50         }else
51         {
52             x = w / m + 1; y = w - w / m * m;
53         }
54         if (a[x][y] == 0) // 无颜色的点
55         {
56             if (dis[t] > dis[u] + edg[i].val + 2 ) //
57             {
58                 dis[t] = dis[u] + edg[i].val + 2;
59                 q.push(make_pair(-dis[t], t));
60             }
61         }else
62         {
63             if (dis[t] > dis[u] + edg[i].val)
64             {
65                 dis[t] = dis[u] + edg[i].val ;
66                 q.push(make_pair(-dis[t], t));
67             }
68         }
69     }
70 }
71 }
72 int main()
73 {
74     scanf("%d%d", &m, &n);
75     int x, y, c;
76     for (int i = 1; i <= n; ++i)
77     {
78         scanf("%d%d%d", &x, &y, &c);
79         a[x][y] = ++c; // 1:红色 2:黄色 0:无色
80     }
81     for (int i = 1; i <= m; ++i)
82         for (int j = 1; j <= m; ++j)
83         {
84             for (int k = 0; k < 4; ++k)
85             {
86                 x = i + dx[k], y = j + dy[k];
87                 if (x < 1 || x > m || y < 1 || y > m) continue;
88                 if (a[i][j] == 1)
89                 {
90                     if (a[x][y] == 1) sadd(m * (i - 1) + j, m * (x - 1) + y, 0);
91                     else if (a[x][y] == 2) sadd(m * (i - 1) + j, m * (x - 1) + y, 1);
92                     else
93                     {
94                         sadd(m * (i - 1) + j, m * (x - 1) + y, 0); //
95                         sadd(m * (i - 1) + j, m * (x - 1) + y + m * m, 1); //
96                         vis[x][y] = 1; // 该点被拆过
97                     }
98                 }else if (a[i][j] == 2)
99                 {
100                     if (a[x][y] == 1) sadd(m * (i - 1) + j, m * (x - 1) + y, 1);
101                     else if (a[x][y] == 2) sadd(m * (i - 1) + j, m * (x - 1) + y, 0);
102                     else
103                     {
104                         sadd(m * (i - 1) + j, m * (x - 1) + y, 1); //
105                         sadd(m * (i - 1) + j, m * (x - 1) + y + m * m, 0); //
106                         vis[x][y] = 1; // 该点被拆过
107                     }
108                 }else if (vis[i][j]) // 该点被拆过
109                 {
110                     if (a[x][y] == 1)
111                     {

```

```

112         sadd(m * (i - 1) + j, m * (x - 1) + y, 0);
113         sadd(m * (i - 1) + j + m * m, m * (x - 1) + y, 1);
114     }
115     else if (a[x][y] == 2)
116     {
117         sadd(m * (i - 1) + j, m * (x - 1) + y, 1);
118         sadd(m * (i - 1) + j + m * m, m * (x - 1) + y, 0);
119     }
120 }
121 }
122 }
123 dij();
124 if (a[m][m] == 0)
125 {
126     if (dis[m * m] == 0x3f3f3f3f && dis[m * m * 2] == 0x3f3f3f3f) printf("-1\n");
127     else printf("%d\n", min(dis[m * m], dis[m * m * 2]));
128 }else
129 {
130     if (dis[m * m] == 0x3f3f3f3f ) printf("-1\n");
131     else printf("%d\n", dis[m * m]);
132 }
133 return 0;
}

```

## 算法拓展

1.bfs求最短路。以上方法建图之后，边权为0或1，用bfs+双端队列也可以求出最短路，时间效率更高。

2.dfs+剪枝。因为可以向四个方向移动，所以记忆化有后效性。可以考虑dfs。dfs(int x,int y,int val,int c)：已经搜索到(x,y)处，最短路为val，(x,y)的颜色为c。无色的点不能向无色的点走。设全局变量minval记录最值。

一个思维上的优化：如果u是有颜色的，v是无色的，v变色后和u的颜色一样即可，效果不会更差。

```

1  #include <iostream>
2  #include <cstdio>
3  #include <cstring>
4  #include <cmath>
5  #include <algorithm>
6  #define ll long long
7  using namespace std;
8  int a[110][110], m, n, vis[110][110];
9  int f[110][110];
10 int dx[4] = {-1, 0, 1, 0},
11     dy[4] = {0, 1, 0, -1};
12 // f[x][y]:(x, y)到(m, m)的最小花费
13 int minval = 1e8;
14 void dfs(int x, int y, int val, int c)
15 {
16     if (val > minval) return;
17     if (x == m && y == m)
18     {
19         minval = min(minval, val);
20         return;
21     }
22     for (int k = 0; k < 4; ++k)
23     {
24         int sx = x + dx[k], sy = y + dy[k];
25         if (sx < 1 || sx > m || sy < 1 || sy > m) continue;
26         if (vis[sx][sy]) continue;
27         if (a[x][y] == 0 && a[sx][sy] == 0) continue;
28         vis[sx][sy] = 1;
29         if (a[x][y])
30         {
31             if (a[sx][sy])
32             {
33                 if (f[sx][sy] > f[x][y] + (a[x][y] == a[sx][sy] ? 0 : 1))
34                 {

```

```

35         f[sx][sy] = f[x][y] + (a[x][y] == a[sx][sy] ? 0 : 1);
36         dfs(sx, sy, f[sx][sy], a[sx][sy]);
37     }
38     }else
39     {
40         if (f[sx][sy] > f[x][y] + 2)
41         {
42             f[sx][sy] = f[x][y] + 2;
43             dfs(sx, sy, f[sx][sy], a[x][y]);
44         }
45     }
46     }else
47     {
48         if (f[sx][sy] > f[x][y] + (c == a[sx][sy] ? 0 : 1))
49         {
50             f[sx][sy] = f[x][y] + (c == a[sx][sy] ? 0 : 1);
51             dfs(sx, sy, f[sx][sy], a[sx][sy]);
52         }
53     }
54     }
55     vis[sx][sy] = 0;
56 }
57 }
58 int main()
59 {
60     scanf("%d%d", &m, &n);
61     int x, y, c;
62     for (int i = 1; i <= n; ++i)
63     {
64         scanf("%d%d%d", &x, &y, &c);
65         a[x][y] = ++c; // 1:红色 2:黄色 0:无色
66     }
67     memset(f, 0x3f, sizeof(f));
68     vis[1][1] = 1;
69     f[1][1] = 0;
70     dfs(1, 1, 0, a[1][1]);
71     if (minval != 1e8) printf("%d\n", minval); else printf("-1\n");
72     return 0;
73 }

```

#### 4.跳房子

##### 算法分析

对于花费的金币 $g$ ，可以计算出机器人每次弹跳的距离 $[s, t]$ 。弹跳是在一条线上进行的。下面就好想了。dp。设 $f[i]$ ：跳到位置 $i$ 时的最大分数。则：

$$f[i] = \max \{f[j] + i \text{ 的分数} \}$$

$j$ 的取值范围是： $[i - t, i - s]$

要在决策 $j$ 中取最大值的 $f[j]$ ，而且决策 $j$ 有一定的区间范围，用单调队列优化dp。

需要注意的一点：不是所有的位置都能被跳到，只有被更新过的位置才能作为下一个的决策。代码中用ssav数组存待考察的决策，对于其中的决策，如果满足条件，则入队列。

如果枚举的 $g$ 不满足，则枚举更大的 $g$ 。容易证明，枚举 $g$ ，答案具有单调性。可以二分枚举。整体时间复杂度 $O(n \log n)$ 。

```

1  #include <iostream>
2  #include <cstdio>
3  #include <cstring>
4  #include <cmath>
5  #include <algorithm>
6  #include <queue>
7  #define ll long long
8  using namespace std;

```

```

9 struct node
10 {
11     int dis, val;
12 }a[500010];
13 ll f[500010];
14 int ssav[500010], sl, sr, n, d, k;
15 int q[500010];
16 ll chk(int g)
17 {
18     memset(f, 0, sizeof(f));
19     memset(q, 0, sizeof(q));
20     int l, r, s, t;
21     if (g < d)
22     {
23         s = d - g; t = d + g;
24     }else
25     {
26         s = 1; t = d + g;
27     }
28     // 每次跳的距离区间[s, t]
29     l = r = 1;
30     q[1] = 0;
31     f[0] = 0;
32     sl = 1; sr = 0;
33     int i;
34     for (i = 1; i <= n; ++i)
35     {
36         for (int k = sl; k <= sr; ++k)
37         {
38             if (a[ ssav[k] ].dis <= a[i].dis - s )
39             {
40                 ++sl;
41                 while (l <= r && f[ssav[k]] >= f[q[r]]) --r;
42                 q[++r] = ssav[k];
43             }else break;
44         }
45         while (l <= r && a[ q[l] ].dis < a[i].dis - t) ++l;
46         if (l <= r && (a[ q[l] ].dis >= a[i].dis - t && a[ q[l] ].dis <= a[i].dis - s))
47         {
48             ssav[++sr] = i;
49             f[i] = f[q[l]] + a[i].val;
50         }
51     }
52     ll ans = -1e12;
53     for (int i = 1; i <= n; ++i) ans = max(ans, f[i]);
54     return ans >= k;
55 }
56 int main()
57 {
58     scanf("%d%d%d", &n, &d, &k);
59     for (int i = 1; i <= n; ++i) scanf("%d%d", &a[i].dis, &a[i].val);
60     int l = 0, r = 500000 + 1;
61     while (l < r)
62     {
63         int g = (l + r) >> 1;
64         if (chk(g)) r = g; else l = g + 1;
65     }
66     if (l == 500000 + 1) printf("-1\n"); else printf("%d\n", l);
67     return 0;
68 }

```