

1. scanf printf 输入输出优化

```
1 double :%lf
2
3 long long:%lld
4
5 ios::sync_with_stdio(false);
6 cin.tie(0);
7 cout.tie(0);
8
9 用"\n"而不是endl
10
11
```

2. mod运算

```
1 a+b: (a%MOD+b%MOD)%MOD
2 a-b: ((a-b)%MOD+MOD)%MOD
```

3. 前缀和

3.1. 一维

3.2. 二维

```
int x1,x2,y1,y2;//x1,y1是左上角的坐标, 另一对是右下角的坐标
cin >> x1 >> y1 >> x2 >> y2;
cout << f[x2][y2] - f[x1 - 1][y2] - f[x2][y1 - 1] + f[x1 - 1][y1 - 1];
```

4. 差分

5. 二分

5.1. 第一种

```
1  int main()
2  {
3      int l;
4      int r;
5      while(l < r)
6      {
7          int mid = (l + r) / 2;
8          if(check())
9          {
10             r = mid; // 这里是 r = mid, 说明[l,mid]是合法范围
11          }
12          else
13          {
14             l = mid + 1; // [l,mid]这个范围都不是合法范围, 所以下一次查找直接从
15             l = mid + 1开始了
16             //最后的l,r是答案 因为 l == r, 最终就是答案。
17          }
18      }
19  }
20
```

5.2. 第2种

```
1  int main()
2  {
3      int l;
4      int r;
5      while(l < r)
6      {
7          int mid = (l + r + 1) / 2; // 这里要 l + r + 1 要不然会死循环
8          if(check())
9          {
10             l = mid; // mid这个位置 满足条件之后 查找 [mid, right]的位置, 所以l移到mid的位置
11          }
12          else
13          {
14             r = mid - 1; // [mid,r] 不满足条件, 所以要移到满足条件的一方, r
15             = mid - 1
16          }
17          //最后的l,r是答案 因为 l == r
18      }
19  }
20
21
```

6. 单调栈

```
1  #include <bits/stdc++.h>
2  using namespace std;
3  #define int long long
4  const int N = 1e5 + 5;
5  int a[N];
6  int l[N];
7  int r[N];
8  int n;
9  stack<pair<int, int>>s1;
10 stack<pair<int, int>>s2;
11 int ans;
12 signed main()
13 {
14     cin >> n;
15     for (int i = 1; i <= n; i++)
16     {
17         cin >> a[i];
18     }
19     for (int i = 1; i <= n; i++)
20     {
21         while (!s1.empty() && s1.top().second >= a[i]) // 左边第一个小于它的
22         {
23             s1.pop();
24         }
25
26         if (s1.empty())
27         {
28             l[i] = 0;
29         }
30         else
31         {
32             l[i] = s1.top().first;
33         }
34         s1.push({ i, a[i] });
35     }
36     for (int i = n; i ; i--)
37     {
38         while (!s2.empty() && s2.top().second >= a[i])
39         {
40             s2.pop();
41         }
42
43         if (s2.empty())
44         {
45             r[i] = n+1;
46         }
47         else
48         {
49             r[i] = s2.top().first;
50         }
```

```

51     s2.push({ i,a[i] });
52 }
53
54 for (int i = 1; i <= n; i++)
55 {
56     int l1 = l[i] + 1;
57     int r1 = r[i] - 1;
58
59     int length = r1 - l1 + 1;
60
61     ans = max(ans, length * a[i]);
62 }
63 cout << ans;
64
65
66
67
68 return 0;
69 }

```

7. kruskal

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  #define int long long
4  const int N=4e5+50;
5  struct edge
6  {
7      int a,b,w;
8      bool operator<(const edge&e)
9      {
10         return w<e.w;
11     }
12 };
13 edge e[N];
14
15 int fa[N];
16
17 int find(int u)
18 {
19     if (u!=fa[u])
20     {
21         fa[u]=find(fa[u]);
22     }
23     return fa[u];
24 }
25 void merge(int a,int b)
26 {
27     a=find(a);
28     b=find(b);
29
30     fa[a]=b;

```

```

31 }
32 int n,m;
33 void kruskal()
34 {
35     for(int i=1;i<=n;i++)
36     {
37         fa[i]=i;
38     }
39     int cnt=0;
40     int ans=0;
41     sort(e+1,e+1+m);
42     for(int i=1;i<=m;i++)
43     {
44         int a=e[i].a;
45         int b=e[i].b;
46         if (find(a)!=find(b))
47         {
48             ans+=e[i].w;
49             merge(a,b);
50             cnt++;
51         }
52     }
53     // cout<<cnt<<endl;
54     if (cnt==n-1)
55     {
56         cout<<ans<<endl;
57     }
58     else
59     {
60         cout<<"impossible"<<endl;
61     }
62 }
63 }
64 signed main()
65 {
66     cin>>n>>m;
67     for(int i=1;i<=m;i++)
68     {
69         cin>>e[i].a>>e[i].b>>e[i].w;
70     }
71     kruskal();
72 }
73 return 0;
74 }

```

8. prim

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 const int N=2*500+10;
4 int g[N][N];
5 int n,m;

```

```

6  const int INF=0x3f3f3f3f;
7  const int fill_val=0x3f;
8  bool vis[N];
9  int dis[N];
10 int ans;
11 void prim()
12 {
13     memset(dis,fill_val,sizeof dis);
14     dis[1]=0;
15     for(int i=0;i<n;i++)
16     {
17         int t=-1;
18         for(int j=1;j<=n;j++)
19         {
20             if(!vis[j]&&(t== -1 || dis[j]<dis[t]))
21             {
22                 t=j;
23             }
24         }
25         if(dis[t]==INF)
26         {
27             printf("impossible");
28             return;
29         }
30         ans+=dis[t];
31         vis[t]= true;
32
33         for(int j=1;j<=n;j++)
34         {
35             dis[j]=min(dis[j],g[t][j]);
36         }
37     }
38     printf("%d",ans);
39 }
40 int main()
41 {
42     scanf("%d%d",&n,&m);
43     /*for(int i=1;i<=n;i++)
44     {
45         for(int j=1;j<=n;j++)
46         {
47             if(i!=j)
48             {
49                 g[i][j]=INF;
50             }
51         }
52     }*/
53     memset(g,fill_val,sizeof g);
54     for(int i=0;i<m;i++)
55     {
56         int u,v,w;
57         scanf("%d%d%d",&u,&v,&w);
58         g[v][u]=g[u][v]=min(g[u][v],w);
59     }
60     prim();

```

```
61
62
63     return 0;
64 }
```

9. dijkstra

```
1  #include <bits/stdc++.h>
2  using namespace std;
3  #define int long long
4  const int N=2e5+50;
5  //int belong[N];
6  //vector<int>block[N];
7  int d[N];
8  bool vis[N];
9  int h[N];
10 int ne[N];
11 int w[N];
12 int target[N];
13 int idx=1;
14 void add(int a,int b,int c)
15 {
16     target[idx]=b;
17     w[idx]=c;
18     ne[idx]=h[a];
19     h[a]=idx++;
20 }
21 int n,m;
22 #define pii pair<int,int>
23 void dijkstra()
24 {
25     memset(d,0x3f,sizeof d);
26     d[1]=0;
27     priority_queue<pii,vector<pii>,greater<pii>>q;
28
29     q.push({0,1});
30     while (!q.empty())
31     {
32         auto t=q.top();
33         q.pop();
34         int v=t.second,dis=t.first;
35
36         if (vis[v])
37         {
38             continue;
39         }
40         vis[v]= true;
41
42         for(int i=h[v];i;i=ne[i])
43         {
44             int j=target[i];
45             if (d[v]+w[i]<d[j])
```

```

46         {
47             d[j]=d[v]+w[i];
48             q.push({d[j],j});
49         }
50     }
51 }
52
53 if (d[n]>0x3f3f3f3f)
54 {
55     cout<<-1<<endl;
56 }
57 else
58 {
59     cout<<d[n]<<endl;
60 }
61
62 }
63 signed main()
64 {
65     cin>>n>>m;
66     for(int i=0;i<m;i++)
67     {
68         int x,y,z;
69         cin>>x>>y>>z;
70         add(x,y,z);
71     }
72
73     dijkstra();
74
75
76
77     return 0;
78 }

```

10. spfa最短路

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  #define int long long
4
5  int n, m;
6  const int N = 2e5 + 50;
7  int dis[N];
8  bool vis[N];
9  vector<pair<int, int>>to[N]; // dest val
10 void add(int a, int b, int c)
11 {
12     to[a].push_back({ b,c });
13 }
14
15 void spfa()
16 {

```



```

17     memset(dis, 0x3f, sizeof dis);
18     dis[1] = 0;
19     queue<int>q;
20     q.push(1);
21     while (!q.empty())
22     {
23         auto t = q.front();
24         q.pop();
25         vis[t] = false;
26         for (auto& i : to[t])
27         {
28             int w = i.second;
29             int dest = i.first;
30             if (dis[t] + w < dis[dest])
31             {
32                 dis[dest] = dis[t] + w;
33                 if (!vis[dest])
34                 {
35                     vis[dest] = true;
36                     q.push(dest);
37                 }
38             }
39         }
40     }
41
42     if (dis[n] > 0x3f3f3f3f)
43     {
44         cout << "impossible";
45     }
46     else
47     {
48         cout << dis[n];
49     }
50     cout << endl;
51 }
52
53 signed main()
54 {
55     cin >> n >> m;
56     for (int i = 0; i < m; i++)
57     {
58         int x, y, z;
59         cin >> x >> y >> z;
60         add(x, y, z);
61     }
62     spfa();
63
64
65     return 0;
66 }

```

11. spfa求负环

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  const int N=20000+50;
4  int n;
5  int w[N],h[N],ne[N],target[N];
6  int idx=1;
7  bool vis[N];
8  int dis[N];
9  void add(int a,int b,int c)
10 {
11     w[idx]=c;
12     target[idx]=b;
13     ne[idx]=h[a];
14     h[a]=idx++;
15 }
16 int m;
17 int cnt[N];
18 const int fill_val=0x3f;
19 const int INF=0x3f3f3f3f;
20 void spfa()
21 {
22     memset(dis,fill_val,sizeof dis);
23     dis[1]=0;
24     queue<int>q;
25     for(int i=1;i<=n;i++)
26     {
27         q.push(i);
28         vis[i]= true;
29     }
30     while (!q.empty())
31     {
32         auto t=q.front();
33         q.pop();
34         vis[t]= false;
35         for(int i=h[t];i;i=ne[i])
36         {
37             int j=target[i];
38             if(dis[t]+w[i]<dis[j])
39             {
40                 dis[j]=dis[t]+w[i];
41                 cnt[j]=cnt[t]+1;
42                 if(cnt[j]>=n)
43                 {
44                     printf("Yes");
45                     return;
46                 }
47                 if(!vis[j])
48                 {
49                     q.push(j);
50                 }
51             }
52         }
53     }
54     printf("No");
55 }

```

```

56 int main()
57 {
58     scanf("%d%d",&n,&m);
59     for(int i=0;i<m;i++)
60     {
61         int x,y,z;
62         scanf("%d%d%d",&x,&y,&z);
63         add(x,y,z);
64     }
65     spfa();
66
67     return 0;
68 }
69

```

12. kosaraju求强联通分量

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  const int N = 2e6 + 50;
4  vector<int>g1[N];
5  vector<int>g2[N];
6  int scc_id;//强联通分量的id
7  int belong[N];//属于那个强联通
8  int br[N];//退出序列
9  int n, m;
10 int in[N];
11 bool vis1[N];
12 bool vis2[N];
13 int br_cnt;
14 void dfs1(int u)
15 {
16     for (auto& i : g1[u])
17     {
18         if (!vis1[i])
19         {
20             vis1[i] = true;
21
22             dfs1(i);
23         }
24     }
25     br[++br_cnt] = u;
26 }
27
28 void dfs2(int u)
29 {
30     for (auto& i : g2[u])
31     {
32         if (!vis2[i])
33         {
34             vis2[i] = true;
35             belong[i] = scc_id;

```

```

36         dfs2(i);
37     }
38 }
39 }
40
41 void kosaraju()
42 {
43     for (int i = 1; i <= n; i++)
44     {
45         if (!vis1[i])
46         {
47             vis1[i] = true;
48             dfs1(i);
49         }
50     }
51
52     for (int i = n; i; i--)
53     {
54         int j = br[i];
55         if (!vis2[j])
56         {
57             vis2[j] = true;
58             scc_id++;
59             belong[j] = scc_id;
60             dfs2(j);
61         }
62     }
63
64     for (int i = 1; i <= n; i++)
65     {
66         for (auto& j : g1[i])
67         {
68             if (belong[i] != belong[j])
69             {
70                 in[belong[j]]++;
71             }
72         }
73     }
74     int ans = 0;
75     for (int i = 1; i <= scc_id; i++)
76     {
77         if (!in[i])
78         {
79             ans++;
80         }
81     }
82     cout << ans;
83
84
85 int main()
86 {
87     cin >> n >> m;
88     for (int i = 0; i < m; i++)
89     {
90         int a, b;

```

```

91         cin >> a >> b;
92         g1[a].push_back(b);
93         g2[b].push_back(a);
94     }
95     kosaraju();
96
97
98     return 0;
99 }

```

13. 树状数组

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  #define int long long
4  const int N=10e5+50;
5  int a[N];
6  int s[N];
7  int n,m;
8  int lb(int v)
9  {
10     return -v&v;
11 }
12 int getAnswer(int pos)
13 {
14     int ans=0;
15     for(int i=pos;i;i-=lb(i))
16     {
17         ans+=s[i];
18     }
19     return ans;
20 }
21 void update(int pos,int v)
22 {
23     for(int i=pos;i<=n;i+=lb(i))
24     {
25         s[i]+=v;
26     }
27 }
28 signed main()
29 {
30     ios::sync_with_stdio(false);
31     cin>>n>>m;
32     for(int i=1;i<=n;i++)
33     {
34         int val;
35         cin>>val;
36         update(i,val);
37     }
38     for(int i=0;i<m;i++)
39     {
40         int flag,x,y;

```

```

41         cin>>flag>>x>>y;
42         if (flag==1)
43         {
44             update(x,y);
45         }
46         else
47         {
48             cout<<getAnswer(y)-getAnswer(x-1)<<endl;
49         }
50     }
51
52
53
54     return 0;
55 }
56

```

14. 线段树

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  #define int long long
4  #define lc 2*pos
5  #define rc lc+1
6  int n,m;
7  const int N=6e5+50;
8  int a[N];
9  int s[N*4];
10 int p,v;
11 int getAnswer(int pos,int l,int r,int p1,int p2)
12 {
13     if (l==p1&&r==p2)
14     {
15         return s[pos];
16     }
17     int mid=l+r>>1;
18     if (p2<=mid)
19     {
20         return getAnswer(lc,l,mid,p1,p2);
21     }
22     else if (p1>mid)
23     {
24         return getAnswer(rc,mid+1,r,p1,p2);
25     }
26     else
27     {
28         int lc_ans=getAnswer(lc,l,mid,p1,mid);
29         int rc_ans=getAnswer(rc,mid+1,r,mid+1,p2);
30
31         return lc_ans+rc_ans;
32     }
33 }

```

```

34 void update(int pos,int l,int r)
35 {
36     if (l==r)
37     {
38         s[pos]+=v;
39         return ;
40     }
41     int mid=l+r>>1;
42     if (p<=mid)
43     {
44         update(lc,l,mid);
45     }
46     else
47     {
48         update(rc,mid+1,r);
49     }
50     s[pos]=s[lc]+s[rc];
51 }
52
53 signed main()
54 {
55     ios::sync_with_stdio(false);
56     cin>>n>>m;
57     for(int i=1;i<=n;i++)
58     {
59         cin>>a[i];
60         p=i;
61         v=a[i];
62         update(1,1,N);
63     }
64     for(int i=0;i<m;i++)
65     {
66         int flag,x,y;
67         cin>>flag>>x>>y;
68         if (flag==1)
69         {
70             p=x;
71             v=y;
72             update(1,1,N);
73         }
74         else
75         {
76             cout<<getAnswer(1,1,N,x,y)<<endl;
77         }
78     }
79
80
81
82
83     return 0;
84 }
85

```

15. 01背包

```
1  #include <bits/stdc++.h>
2  using namespace std;
3  #define int long long
4  const int N=2*1000+50;
5  int f[10][N];
6  int n,c;
7  int val[N];
8  int cost[N];
9  signed main()
10 {
11     cin>>n>>c;
12     for(int i=1;i<=n;i++)
13     {
14         cin>>cost[i]>>val[i];
15     }
16
17     for(int j=1;j<=c;j++)
18     {
19         if(j<=cost[1])
20         {
21             f[1][j]=val[1];
22         }
23     }
24     for(int i=2;i<=n;i++)
25     {
26         int now=i&1;
27         int pre=(i-1)&1;
28
29         for(int j=0;j<=c;j++)
30         {
31             f[now][j]=f[pre][j];
32             if (j<=cost[i])
33             {
34                 f[now][j]=max(f[now][j], f[pre][j-cost[i]]+val[i]);
35             }
36         }
37     }
38     cout<<f[n&1][c];
39
40
41
42
43
44     return 0;
45 }
```

16. 完全背包

```
1  #include <bits/stdc++.h>
```



```

2   using namespace std;
3   #define int long long
4   const int N=2*1000+50;
5   int f[10][N];
6   int n,c;
7   int val[N];
8   int cost[N];
9   signed main()
10  {
11      cin>>n>>c;
12      for(int i=1;i<=n;i++)
13      {
14          cin>>cost[i]>>val[i];
15      }
16
17      for(int i=1;i<=n;i++)
18      {
19          int now=i&1;
20          int pre=(i-1)&1;
21          for(int j=1;j<=c;j++)
22          {
23              f[now][j]=f[pre][j];
24              if (j>=cost[i])
25              {
26                  f[now][j]=max(f[now][j], f[now][j-cost[i]]+val[i]);
27              }
28          }
29      }
30      cout<<f[n&1][c];
31
32
33
34
35      return 0;
36  }

```

17. 多重背包

```

1   #include <bits/stdc++.h>
2   using namespace std;
3   #define int long long
4   const int N=2*2000+50;
5   int f[10][N];
6   int n,c;
7   int val[N];
8   int cost[N];
9   int num[N];
10  vector<int>af_val;
11  vector<int>af_cost;
12  vector<int>af_num;
13  signed main()
14  {

```

```

15     cin>>n>>c;
16
17     for(int i=1;i<=n;i++)
18     {
19         cin>>cost[i]>>val[i]>>num[i];
20     }
21
22     for(int i=1;i<=n;i++)
23     {
24         int start=1;
25         int backup=num[i];
26         while (backup-start>=0)
27         {
28             af_val.push_back(start*val[i]);
29             af_cost.push_back(start*cost[i]);
30             af_num.push_back(start);
31             backup-=start;
32             start*=2;
33         }
34         if (backup)
35         {
36             af_val.push_back(backup*val[i]);
37             af_cost.push_back(backup*cost[i]);
38             af_num.push_back(backup);
39         }
40     }
41     // 0 1
42     for(int i=0;i<af_num.size();i++)
43     {
44         int now=i&1;
45         int pre=(i-1)&1;
46         for(int j=0;j<=c;j++)
47         {
48             f[now][j]=f[pre][j];
49             if (j>=af_cost[i])
50             {
51                 f[now][j]=max(f[now][j], f[pre][j-af_cost[i]]+af_val[i]);
52             }
53         }
54     }
55     cout<<f[(af_num.size()-1)&1][c];
56
57
58     return 0;
59 }

```

18. 分组背包

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  #define int long long
4  const int N = 2 * 2000 + 50;

```

```

5  int f[10][N];
6  vector<pair<int, int>>v[N]; // cost val
7  int n, c;
8
9  signed main()
10 {
11     cin >> n >> c;
12
13     for (int i = 1; i <= n; i++)
14     {
15         int s;
16         cin >> s;
17         for (int j = 0; j < s; j++)
18         {
19             int val, cost;
20             cin >> cost >> val;
21             v[i].push_back({ cost, val });
22         }
23     }
24
25     for (int i = 1; i <= n; i++)
26     {
27         int now = i & 1;
28         int pre = (i - 1) & 1;
29
30         for (int j = 0; j <= c; j++)
31         {
32             f[now][j] = f[pre][j];
33             for (int k = 0; k < v[i].size(); k++) // 填入哪个物品
34             {
35                 if (j >= v[i][k].first)
36                 {
37                     f[now][j] = max(f[now][j], f[pre][j - v[i][k].first] +
v[i][k].second);
38                 }
39             }
40         }
41     }
42
43     cout << f[n & 1][c];
44
45     return 0;
46 }

```

19. 矩阵快速幂

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  #define int long long
4  const int N=20;
5  int n,p;

```

```

6 struct matrix
7 {
8     int a[N][N];
9     int row;
10    int col;
11    matrix(int row,int col):row(row),col(col)
12    {
13        memset(a,0,sizeof a);
14
15
16    }
17    matrix(const matrix&m)
18    {
19        memcpy(a,m.a,sizeof m.a);
20        row=m.row;
21        col=m.col;
22    }
23
24    matrix operator*(matrix&m)
25    {
26        matrix temp(row,m.col);
27
28        for(int i=1;i<=row;i++)
29        {
30            for(int j=1;j<=m.col;j++)
31            {
32                for(int k=1;k<=col;k++)
33                {
34                    temp.a[i][j]+=(a[i][k]*m.a[k][j])%p;
35                    temp.a[i][j]%=p;
36                }
37            }
38        }
39
40        return temp;
41
42    }
43
44    matrix ones(int row)
45    {
46        matrix ans(row,row);
47
48        for(int i=1;i<=row;i++)
49        {
50            ans.a[i][i]=1;
51        }
52        return ans;
53    }
54
55    void fast_pow(int num)
56    {
57        matrix ans=ones(row);
58        matrix t_m=*this;
59
60        int t=num;

```

```

61
62     while (t)
63     {
64         if (t&1)
65         {
66             ans=t_m*ans;
67         }
68         t_m=t_m*t_m;
69         t>>=1;
70     }
71     *this=ans;
72 }
73
74 };
75
76 int t;
77
78
79 signed main()
80 {
81     cin>>t;
82     for(int i=0;i<t;i++)
83     {
84         cin>>n>>p;
85         if (n<=2)
86         {
87             cout<<1<<endl;
88         }
89         else
90         {
91             matrix m(2,2);
92             m.a[1][1]=m.a[1][2]=m.a[2][1]=1;
93
94
95             m.fast_pow(n-2);
96
97             matrix src(2,1);
98             src.a[1][1]=1;
99             src.a[2][1]=1;
100
101
102             matrix ans=m*src;
103
104             cout<<ans.a[1][1]<<endl;
105         }
106     }
107
108     return 0;
109 }

```

20. 分层图 (k层)

21. [JLOI2011] 飞行路线

21.1. 题目描述

Alice 和 Bob 现在要乘飞机旅行，他们选择了一家相对便宜的航空公司。该航空公司一共有 n 个城市设有业务，设这些城市分别标记为 0 到 $n - 1$ ，一共有 m 种航线，每种航线连接两个城市，并且航线有一定的价格。

Alice 和 Bob 现在要从一个城市沿着航线到达另一个城市，途中可以进行转机。航空公司对他们这次旅行也推出优惠，他们可以免费在最多 k 种航线上搭乘飞机。那么 Alice 和 Bob 这次出行最少花费多少？

21.2. 输入格式

第一行三个整数 n, m, k ，分别表示城市数，航线数和免费乘坐次数。

接下来一行两个整数 s, t ，分别表示他们出行的起点城市编号和终点城市编号。

接下来 m 行，每行三个整数 a, b, c ，表示存在一种航线，能从城市 a 到达城市 b ，或从城市 b 到达城市 a ，价格为 c 。

21.3. 输出格式

输出一行一个整数，为最少花费。

21.4. 样例 #1

21.4.1. 样例输入 #1

```
1 | 5 6 1
2 | 0 4
3 | 0 1 5
4 | 1 2 5
5 | 2 3 5
6 | 3 4 5
7 | 2 3 3
8 | 0 2 100
```

21.4.2. 样例输出 #1

```
1 | 8
```

21.5. 提示

21.5.0.1. 数据规模与约定

对于 30% 的数据， $2 \leq n \leq 50$ ， $1 \leq m \leq 300$ ， $k = 0$ 。

对于 50% 的数据， $2 \leq n \leq 600$ ， $1 \leq m \leq 6 \times 10^3$ ， $0 \leq k \leq 1$ 。

对于 100% 的数据， $2 \leq n \leq 10^4$ ， $1 \leq m \leq 5 \times 10^4$ ， $0 \leq k \leq 10$ ， $0 \leq s, t, a, b \leq n$ ， $a \neq b$ ， $0 \leq c \leq 10^3$ 。

另外存在一组 hack 数据。

代码：

```
1  #include <bits/stdc++.h>
2  using namespace std;
3  #define int long long
4  const int N=2e5+50;
5  #define pii pair<int,int>
6  vector<pii>to[N];
7  void add(int a,int b,int c)
8  {
9      to[a].push_back({b,c});
10 }
11 int n,m,k;
12 int s,t;//begin last
13 int d[N];
14 bool vis[N];
15 void dijkstra()
16 {
17     memset(d,0x3f,sizeof d);
18     d[s]=0;
19     priority_queue<pii,vector<pii>,greater<pii>>q;
20     q.push({0,s});
21     while (!q.empty())
22     {
23         auto t=q.top();
24         q.pop();
25
26         int v=t.second;
27         if (vis[v])
28         {
29             continue;
30         }
31         vis[v]= true;
32
33         for(auto&i:to[v])
34         {
35             int j=i.first;
36             int w=i.second;
37             if (d[v]+w<d[j])
38             {
39                 d[j]=d[v]+w;
40                 q.push({d[j],j});
41             }
42         }
43     }
44     //k n-1
45     int ans=0x3f3f3f3f;
46     for(int i=0;i<=min(k,n-1);i++)
47     {
48         ans=min(ans,d[i*n+t]);
49     }
50     cout<<ans;
```

```

51
52 }
53 signed main()
54 {
55     cin>>n>>m>>k;
56     cin>>s>>t;
57     for(int i=0;i<m;i++)
58     {
59         int a,b,c;
60         cin>>a>>b>>c;
61         for(int j=0;j<15;j++)
62         {
63             add(j*n+a,j*n+b,c);
64             add(j*n+b,j*n+a,c);
65         }
66         for(int j=0;j<14;j++)
67         {
68             add(j*n+a,(j+1)*n+b,0);
69             add(j*n+b,(j+1)*n+a,0);
70         }
71     }
72     dijkstra();
73
74
75
76     return 0;
77 }

```

22. 分层图2层

D: 机关秘境

[提交](#)
[记录](#)

题目描述

“留云借风真君”十分无聊，于是他发明了一个新的机关，由于他的两个老朋友都出去玩了，于是他开出了“100 原石”的高价，让旅行者来测试机关，小 A 学业繁忙，当然不可能为了区区“100 原石”来打这个秘境，于是小 A 希望你帮忙来打，作为今天的第四个委托。

秘境中有 n 个浮空岛，岛之间有 m 个桥相连，旅行者从起点走到终点即可通过秘境（保证旅行者能够从起点到达终点）。 n 个岛的编号为 $1 \rightarrow n$ ，1 号岛是起点， n 号岛是终点。

桥在初始时，处于混沌状态，可能崎岖不平，可能此路不通，可能一马平川……桥的状态每一秒都在变化，在不同的状态下，桥可能会变得无法通过，或有不同的通过时间。对于一个确定的桥而言，它只能从**两种**不同的状态之间来回切换，具体来说，桥在开始时是第一种状态，一秒后变为第二种状态，再过一秒变回第一种状态，以此类推。（对于状态的详细描述，见输入格式）

旅行者可以在岛上等待桥的变化，而当旅行者走到桥上的那一瞬间开始，桥就会退出混沌状态，保持此刻的状态，此后不再变化了。

根据攻略，你已经知道了所有的桥在不同的状态下，通过所需要的时间。旅行者需要在尽可能短的时间内，从起点走到终点，请你帮忙计算出通过秘境所需的最短时间。

输入格式

第一行两个数字 n, m ，分别表示岛的数量和桥的数量。

接下来 m 行，每行 4 个数字 u, v, t_1, t_2 ，表示桥连接了编号为 u 和 v 的两个岛，两种状态通过的时间分别为 t_1, t_2 。如果通过时间为 -1 ，则表示此时不能过桥。


```

1  #include <bits/stdc++.h>
2  using namespace std;
3  #define int long long
4  const int N = 2e5 + 50;
5  int n, m;
6  vector<pair<int, int>>v[N]; //dest val
7  int d[N];
8  bool vis[N];
9  void add(int a, int b, int c)
10 {
11     v[a].push_back({ b,c });
12 }
13 #define pii pair<int,int>
14 void dijkstra()
15 {
16     memset(d, 0x3f, sizeof d);
17
18     d[1] = 0;
19     priority_queue<pii, vector<pii>, greater<pii>>q;
20     q.push({ 0,1 });
21
22
23     while (!q.empty())
24     {
25         auto t = q.top();
26         q.pop();
27
28         int u = t.second;
29         if (vis[u])
30         {
31             continue;
32         }
33         vis[u] = true;
34
35         for (auto& iter : v[u])
36         {
37
38             int j = iter.first;
39             int w = iter.second;
40             if (d[u] + w < d[j])
41             {
42                 d[j] = d[u] + w;
43                 q.push({ d[j],j });
44             }
45         }
46     }
47     /*for (int i = 1; i <= n; i++)
48     {
49         cout << "i==" << i << "   d[i]==" << min(d[i], d[i + n]) << endl;
50     }*/
51     cout<<min(d[n],d[2*n]);
52 }
53
54
55 signed main()

```

```

56 {
57     cin >> n >> m;
58     for (int i = 0; i < m; i++)
59     {
60         int a, b, t1, t2;
61         cin >> a >> b >> t1 >> t2;
62
63         if (t1 != -1)
64         {
65             if (t1 & 1)
66             {
67                 add(a, n + b, t1);
68                 add(b, n+a, t1);
69             }
70             else
71             {
72                 add(a, b, t1);
73                 add(b, a, t1);
74             }
75         }
76         if (t2 != -1)
77         {
78             add(a, n + a, 1);
79             add(n+a,a,1);
80             if (t2 & 1)
81             {
82                 add(n+a, b, t2);
83                 add(n+b, a, t2);
84             }
85             else
86             {
87                 add(n + a, n + b, t2);
88                 add(n + b, n + a, t2);
89             }
90         }
91     }
92
93     /*for(int i=1;i<=2*n;i++)
94     {
95         cout<<"i=="<<i<<"\n";
96         for(auto &iter:v[i])
97         {
98             cout<<"dest=="<<iter.first<<"   val=="<<iter.second<<endl;
99         }
100     }*/
101
102     dijkstra();
103
104
105     return 0;
106 }

```

23. 没有上司的舞会

```
1  #include <bits/stdc++.h>
2  using namespace std;
3  #define int long long
4  const int N=2*6000+50;
5  int happy[N];
6  vector<int>toEmployee[N];
7  int f[N][10]; // 0表示不来, 1表示来
8  //遍历其每一个下属
9  int n;
10 void find(int pos)
11 {
12     for(int i=0;i<toEmployee[pos].size();i++)
13     {
14         find(toEmployee[pos][i]);
15     }
16
17     f[pos][1]+=happy[pos];
18
19     for(int i=0;i<toEmployee[pos].size();i++)
20     {
21
22         int j=toEmployee[pos][i];
23         f[pos][0]+=max(f[j][0],f[j][1]);
24         f[pos][1]+=f[j][0];
25     }
26
27 }
28
29 bool hasBoss[N];
30 signed main()
31 {
32     cin>>n;
33
34     for(int i=1;i<=n;i++)
35     {
36         cin>>happy[i];
37     }
38
39     for(int i=0;i<n-1;i++)
40     {
41         int l,k;
42         cin>>l>>k;
43         toEmployee[k].push_back(l);
44         hasBoss[l]= true;
45     }
46
47
48     int root=-1;
49     for(int i=1;i<=n;i++)
50     {
51         if (!hasBoss[i])
52         {
```

```
53         root=i;
54         break;
55     }
56 }
57
58 find(root);
59
60 cout<<max(f[root][0],f[root][1]);
61
62
63
64 return 0;
65 }
66
```