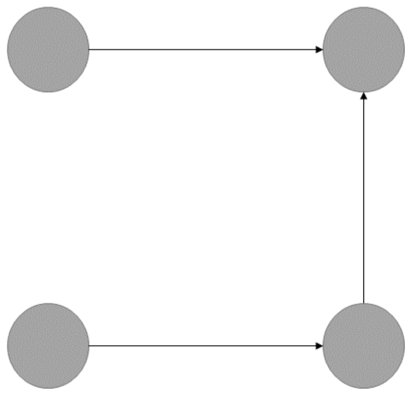
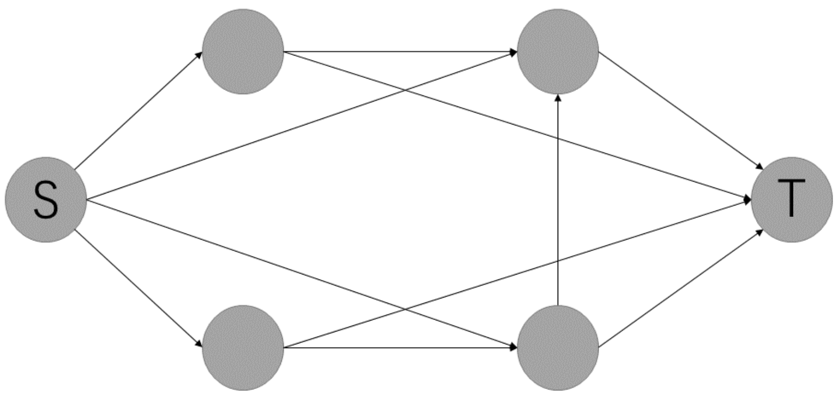
清理雪道

比如有一张这样的图：



如果想要满足题目所要求的每条雪道都要有人清理，那么就是说每条边都要经过。

添加一个源点 S，汇点 T：



一次滑行相当于从 S 到 T 新增了 1 的流量。

那么此题就变成了要求每条边的流量至少为 1，求最小流。

有源汇上下界最小流

~~自学~~

先判断是否可行，用到的是可行流。

无源汇具体方法：

- 把原图建好，流量为上界 - 下界，此时最大流不一定流量平衡；
- 新建源汇 S、T；
- 设每个点可以流入的最大流量为 in_u，流出的为 ou_u，S 向所有 $in_u > ou_u$ 的点连边，流量为 $in_u - ou_u$，所有 $in_u < ou_u$ 的点向 T 连边流量为 $ou_u - in_u$；
- 对于 S、T 跑最大流；
- 称与新建源汇 S、T 相连的边为附加边，如果当前附加边满流，那么说明当前节点流量平衡，如果所有 S 出去的附加边都满流了，那么说明可行流存在。

如果有源汇（此题），那么从原图的汇点到源点连一条上界正无穷下界为 0 的边，即转化成无源汇（所有到 T 的流量还能回到 S）。

现在要求最小流。

我们现在原图上跑一遍无源汇可行流，再连上汇点到源点的边，流量为 0 到正无穷，再跑一次最大流。此时汇点到源点的边上的流量即为最小流（S 到 T 的所有流量都通过此边回到 S）。

第一次最大流的用处是尽量减小总体的流量，以保证第二次跑出来的流是最小的。

回到题目上来

那么此题的建边就很显然了。

```
void AddEdge(int u, int v, int w) {
  e[++edge_cnt] = Edge(v, w, head[u]);
  head[u] = edge_cnt;
}
void AddEdge(int u, int v, int l, int r) {
  AddEdge(u, v, r - l);
  AddEdge(v, u, 0);
  d[u] -= l;
  d[v] += l;
}

scanf("%d", &n);
int s = n + 1, t = n + 2;
for (int u = 1, m; u <= n; ++u) {
  AddEdge(s, u, 0, INF);
  AddEdge(u, t, 0, INF);
  scanf("%d", &m);
  for (int i = 1, v; i <= m; ++i) {
    scanf("%d", &v);
    AddEdge(u, v, 1, INF);
  }
}
int S = s + 2, T = t + 2;
for (int i = 1; i <= t; ++i) {
  if (d[i] > 0)
    AddEdge(S, i, d[i]), AddEdge(i, S, 0);
  else if (d[i] < 0)
    AddEdge(i, T, -d[i]), AddEdge(T, i, 0);
}
```

然后两次最大流求有源汇上下界最小流。

```
while (Bfs(S, T))
  Dfs(S, INF, T);
AddEdge(t, s, 0, INF);
while (Bfs(S, T))
  Dfs(S, INF, T);
```