

1

有 n 个城市通过 m 个航班连接。每个航班都从城市 u 开始，以价格 w 抵达 v 。现在给定所有的城市和航班，以及出发城市 src 和目的地 dst ，你的任务是找到从 src 到 dst 最多经过 k 站中转的最便宜的价格。如果没有这样的路线，则输出 -1 。

提示：

- n 范围是 $[1, 100]$ ，城市标签从 0 到 $n - 1$ 。
- 航班数量范围是 $[0, n(n - 1)/2]$ 。
- 每个航班的格式 $(src, dst, price)$ 。
- 每个航班的价格范围是 $[1, 10000]$ 。
- k 范围是 $[0, n - 1]$ 。

航班没有重复，且不存在环路。

具体要求：

Bellman-Ford 算法的使用。要注意到转机次数要小于等于 k ，而对一个点利用所有边进行松弛的时候，会出现利用多条边即多次转机的情况。

分析

Bell-Ford算法对0到 $n - 1$ 次转机都进行了松弛。此题规定转机次数要小于等于 k ，于是我们考虑对Bell-Ford算法做出修改。对于每个转机次数分别进行Bell-Ford算法。每趟运行时，只能从上一趟的结果中进行松弛。

伪代码

```

1 function(n, flights, src, dst, k)
2     d[n][k+1] <- inf //初始化距离为正无穷
3     d[src][0] <- 0
4     //更新直达
5     for flight in flights
6         if flight.from==src do
7             d[flight.to][0]=min(d[flight.to][0],flight.price) //避免同程不同价机票
8     //对1到k次转机
9     for i <- 1 to k
10        for i <- 1 to n-1
11            for flight in flights
12                d[flight.to][i]=min(d[flight.to][i],d[flight.from][i-1]+flight.price
13    price <- inf
14    for i <- 0 to k
15        price=min(price,d[dst][i])
16    return (ans == price)?-1:price

```

2

有 N 个网络节点，标记为 1 到 N 。给定一个列表 $times$ ，表示信号经过有向边的传递时间。
 $times[i] = (u, v, w)$ ，其中 u 是源节点， v 是目标节点， w 是一个信号从源节点传递到目标节点的时间。

现在，我们从某个节点 K 发出一个信号。需要多久才能使所有节点都收到信号？如果不能使所有节点收到信号，返回 -1 。

提示：

N 的范围在 $[1, 100]$ 之间。

K 的范围在 $[1, N]$ 之间。

$times$ 的长度在 $[1, 6000]$ 之间。

所有的边 $times[i] = (u, v, w)$ 都有 $1 \leq u, v \leq N$ 且 $0 \leq w \leq 100$ 。

具体要求：

即找到图中某个点到其他所有点的路径，从中选取耗时最长的路径。使用 Dijkstra 算法求出到所有点的 $dist$ 值。

分析

此题为经典的有向权图求解最短路径。于是只需简单运用Dijkstra算法进行求解。求解后取最短路径的最大值即得问题解。

伪代码

```
1 function(times, n, k)
2     d[n] <- inf
3     S <- set()
4     Q <- set(range(1,n+1))
5     w[k][k] <- inf
6     for time in times
7         w[time.u][time.v] <- time.w
8     for i <- 1 to k
9         d[i] <- w[k][i]
10    d[k] <- 0
11    S.add(k)
12    Q.del(k)
13    \\dijkstra
14    while(Q.len()>0)
15        u=d.min().index
16        S.add(u)
17        Q.del(u)
18        for i in Q
19            d[i] <- min(d[i],d[u]+w[u][i])
20
21    return d.max()==inf ? -1 : d.max()
```

3

某省自从实行了很多年的畅通工程计划后，终于修建了很多路。不过路多了也不好，每次要从一个城镇到另一个城镇时，都有许多种道路方案可以选择，而某些方案要比另一些方案行走的距离要短很多。这让行人很困扰。现在，已知起点和终点，请你计算出要从起点到终点，最短需要行走多少距离。

提示：

本题目包含多组数据，每组数据第一行包含两个正整数 N 和 M ($0 < N < 200, 0 < M < 1000$)，分别代表现有城镇的数目和已修建的道路的数目。城镇分别以 $0 \sim N - 1$ 编号。接下来是 M 行道路信息。每一行有三个整数 A, B, X ($0 \leq A, B < N, A \neq B, 0 < X < 10000$)，表示城镇 A 和城镇 B 之间有一条长度为 X 的双向道路。

再接下一行有两个整数 S, T ($0 \leq S, T < N$)，分别代表起点和终点。对于每组数据，请在一行里输出最短需要行走的距离。如果不存在从 S 到 T 的路线，就输出 -1 。

具体要求：

用 Floyd 求任意两点最短路后，选取出题目要求的输出。

分析

按要求使用Floyd算法。

伪代码

```
1 function Floyd(N,M,S,T)
2 //创建邻接矩阵
3 f[N][N] <- inf
4 for m in M
5     f[m.A][m.B] <- m.X
6 for i <- 0 to N-1
7     f[i][i] <- 0
8 //Floyd
9 for k <- 0 to N-1 do
10     for i <- 0 to N-1 do
11         for j <- 0 to N-1 do
12             f[i][j] <- min(f[i][j], f[i][k] + f[k][j])
13 return f[S][T]==inf ? -1 : f[S][T]
```

4

给出一个网络图，以及其源点和汇点，求出其网络最大流。

提示：

输入格式：第一行包含四个正整数 n, m, s, t ，分别表示点的个数、有向边的个数、源点序号、汇点序号。接下来 M 行每行包含三个正整数 u_i, v_i, w_i ，表示第 i 条有向边从 u_i 出发，到达 v_i ，边权为 w_i （即该边最大流量为 w_i ）。

输出格式：一行，包含一个正整数，即为该网络的最大流。

具体要求：

使用 Ford-Fulkerson 算法。

分析

按要求使用Ford-Fulkerson算法。

伪代码

```

1 function(G, s, t)
2     for each edge in G.E
3         edge.f = 0
4
5     // 在余图中寻找增广路径, 并更新
6     while((p <- search(G)))
7         fmin = findminf(p)
8         for edge in p
9             if edge in G.E
10                 edge.f <- edge.f + fmin
11             else
12                 edge.f <- edge.f - fmin
13
14     // 返回最大流
15     ans = 0
16     for edge in G.E
17         if u == s
18             ans <- ans + edge.f
19         if v == s
20             ans <- ans - edge.f
21     return ans

```

5

题目内容:

学校放假了, 有些同学回家了, 而有些同学则有以前的好朋友来探访, 那么住宿就是一个问题。

比如 A 和 B 都是学校的学生, A 要回家, 而 C 来看 B , C 与 A 不认识。我们假设每个人只能睡和自己直接认识的人的床。那么一个解决方案就是 B 睡 A 的床而 C 睡 B 的床。而实际情况可能非常复杂, 有的人可能认识好多在校学生, 在校学生之间也不一定都互相认识。

我们已知一共有 n 个人, 并且知道其中每个人是不是本校学生, 也知道每个本校学生是否回家。问是否存在一个方案使得所有不回家的本校学生和来看他们的其他人都有地方住。

提示:

输入格式: 第一行一个数 T 表示数据组数。接下来 T 组数据, 每组数据第一行一个数 n 表示涉及到的总人数。接下来一行 n 个数, 第 i 个数表示第 i 个人是否是在校学生 (0 表示不是, 1 表示是)。再接下来一行 n 个数, 第 i 个数表示第 i 个人是否回家 (0 表示不回家, 1 表示回家, 注意如果第 i 个人不是在校学生, 那么这个位置上的数是一个随机的数, 你应该在读入以后忽略它)。接下来 n 行每行 n 个数, 第 i 行第 j 个数表示 i 和 j 是否认识 (1 表示认识, 0 表示不认识, 第 i 行 i 个的值为 0, 但是显然自己还是可以睡自己的床), 认识的关系是相互的。

输出格式：

对于每组数据，如果存在一个方案则输出 ^_^， 否则输出 T_T。(注意输出的都是半角字符，即三个符号的 ASCII 码分别为 94, 84, 95)

具体要求：

应用匈牙利算法求最大匹配问题。

分析

此题情境可抽象为二分图的匹配问题。

左边节点为所有学生的床，右边节点为留校学生和来访学生的和。当床的主人与学生认识时，两节点存在边（显然床的主人认识自己）。于是求最大匹配，当最大匹配是匹配了所有人时，则输出 ^_^， 否则输出 T_T 。

伪代码

```

1 //先按照分析建立二分图
2 //二分图结构为床顶点集U, 人顶点集V, 边E
3 create(H)
4     G <- emptygraph()
5     //初始化顶点集
6     for people in H.peoples
7         if (people.isstudent)
8             G.U.add(people.index)
9             if(!(people.isleave))
10                 G.V.add(people.index)
11         else
12             G.V.add(people.index)
13     //初始化边集
14     for a in H.a
15         if(a.x in G.U and a.y in G.V)
16             G.E.add(newedge(a.x,a.y)
17     //自己可以用自己的床
18     for a in E.U
19         G.E.add(newedge(a,a))
20 //求取最大匹配
21 search(G)
22     M <- G
23     M.E <- emptyset()
24     //搜索增广路径
25     while((p <- search(G,M))
26         for edge in p
27             if edge in G.E
28                 M.E.add(edge)
29                 G.E.del(edge)
30             else
31                 G.E.add(edge)
32                 M.E.del(edge)
33     //此时M为最大匹配
34     count <- 0
35     existset <- emptyset()
36     for edge in M.E
37         if(!(edge.v in existset))
38             count <- count+1
39             existset.add(edge.v)
40     return count==len(M.E) ? 1 : 0
41
42 function(H)
43     arr <- search(create(H)) ? `^_^` : `T_T`
44     print(arr)

```