第八章-作业

请按照题目中要求的具体算法用伪代码进行题目解答。

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。

原题：787. K 站中转内最便宜的航班 - 力扣（LeetCode）

答：

思路分析：

注意到 n 的范围很小，考虑改进算法，定义 $d[i][j]$ 表示从 src 出发到城市 i 恰好经过 j 次中转的最便宜价格，然后对每个 k 进行一次 *Bellman-ford* 算法。

伪代码：

```
function findCheapestPrice(n, flights, src, dst, k)
     d[n][n] <- INFTY // 定义距离数组并初始化为无穷
     // 将能从 src 直达的城市的 d 进行更新
     d[src][0] <- 0
     for each v in flights do
          if v.from == src then
               d[v.to][0] <- Min(d[v.to][0], v.price)
     // 对 1 到 k 次中转的情况分别进行 Bellman-ford 算法
     for r <- 1 to k do
          for i <- 1 to n-1 do
               for each v in flights do
                    if d[v.to][r] > d[v.from][r-1] + v.price then
                         d[v.to][r] <- d[v.from][r-1] + v.price
     // 计算答案
     ans <- INFTY
     for i <- 0 to k do
          ans <- Min(ans, d[dst][i])
     return ans == INFTY ? -1 : ans
```

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。

原题：743. 网络延迟时间 - 力扣（LeetCode）

答：

思路分析：

使用 *Dijkstra* 算法求出所有点的 $dist$，其中的最大值即为答案。

伪代码：

```
function networkDelayTime(times, n, k)
     d[n+1] <- INFTY
     marked[n+1] <- 0
     d[k] <- 0
     for r <- 1 to N-1 do
          x <- 0
          min_d <- INFTY
          for i <- 1 to n do
               if !marked[i] and d[i] < min_d then
                    min_d <- d[i]
                    x <- i
          marked[x] <- 1
          for each e in times do
               if e.u == x then
                    if !marked[e.v] and d[e.v] > d[x] + e.w then
                         d[e.v] <- d[x] + e.w
     ans <- 0
     for i <- 1 to n do
          ans <- Max(ans, d[i])
     return ans == INFTY ? -1 : ans
```

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 $A, B, X\ (0 \leq A, B < N, A! = B, 0 < X < 10000)$,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。

答：

伪代码：

```
function Floyd(N, S, T)
     // 设权矩阵为 f
     for k <- 0 to N-1 do
          for i <- 0 to N-1 do
               for j <- 0 to N-1 do
                    f[i][j] <- Min (f[i][j], f[i][k] + f[k][j])
     return f[S][T] == INFTY ? -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。

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 ui, vi, wi，表示第 i 条有向边从 ui 出发，到达 vi，边权为 wi（即该边最大流量为 wi）。

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。

答：

伪代码：

```
function FORD-FULKERSON(G, s, t)
     // 初始流为0
     for each edge(u, v) in G.E do
          (u, v).f <- 0
```

```
    // 在残存网络中寻找增广路径，并更新流量
    while there exists a path p from s to t in the residual network Gf
        cf(p) <- Min(cf(u, v), (u, v) in p)
        for each edge(u, v) in p do
            if (u, v) in G.E then
                (u, v).f <- (u, v).f + cf(p)
                else (v, u).f <- (v, u).f - cf(p)
    // 返回最大流
    ans <- 0
    for each edge(u, v) in G.E do
        if u == s then
            ans <- ans + (u, v).f
        if v == s then
            ans <- ans - (u, v).f
    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。

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 “^_^” (不含引号) ，否则输出 “T_T”(不含引号)。

(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。

答：

思路分析：建立二分图

- L 为留校学生和校外人员的集合

- R 为所有本校学生的床位

若 L_i 认识（或者就是）R_j 的主人则在 L_i 和 R_j 之间建一条边。

用匈牙利算法求最大二分图匹配，看能否满足 L 中所有人的需求。

伪代码：

- $match$ 数组用来记录每个点的匹配对象
- vis 数组用来在 dfs 中记录点是否被搜索过

```
function dfs(E, u, clk)
     for each edge(u, v) in E do
          if vis[v] != clk then
               vis[v] <- clk
               if match[v] == 0 or dfs(E, match[v], clk) then
                    match[u] <- v
                    return true
     return false
```

```
function Hungarian(E, L, R)
     // 边集、人、床位
     // 若求得的最大匹配数等于 L 的大小，说明存在一个方案；否则说明不存在
     clk <- 0
     ans <- 0
     for each u in L do
          clk ++ // 用来标记搜索
          if dfs(E, u, clk) == true then
               ans ++
     if ans == L.size() then
          return true
          else return false
```