浅谈最短径路问题中的分层思想

福建省泉州市第七中学　吕子鈜

摘要

分层思想与最短路径算法都在许多领域有着广泛的应用。本文通过对一些信息学竞赛试题的分析，从建立模型和优化算法两个方面阐述了分层思想在最短路径问题中的应用，并分别对这两个方面作了总结。

关键字

最短路径　分层　图论

正文

1.　引言

最短路径问题是图论中的一个经典问题。由于问题中边的权值往往可以从距离引申为其他沿路径线性积累的度量，如：时间、花费等，所以最短路径问题在实际生活中有着广泛的应用，如：城市规划、交通导航、网络寻优等。

分层思想作为一个重要的思想，也有着许多应用，特别在是某些高效的方法中，如：动态规划中的阶段划分、图论中基于求阻塞流的最大流算法等。

将分层思想应用到最短路径问题中，正是分层思想和最短路径问题的强强联合。

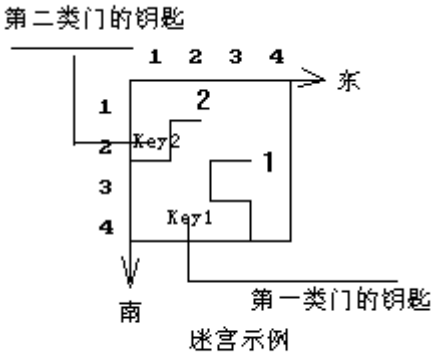
2.　利用分层思想建立模型

这一部分中，作者将简要介绍利用分层思想建立模型的三个问题：拯救大兵瑞恩、fence 和 cow relay，希望能对利用分层思想解题起到抛砖引玉的作用。

2.1 例题一 拯救大兵瑞恩[1]

题目：

有一个长方形的迷宫，被分成了 N 行 M 列，共 N×M 个单元。南北或东西方向相邻的两个单元之间可以互通，或者存在一扇锁着的门，又或者存在一堵不可逾越的墙。迷宫中有一些单元存放着钥匙，并且所有的门被分为 P 类，打开同一类的门的钥匙相同，打开不同类的门的钥匙不同。从一个单元移动到另一个相邻单元的时间为 1，拿取所在单元的钥匙的时间以及用钥匙开门的时间忽略不计。求从(1,1)到(N,M)的最短时间。



N,M 不大于 15，P 不大于 10。

分析：

如果忽略了门和钥匙，我们可以把每个单元看成顶点，相互连通的单元之间连一条边权为 1 的边，那么本题就是一个标准的最短路问题，可以直接使用最短路算法求解。

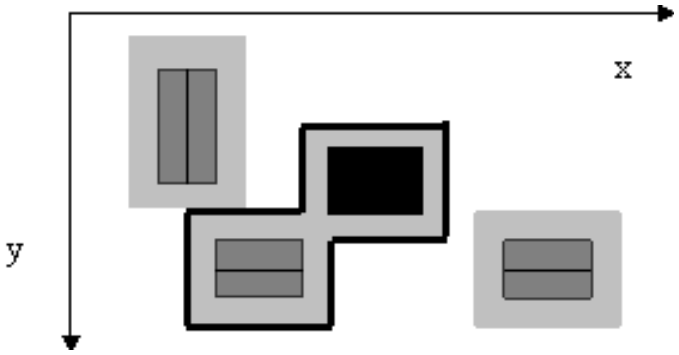
由于有了门和钥匙的因素，所以必须考虑钥匙状态对门的影响。我们把图分成 2^P 层，分别对应持有钥匙的 2^P 种状态，并根据钥匙的状态改造每层节点使相邻的连通节点间有长度为 1 的边。对于存有钥匙的顶点，则应该向表示得到钥匙后钥匙状态的层的对应顶点连一条长度为 0 的边。

在新图上就能直接使用最短路算法求解。由于得到的图所有的边权都为 0 或 1，因此可以使用宽度优先搜索求最短路。时间复杂度和空间复杂度均为 $O(2^PNM)$。

2.2 例题二 fence[2]

题目：

有一个巨大的庄园，由一幢主楼和其他若干幢建筑组成。园主想建一道栅栏围住主楼，但是有一个条件：栅栏不能离建筑太近。更确切的说，每个建筑被包围在一个矩形区域内，而栅栏不能位于这个矩形的内部。矩形的边与坐标轴平行，而篱笆的每个部分也必须和坐标轴平行。求能够围住主楼的栅栏的最小长度。



建筑数 m 不大于 100，所有的坐标是不大于 10000 的非负整数。

[1] Chinese Team Selection Competition 1999

[2] Baltic Olympiad in Informatics 2007

分析：

问题要求最小化一个线性积累的度量，我们尝试用最短路算法解决此问题。

建立一张单层图，把每个矩形的顶点当作是新图的顶点。由于栅栏不能进入矩形内部而且栅栏必须和坐标轴平行，所以当一对顶点之间的矩形区域内没有其他矩形时，在这一对顶点之间添加一条边，边权为两点的曼哈顿距离，表示两个顶点之间能够连接一段长度为边权的栅栏。

在新图中，每一条回路对应着一种栅栏的方案，回路的长度就是栅栏的长度。显然，如果矩形的某个顶点被其他矩形包含，那么在新图中这个顶点的度为 0，因此应该忽略这个点。

但是并不是每一条回路都是满足题意的，因为回路必须包围主楼，亦即主楼必须在回路组成的多边形之中。计算几何中判断点是否在多边形中的方法是：从点引出一条射线，根据射线与多边形交点个数的奇偶性判断是否在形内。因此，我们从主楼引出一条射线。回路是否包含主楼，取决于回路与射线的交点个数的奇偶性。

故我们把图拆分成两层，分别表示与射线交点个数奇偶性不同的两层。如果单层图中的一条边对应的两个顶点在原图中的连线与射线相交，那么在分层图中的边连接不同层的顶点；否则连接同层的顶点。此时，求回路相当于在分层图中求不同层对应的两点之间的最短路。所有对应点对之间最短路中最短的一条的长度即为所求栅栏的最小长度。

假设我们对每个顶点对采用枚举的方法确定顶点之间边的情况，时间复杂度为 $O(m^3)$；对每个顶点都进行一次 Dijkstra 算法求到不同层对应的顶点的最短路，时间复杂度为 $O(m^3)$。故总的时间复杂度为 $O(m^3)$。空间复杂度为 $O(m^2)$。

2.3 例题三 cow relay[3]

题目：

Farmer John 将选 k 只牛组成一个接力队伍。FJ 的农场有 n 块地，m 条唯一的双向边连接两块不同的地，边有一个权值表示牛通过这条边的时间。k 只牛一只接着一只，从第 1 块地跑到第 n 块地。当一只牛第一次到达第 n 块地时，下一只牛就马上从第 1 块地出发。但是任意两只牛的路径必须不同。求出第 k 只牛到达终点的最短时间。

k 不大于 40，n 不大于 800，m 不大于 4000。

分析：

把每块地看成图的顶点，连接两块不同的地的边当成图中的边。非常明显，问题是在边权为正的图中求一个给定顶点对之间的前 k 短路径。

考虑一个非常相似的问题：在边权为正的图中求从一个给定顶点出发的前 k 短路径。我们简单地可以使用宽度优先搜索解决这个问题：维护一个路径长度的优先队列，每次从优先队列中取出长度最短的一条路径，把它删除并加入由这条路径和该路径终点的每条出边分别组成的所有路径，直到得到前 k 条路径。如果图的度为 d，那么时间复杂度不会超过 O(kd+klgk)。

我们可以套用求从一个给定顶点出发的前 k 短路径的算法来解决这个问题，

[3] USACO U S open 2001

但需要把结束条件改成得到前 k 条从 1 到达 n 的路径。由于在边权为正的图中，从 1 到达每个点的路径最多有 k 条可能成为从 1 到 n 的前 k 短路径的子路径，所以拓展的总路径数最多为 kn 条。时间复杂度为 O(km+knlg(kn))。

但是上面的解法的时间复杂度并不令人满意。

再考虑另外一种思路：假设从 1 到 n 的所有路径的集合为 P，第 i 短路为 pi。在每次求出第 i 短路之后，将第 i 短路从图中删除(即 P←P-{pi})，在新图中求最短路即为第 i+1 短路。

我们采用分层的思想实现这一个过程：

用(V，E)表示图，V={1,2… n}表示 n 个节点，E={e_1,e_2… e_m}表示 m 条边。e_k=(i,j)表示由 i 指向 j 的一条有向边，$cost_k$ 或 cost(i,j)则表示该条边的权值。原问题中每一条无向边可以通过拆成两条有向边变成有向。第 i 条最短路 pi={1=v_1,e_1,v_2,e_2… e_{h-1}, v_h=n|v_k∈V,e_k∈E,e_k=(v_k,v_{k+1})}

另设两个节点，s 和 t，增加 2 条边(s,1)和(n,t)，边权均为 0。每一条路径从 1 到 n 的路径第 i 条最短路 pi={1=v_1,e_1,v_2,e_2… e_{h-1}, v_h=n}，都对应一条 s 到 t 的路径 p_{st}i={s,(s,1),1=v_1,e_1,v_2,e_2…e_{h-1}, v_h=n,(n,t),t}，且路径上至少有三条边。令 P_{st} 表示 P 对应的集合。

求出(V,E)中的最短路径 p_{st}1。对于每个节点，将每个节点复制一份加入节点集中，即 V←V+{1’,2’… n’,s’,t’}。对于每条边 e_k=(i,j) ∈E，把(i’,j’)加入边集，即 E←E+(i’,j’)，cost(i’,j’)←cost(i,j)；且如果该边不在 p_{st}1 中，就把(i,j’)加入边集，即 E←E+(i,j’)，cost(i,j’)←cost(i,j)。

由此，我们可以得到一个结论。

结论：如果把原来的节点和对应的复制节点看成相同的点，那么在新图中，从 s 到 t’的路径集合 $P_{st'}$=P_{st}-{p_{st}1}。

简证：如果把原来的节点和对应的复制节点看成相同的点，可以得到

⑴新图没有增加新的边，因此在新图中的从每一条从 s 到 t’路径都属于 P_{st}。

⑵对于每一条属于 P_{st} 的路径 p，且 p≠p_{st}1，假设 p 与 p_{st}1 不同的第一条边是(i,j)，那么在新图中，就存在一条路径{s,(s,1),1=v_1,e_1,v_2,e_2… i,(i,j’),j’… v’$_h$=n’,(n’,t’),t’}，所以 p∈$P_{st'}$。

⑶由于对于每条在 p_{st}1 中的边(i,j)，均不存在(i,j’)，所以 p_{st}1 不属于从 s 到 t’的路径集合。

由以上三点可知，从 s 到 t’的路径集合就是 P_{st}-{p_{st}1}。（证毕）

因此，在新图中求得的从 s 到 t’的最短路，就对应原图的第二短路径 p_{st}2。

重复以上过程，就能依次求得前 k 短路径。

但是，并非所有的节点都是需要复制的。设 v_k 是 p_{st}1 中第一个入度大于 1 的节点，那么需要复制的节点只有路径上 v_k 以及 v_k 后以及的所有节点，即{v_k, v_{k+1},… ,v_h,t}。这是因为对于不在路径上的节点，它们本身与复制的节点距离标号和连边的情况完全相同，而对于路径上 v_k 前的节点的复制的节点，没有从 s 出发并到达这些复制节点的路径。

因为新图只是在原图基础上增加节点和边，对原图的最短路径树没有影响，所以求最短路时，只需要对复制的节点在原有的最短路径树上维护新的最短路径树即可，并不需要重新对整张图求最短路。

算法流程如下：

```
Algorithm-deletion path version：
begin
  determine a shortest path tree in(V,E)
  let p be the shortest path
  now←1
  while now<k do
    begin
      determine v_i,the fist node of p with more than a single incoming arc
      V←V+{v_i,v_{i+1}… v_h,t^(now-1)'}
      for each (x,v_i)∈E and (x,v_i) not in p
        begin
          E←E-{(x,v_i)}+{(x,v'_i)}
          cost(x,v'_i)←cost(x,v_i)
          if x∈{v_i,v_{i+1}… v_h,t^(now-1)'} then
            begin
              E←E+{(x',v'_i)}
              cost(x',v'_i)←cost(x,v_i)
            end
        end
      calculate the distance lable dis(v'_i)←min{dis(x)+cost(x,v'_i)|(x,v'_i)∈E}
      for v_j←v_{i+1}… v_h,t^(now-1)'
        begin
          E←E-{(v_{j-1},v_j)}+{(v'_{j-1},v'_j)}
          cost(v'_{j-1},v'_j)←cost(v_{j-1},v_j)
          for each (x,v_j)∈E and (x,v_j) not in p
            begin
              E←E-{(x,v_j)}+{(x,v'_j)}
              cost(x,v'_j)←cost(x,v'_j)
              if x∈{v_i,v_{i+1}… v_h,t^(now-1)'} then
                begin
                  E←E+{(x',v'_i)}
                  cost(x',v'_i)←cost(x,v_i)
                end
            end
          calculate the distance lable dis(v'_j)←min{dis(x)+cost(x,v'_j)
                                                    |(x,v'_j)∈E}
        end
        let p be the shortest path
        now←now+1
    end
end
```

由于边只在求最短路径树（距离标号）时起作用，且每次新增加的边跟原图的边集相比，只有 $p_{st}1$ 中的边略有不同，所以可以每次直接对边集进行修改而不需要额外的空间储存新加入的边。而拓展的节点总共有 k 层。空间复杂度为 O(kn+m)。

假设采用 SPFA 求最短路径树，时间复杂度为 O(m)；每次改造图并枚举反向星更新最短路径树的时间复杂度为 O(m)，总的时间复杂度为 O(km)。

2.4 小结

我们回顾一下例题解法中对分层思想的利用：例题一中由于钥匙状态对格子连通性有影响，通过分层表示钥匙的不同状态确定连通性，使问题转化成一个一般的最短路问题；例题二中因为无法确定主楼是否在栅栏内，通过分层表示主楼引出的射线交点个数的奇偶性；例题三中通过分层并修改某些边，删除了一条路径，使每层的路径集合不同。

实际上，将图进行分层是因为在同一层图上难以准确地表现出图在不同条件下的状况或图的其他因素。分层的图分别表示不同的条件，加强了图的性质，使得在分层图能够使用基本的最短路算法求解原来的复杂问题。虽然分层在某种程度上增大了问题的规模，但是如果能够充分利用分层的性质，就可以减少冗余，使之降低甚至接近原来的规模。

3. 应用分层思想优化算法

下面，作者将分析两个例题：bic 和 roads，简单介绍了应用分层思想优化算法，并且将分层图最短路和动态规划进行比较。

3.1 例题四 bic[4]

题目：

如今的道路密度越来越大，收费也越来越多，因此选择最佳路径是很现实的问题。城市的道路是双向的，每条道路有固定的旅行时间以及需要支付的费用。路径由连续的道路组成。总时间是各条道路旅行时间的和，总费用是各条道路所支付费用的总和。同样的出发地和目的地，如果路径 A 比路径 B 所需时间少且费用低，那么我们说路径 A 比路径 B 好。对于某条路径，如果没有其他路径比它好，那么该路径被称为最优双调路径。这样的路径可能不止一条，或者说根本不存在。

给出城市交通网的描述信息，起始点和终点城市，求最优双调路径的条数。城市数 n 不大于 100，道路数 m 不大于 300，每条道路的费用和时间都是不大于 100 的非负整数。

分析：

我们把城市看成节点，城市之间的道路看成边，那么本题与一般求最短路的问题相比，不同之处在于边上有费用、时间两个权值。

我们采用分层的思想：设 c 为边费用的最大值，显然最优双调路径的费用不会超过 nc。我们把图拆分成 nc+1 层，表示到达该层顶点的费用分别为 0 到 nc。相应地，每条边拆成 O(nc)条边，边的两个顶点的所在层的费用差表示费用，边

[4] Baltic Olympiad in Informatics 2002

的权值表示时间。那么所得到的图可以直接运用最短路算法求解了。

在得到的新图中，一共有 O(nc)层，每层有 n 个顶点，m 条边，则点数为 $O(n^2c)$，边数为 O(nmc)。

由于边权（时间）都是非负的，我们可以使用 Dijkstra 算法求最短路。又考虑到图是稀疏的，故采用二叉堆实现 Dijkstra 算法中的优先队列。

如果直接实现或者不同顶点在同一费用的临时标号用堆来维护，不同费用的堆又组成一个堆的话，那么时间复杂度为 $O(ncmlg(n^2c))$。

重新观察分层图，由于费用也是非负的，这意味着边只能从一个节点指向同一层或费用更大的层的节点。因此，费用高的层对费用低的层是没有影响的。所以，我们可以按费用从低到高的顺序对每一层求一次最短路，而非一次性对所有点求最短路。每一层求最短路的时间复杂度为 O(mlgn)，总的时间复杂度为 O(ncmlgn)。

3.2 例题五 roads[5]

题目：

n 个城市有单向道路连接。每条路有固定的长度和通行税(用硬币数表示)。Bob 有一定数量的硬币，从城市 1 出发，想要尽快到达城市 n。求在 Bob 能承受通行税的前提下，从 1 到 n 的最短路径。

Bob 拥有的硬币数 k 是不大于 10000 的非负整数，城市数 n 是不大于 100 的正整数，道路数 m 是不大于 10000 的正整数，每条道路的长度是不大于 100 的正整数，每条道路的通行税是不大于 100 的非负整数。

分析：

我们把城市和道路分别看成图的顶点和边。本题跟上一题有非常相似的形式，我们尝试着用同样的方法解决：把图拆分成 k+1 层，表示到达该层顶点所需的通行税分别为 0 到 k，相应地，每条边拆成 O(k)条边，边的两个顶点的所在层的通行税之差表示费用，边的权值表示道路长度。由于道路长度是正整数，我们可以采用 Dijkstra 算法求最短路。因为图是稠密的，所以优先队列直接使用一维数组。时间复杂度为 $O(k(kn^2+m))$。

又因为道路通行税是非负的，仿照上一题，按照通行税从低到高的顺序对每层求最短路。时间复杂度降为 $O(k(n^2+m))$。

由于题目已经给定 Bob 拥有的硬币数，所以我们很自然地直接以通行税的多少进行分层，但是我们忽略了一个条件：道路长度是正整数，而不仅是非负整数。

我们以道路长度进行分层，然后使用动态规划。令 f[i,j]表示到达城市 j 长度为 i 的所有路径所花费的最少硬币数。转移方程为：

$$\begin{cases} f[0,1]=0 \\ f[0,j]=\infty \quad (j=2\ldots n) \\ f[i,j]=\max\{f[i-len,j0]+fee\} \end{cases}$$

(城市 j0 到城市 j 有一条长度为 len，费用为 fee 的道路)

设每条道路长度的最大值为 L，那么时间复杂度为 O(nLm)，效率有所提高。

[5] Central European Olympiad in Informatics 1998

3.3 小结

结合例题分析和分层图的性质，我们总结一下利用分层思想优化的最短路问题。

分层图的层是我们构建模型时复制的，许多图的元素都是相同或相似的，我们不需要增加额外的空间或操作。另外，由于边的权值经常远小于图的层数，因此可以采用类似动态规划中的滚动数组的方法节省空间。

分层图中往往层与层之间拓扑有序，让我们可以逐层求最短路。这样细化问题的结果就是使得问题的规模有所下降（比如 Dijkstra 算法中的优先队列的规模，Bellman-Ford 算法或 SPFA 算法的迭代次数），从而提高算法效率。特别地，当同层的节点拓扑无关时（比如例题五的第二个模型），就可以使用动态规划代替最短路算法。相对于 Dijkstra 算法，动态规划节省了对优先队列的操作，提高了效率。

4. 总结

本文从建立模型和优化算法两个方面阐述了分层思想在最短路径问题中的应用。在建立模型的方面，需要把难以表现同一层图上的因素用分层图表示；在优化算法时，则需要充分的挖掘分层图的特殊性质，有针对性地利用分层性质优化算法。这些都对技巧和创造性有一定的要求。希望本文能对读者有所帮助。

感谢

感谢谢水英老师给我的指导与帮助

感谢刘汝佳教练提出宝贵的意见

参考文献

[1] 余远铭 《最短路算法及其应用》 2006 年国家集训队论文

[2] 肖天 《分层图思想及其在信息学竞赛中的应用》 2004 年国家集训队论文

[3] Baltic Olympiad in Informatics 2002/2007 official solutions

[4] E.Q.V. Martins and J.L.E. Santos. A new shortest paths ranking algorithm. Investigacao Operacional, 20:(1):47-62,2000

附录

附录一：拯救大兵瑞恩 原题

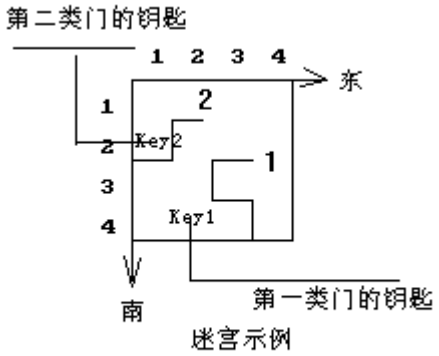
(RESCUE.EXE)

1944 年，特种兵麦克接到国防部的命令，要求立即赶赴太平洋上的一个孤岛，营救被敌军俘虏的大兵瑞恩。瑞恩被关押在一个迷宫里，迷宫地形复杂，但是幸好麦克得到了迷宫的地形图。

迷宫的外形是一个长方形，其在南北方向被划分为 N 行，在东西方向被划分为 M 列，于是整个迷宫被划分为 N*M 个单元。我们用一个有序数对（单元的行号，单元的列号）来表示单元位置。南北或东西方向相邻的两个单元之间可以互通，或者存在一扇锁着的门，又或者存在一堵不可逾越的墙。迷宫中有一些单元存放着钥匙，并且所有的门被分为 P 类，打开同一类的门的钥匙相同，打开不同类的门的钥匙不同。

大兵瑞恩被关押在迷宫的东南角，即（N,M）单元里，并已经昏迷。迷宫只有一个入口，在西北角，也就是说，麦克可以直接进入(1,1)单元。另外，麦克从一个单元移动到另一个相邻单元的时间为 1，拿取所在单元的钥匙的时间以及用钥匙开门的时间忽略不计。

你的任务是帮助麦克以最快的方式抵达瑞恩所在单元，营救大兵瑞恩。



迷宫示例

输入：

第一行是三个整数，依次表示 N,M,P 的值；

第二行是一个整数 K，表示迷宫中门和墙的总个数；

第 I+2 行（1<=I<=K），有 5 个整数，依次为 Xi1,Yi1,Xi2,Yi2,Gi：

当 Gi>=1 时，表示(Xi1,Yi1)单元与(Xi2,Yi2)单元之间有一扇第 Gi 类的门，当 Gi=0 时，表示(Xi1,Yi1)单元与(Xi2,Yi2)单元之间有一堵不可逾越的墙；

（其中，|Xi1-Xi2|+|Yi1-Yi2|=1，0<=Gi<=P）

第 K+3 行是一个整数 S，表示迷宫中存放的钥匙总数；

第 K+3+J 行(1<=J<=S)，有 3 个整数，依次为 Xi1,Yi1,Qi：表示第 J 把钥匙存放在(Xi1,Yi1)单元里，并且第 J 把钥匙是用来开启第 Qi 类门的。（其中 1<=Qi<=P）

注意：输入数据中同一行各相邻整数之间用一个空格分隔。

输出：

输出文件只包含一个整数 T，表示麦克营救到大兵瑞恩的最短时间的值，若不存在可行的营救方案则输出-1。

输入输出示例：

输入文件
4 4 9
9
1 2 1 3 2
1 2 2 2 0
2 1 2 2 0
2 1 3 1 0
2 3 3 3 0
2 4 3 4 1
3 2 3 3 0
3 3 4 3 0
4 3 4 4 0
2
2 1 2
4 2 1
输出文件
14
参数设定：
3<=N,M<=15;
1<=P<=10;

附录二：fence 原题

Building a Fence

Leopold is indeed a lucky fellow. He just won a huge estate in the lottery. The estate contains several grand buildings in addition to the main mansion, in which he intends to live from now on.However, the estate lacks a fence protecting the premises from trespassers, which concerns Leopold to a great extent. He wants to build a fence and, in order to save money, he decides it is sufficient to have a fence that encloses the main mansion, except for one important restriction: the fence must not lie too close to any of the buildings. To be precise, seen from above, each building is enclosed in a surrounding forbidden rectangle within which no part of the fence may lie. The rectangles' sides are parallel to the x- and y-axis. Each part of the fence must also be parallel either to the x-axis or the y-axis.
Help Leopold to compute the minimum length of any allowed fence enclosing the main mansion.

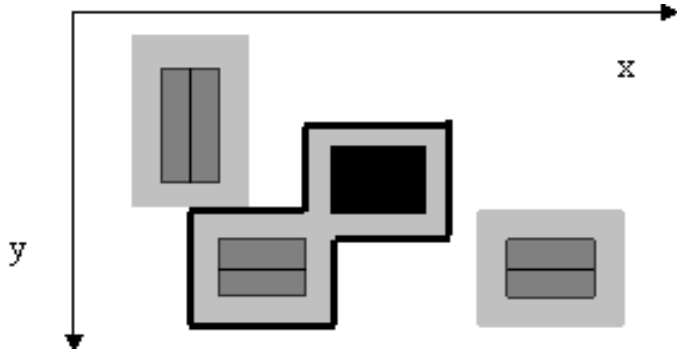


Figure 1: The main mansion (black) and three other buildings with surrounding forbidden

rectangles. The thick black line shows a shortest allowed fence enclosing the main mansion.

Input

The input is read from a text file named fence.in. The first line of the input file contains a positive integer m ($1 \le m \le 100$), the number of buildings of the estate. Then follow m lines each describing a forbidden rectangle enclosing a building. Each row contains four space-separated integers tx, ty,bx, and by, where (tx, ty) are the coordinates of the upper left corner and (bx, by) the coordinatesof the bottom right corner of the rectangle. All coordinates obey $0 \le tx < bx \le 10{,}000$ and$0 \le ty < by \le 10{,}000$. The first rectangle is the forbidden rectangle enclosing the main mansion.

Output

The output is written into a text file named fence.out. It contains one line with a single positive integer equal to the minimum length of any allowed fence enclosing the main mansion.

Example

Fence.in	fence.out
4 8 4 13 8 2 1 6 7 4 7 9 11 14 7 19 11	32

Grading

In 30% of the testcases $m \le 10$ holds.

附录三：cow relay 原题

Cow Relays [Brian Dean, 2001]

Farmer John has formed a relay team for a race by choosing K (2 <= K <= 40) of his cows. The race is run on FJ's farm which has N (4 <= N < 800) fields numbered 1..N and M (1 <= M <= 4000) unique bidirectional pathways that connect pairs of different fields. You will be given the time it takes a cow to traverse each pathway.

The first cow begins the race in field #1 and runs to the finish line in field #N. As soon as the first cow finishes, the next cow then starts from field #1 and runs to field #N and so on. For this race, no two cows can follow precisely the same route (a route is a sequence of fields).

Write a program which computes the minimum possible time required for FJ's relay team. It is guaranteed that some minimum possible time exists. Any cows can revisit a path in her trip to the other barn if that turns out to be required for a "best" solution. As soon as a cow enters field #N, her relay leg is finished.

PROBLEM NAME: relay

INPUT FORMAT:

Line 1: One line with three integers: K, N, and M

Lines 2..M+1: Each line contains three integers describing a path: the starting field, the ending field, and the integer time to traverse the path (in the range 1..9500).

SAMPLE INPUT (file relay.in):

```
4 5 8
1 2 1
1 3 2
1 4 2
2 3 2
2 5 3
3 4 3
3 5 4
4 5 6
```

OUTPUT FORMAT:

One line with a single integer that is the minimum possible time to run a relay.

SAMPLE OUTPUT (file relay.out):

```
23
[Namely: Cow 1: 1->2->5          4
         Cow 2: 1->3->5          6
         Cow 3: 1->2->1->2->5    6
         Cow 4: 1->2->3->5       7]
```

附录四：bic 原题

Bicriterial routing

The network of pay highways in Byteland is growing up very rapidly. It has become so dense, that the choice of the best route is a real problem. The network of highways consists of bidirectional roads connecting cities. Each such road is characterized by the traveling time and the toll to be paid.

The route is composed of consecutive roads to be traveled. The total time needed to travel the route is equal to the sum of traveling times of the roads constituting the route. The total fee for the route is equal to the sum of tolls for the roads of which the route consists. The faster one can travel the route and the lower the fee, the better the route. Strictly speaking, one route is better than the other if one can travel it faster and does not have to pay more, or vice versa: one can pay less and can travel it not slower than the other one. We will call a route connecting two cities minimal if there is no better route connecting these cities. Unfortunately, not always exists one minimal route – there can be several incomparable routes or there can be no route at all.

Example

The picture below presents an example network of highways. Each road is accompanied by a pair of numbers: the toll and the traveling time.

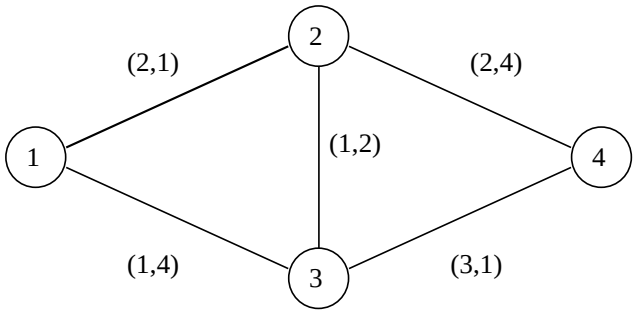
Let us consider four different routes from city 1 to city 4, together with their fees and traveling times: 1-2-4(fee4, time5), 1-3-4(fee4, time5), 1-2-3-4(fee6, time4) and 1-3-2-4(fee4, time10).

Routes 1-3-4 and 1-2-4 are better than 1-3-2-4. There are two minimal pairs fee time: fee4, time5 (roots 1-2-4 and 1-3-4) and fee6, time4 (root 1-2-3-4). When choosing the route we have to decide whether we prefer to travel faster but we must pay more (route 1-2-3-4), or we would rather travel slower but cheaper (route 1-3-4 or 1-2-4).

Task

Your task is to write a program, which:

Reads the description of the highway network and starting and ending cities of the route form the text file bic.in.



Computes the number of different minimal routes connecting the starting and ending city, however all the routes characterized by the same fee and traveling time count as just one route; we are interested just in the number of different minimal pairs fee time.

Writes the result to the output file bic.out.

Input data

There are four integers, separated by single spaces, in the first line of the text file bic.in: number of cities n (they are numbered from 1 to n), $1 \le n \le 100$, number of roads m, $0 \le m \le 300$, starting city s and ending city e of the route, $1 \le s,e \le n$, $s \ne e$. The consecutive m lines describe the roads, one road per line. Each of these lines contains four integers separated by single spaces: two ends of a road p and r, $1 \le p,r \le n$, $p \ne r$, the toll c, $0 \le c \le 100$, and the traveling time t, $0 \le t \le 100$. Two cities can be connected by more than one road.

Output data

Your program should write one integer, number of different minimal pairs fee time for routes from s to e, in the first and only line of the text file bic.out.

Bic.in	**bic.out**	**Comments**
4 5 1 4 2 1 2 1 3 4 3 1 2 3 1 2 3 1 1 4 2 4 2 4	2	This is the case shown on the picture above.

附录五：roads 原题

Problem description

N cities named with numbers 1 ... N are connected with one-way roads. Each road has two parameters associated with it: the road length and the toll that needs to be paid for the road (expressed in the number of coins).

Bob and Alice used to live in the city 1. After noticing that Alice was cheating in the card game they liked to play, Bob broke up with her and decided to move away - to the city N. He wants to get there as quickly as possible, but he is short on cash.

We want to help Bob to find **the shortest path** from the city 1 to the city N **that he can afford** with the amount of money he has.

Input data

The first line of the input file **ROADS.IN** contains the integer K, $0 \le K \le 10000$, maximum number of coins that Bob can spend on his way.

The second line contains the integer N, $2 \le N \le 100$, the total number of cities.

The third line contains the integer R, $1 \le R \le 10000$, the total number of roads.

Each of the following R lines describes one road by specifying integers S, D, L and T separated by single blank characters :

- S is the source city, $1 \le S \le N$
- D is the destination city, $1 \le D \le N$
- L is the road length, $1 \le L \le 100$
- T is the toll (expressed in the number of coins), $0 \le T \le 100$

Notice that different roads may have the same source and destination cities.

Output data

The first and the only line of the output file **ROADS.OUT** should contain the total length of the shortest path from the city 1 to the city N whose total toll is less than or equal K coins.

If such path does not exist, only number -1 should be written to the output file.

Examples

ROADS.IN

```
5
6
7
1 2 2 3
2 4 3 3
3 4 2 4
```

```
1 3 4 1
4 6 2 1
3 5 2 0
5 4 3 2
```

ROADS.OUT

```
11
```

ROADS.IN

```
0
4
4
1 4 5 2
1 2 1 0
2 3 1 1
3 4 1 0
```

ROADS.OUT

-1