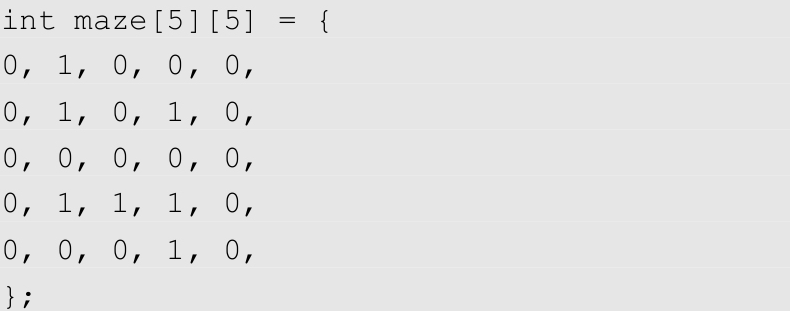
**POJ3984**

**题目描述（POJ3984）：**用一个二维数组表示一个迷宫，其中1表示墙壁，0表示可以走的路，只能横着走或竖着走，不能斜着走，编写程序，找出从左上角到右下角的最短路线。



**输入：**一个5×5的二维数组，表示一个迷宫。数据保证有唯一解。

**输出：**从左上角到右下角的最短路径，格式如以下输出样例所示。



**题解：**本题为典型的迷宫问题，可以使用广度优先搜索解决。定义方向数组dir[4][2]= {{1,0},{-1,0},{0,1},{0,-1}}，定义前驱数组pre[][]记录经过的节点。

**1. 算法设计**

（1）定义一个队列，将起点(0, 0)入队，标记已走过。

（2）如果队列不空，则队头出队。

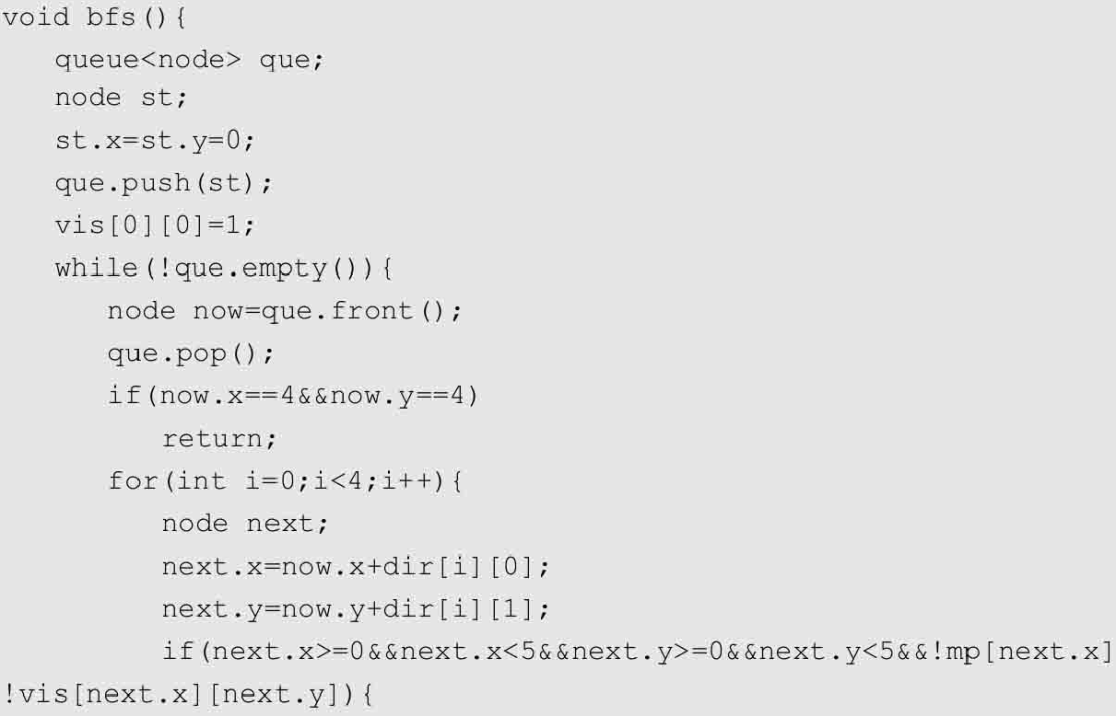
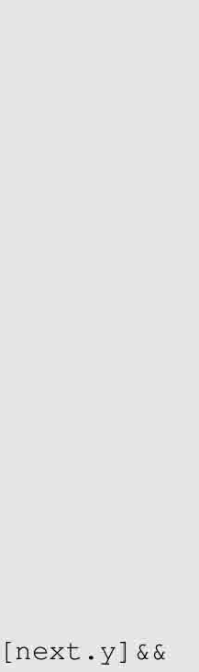
（3）如果队头正好是目标(4, 4)，则退出。

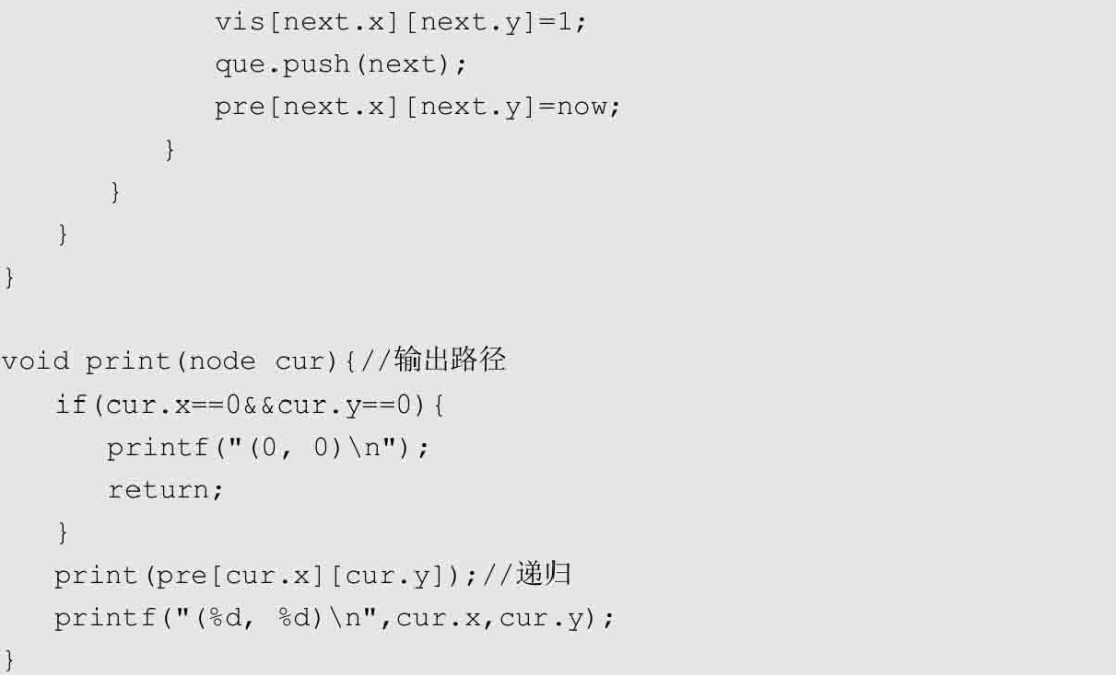
（4）沿着4个方向搜索，如果该节点未出边界、未走过且不是墙壁，则标记走过并入队，用前驱数组记录该节点。

（5）转向步骤2。

（6）根据前驱数组输出从起点到终点的最短路径。

**2. 算法实现**

****

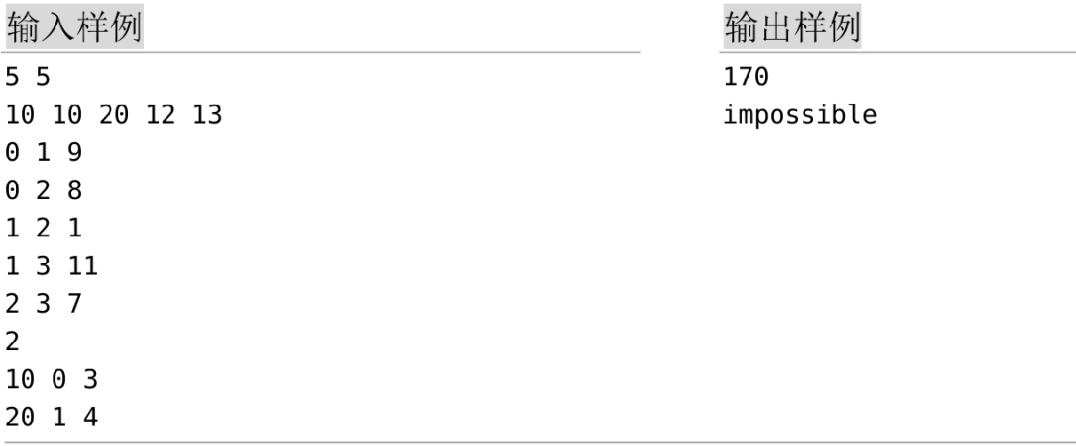
****

**POJ3635**

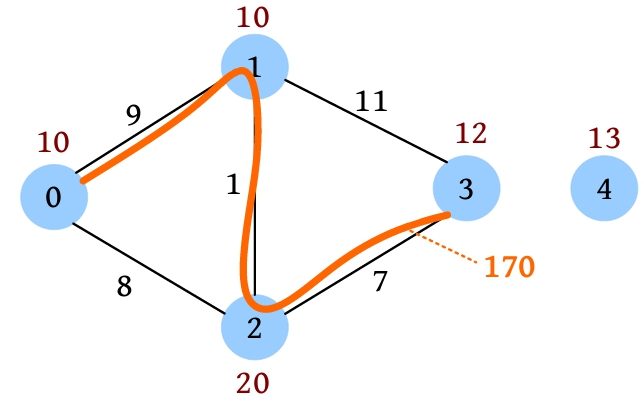
**题目描述（POJ3635）：**城市之间的油价是不一样的，编写程序，寻找最便宜的城市间旅行方式。在旅行途中可以加满油箱。假设汽车每单位距离使用一单位燃料，从一个空油箱开始。

**输入：**输入的第1行包含n（1≤n≤1000）和m（0≤m≤10 000），表示城市和道路的数量。下一行包含n个整数pi（1≤pi≤100），其中pi表示第i个城市的燃油价格。接下来的m行，每行都包含3个整数u、v（0≤u, v<n）和d（1≤d≤100），表示在u和v之间有一条路，长度为d。接下来一行是查询数q（1≤q≤100）。再接下来的q行，每行都包含3个整数c（1≤c≤100）、s和e，其中c是车辆的油箱容量，s是起点城市，e是终点城市。

**输出：**对于每个查询，都输出给定容量的汽车从s到e的最便宜旅程的价格，如果无法从s到e，则输出“impossible”。



**题解：**本题为加油站加油问题。给定n个节点、m条边，每走1单位的路径都会花费1单位的油量，并且不同的加油站价格是不同的。现在有一些询问，每一个询问都包括起点、终点及油箱的容量，求从起点到达终点所需的最少花费。可以采用优先队列分支限界法搜索。



涉及两个维度的图最短路径，一个是费用，一个是地点。可以把当前节点对应的油量抽象成多个节点（例如在位置0有1升油是一个节点，在位置0有2升油是一个节点），把费用看作边，那么最少费用就可以类似Dijsktra算法那样不断地加入节点。于是得到一个扩展节点的策略：每次都加1升油；如果依靠加的油足够走到下一个节点，就走到下一个节点（减去路上消耗的油，花费不变）；在广度优先搜索中将所有可扩展的节点都加入优先队列中，如果到达终点，则返回花费。

**1. 算法设计**

（1）定义一个优先队列，将起点及当前油量、花费作为一个节点(st,0,0)入队。

（2）如果队列不空，则队头(u,vol,cost)出队，并标记该节点油量已扩展，vis[u][vol]=1。

（3）如果u正好是目标ed，则返回花费cost。

（4）如果当前油量小于油箱容量，且u的油量vol+1未扩展，则将该节点(u,vol+1, cost+price[u])入队。

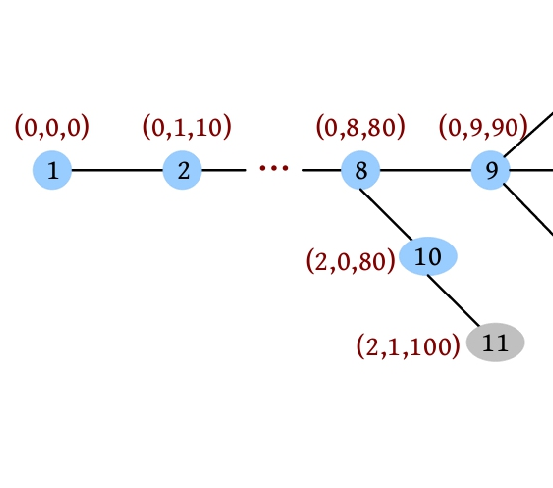
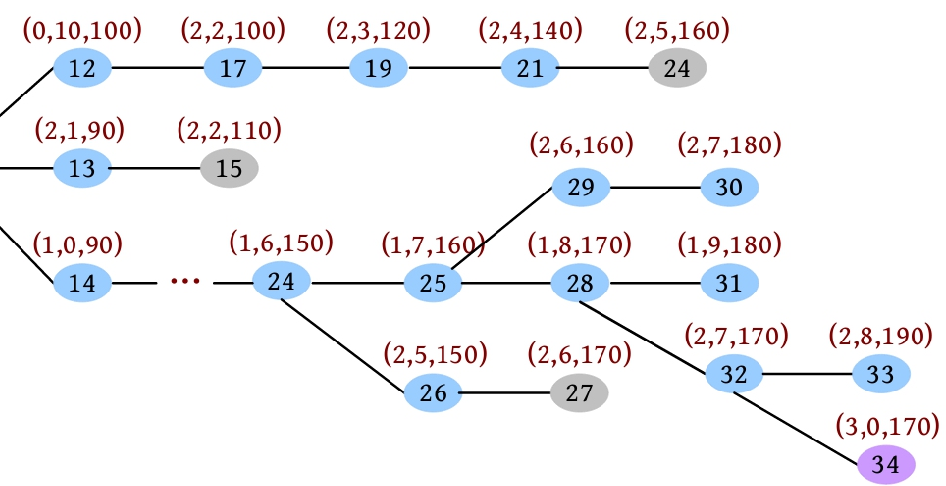
（5）检查u所有邻接点v，如果当前油量大于或等于边权w，且v节点的油量vol-w未扩展，则将该节点(v,vol-w,cost) 入队。

（6）转向步骤2。

**2. 算法优化**

用一个数组dp[i][j]表示在节点i、当前油量为j时的最小花费。在当前节点及油量对应的花费更小时才生成节点，生成的节点会少很多，但由于系统数据量不大，运行时间差不多。

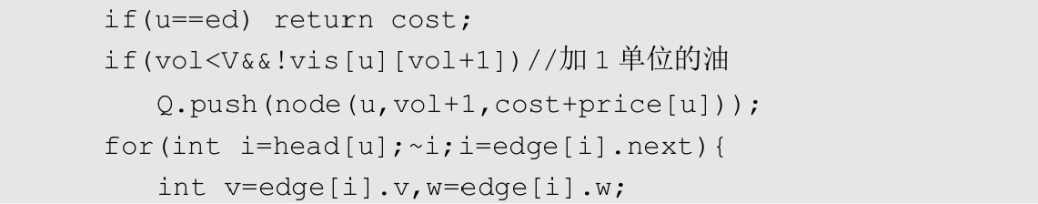
采用优化后的算法生成节点的过程如下图所示。

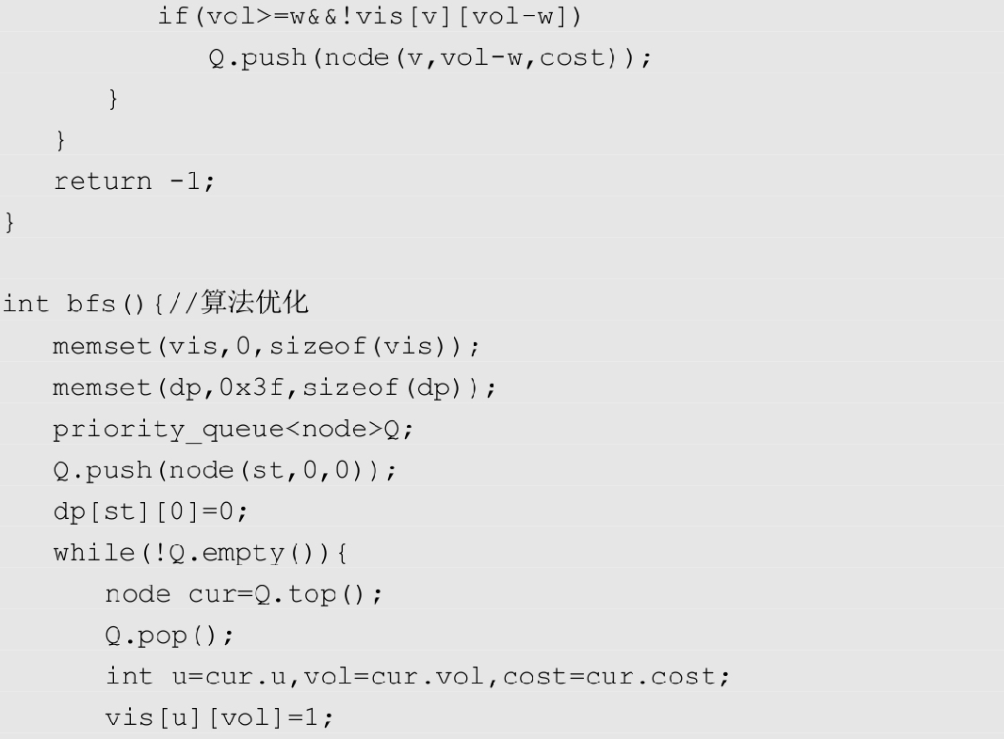


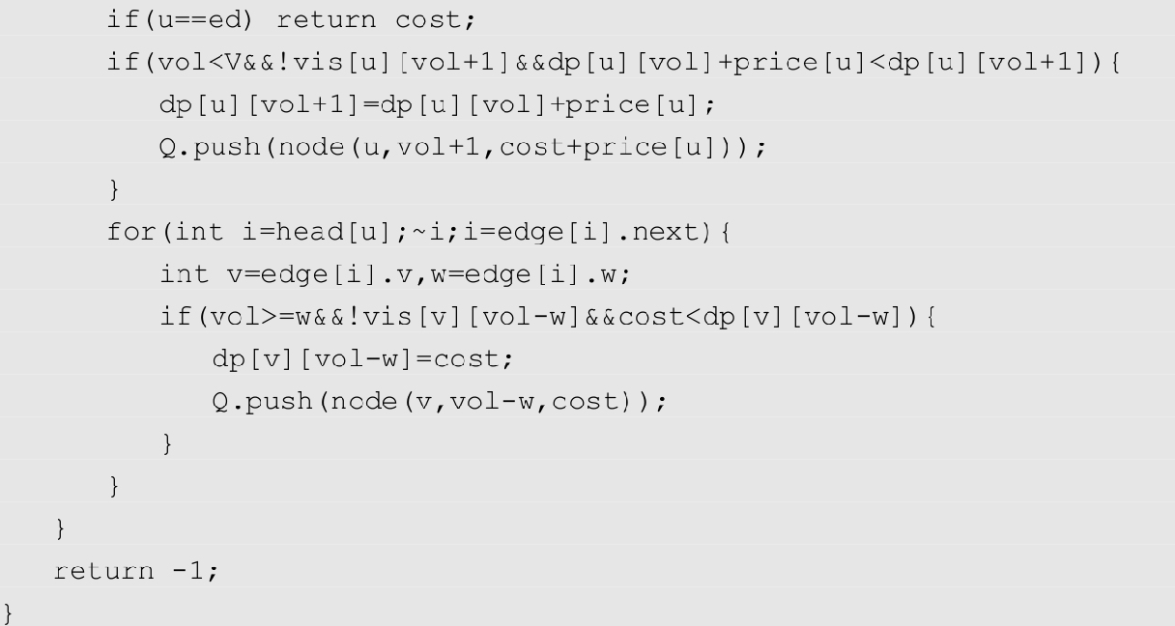
**3. 算法实现**

****

****

****

****

****