最短路

Floyd：

Floyd 算法又称为插点法，是一种用于寻找给定的加权图中多源点之间最短路径的算法。该算法名称以创始人之一、1978 年图灵奖获得者、斯坦福大学计算机科学系教授罗伯特·弗洛伊德命名。

其状态转移方程如下：

map[i,j]:=min{map[i,k]+map[k,j],map[i,j]}；

map[i,j]表示 i 到 j 的最短距离，K 是穷举 i,j 的断点，map[n,n]初值应该为 0，或者按照题目意思来做。

当然，如果这条路没有通的话，还必须特殊处理，比如没有 map[i,k]这条路。

要注意的是，枚举 K 的循环应放在最外层。

练习：

Poj1125、poj3615

Dijkstra：

Dijkstra(迪杰斯特拉)算法是典型的单源最短路径算法，用于计算一个节点到其他所有节点的最短路径。主要特点是以起始点为中心向外层层扩展，直到扩展到终点为止。

算法描述：将图所有点的集合 S 分为两部分，V 和 S-V。V 集合是已经得到最短路径的点的集合，在初始情况下 V 中只有一个顶点 t，S-V 是还未得到最短路径点的集合。然后，在每一次迭代过程中取得 S-V 集中到 V 集合任一点距离最短的点，将其加到 V 集合，从 V-S 集合删除。重复此过程直到 S-V 集合为空。

算法示例:

```
for (i=1;i<n;i++)
{   ans=1000000000;
    for  (j=1;j<=n;j++)  if  ((f[j])&&(dist[j]<ans))  ans=dist[j],st=j;
f[st]=false;
    for    (j=1;j<=n;j++)    if    ((f[j])&&(dist[j]>dist[st]+g[st][j]))
dist[j]=dist[st]+g[st][j];
    }
```

练习：

hdu2544

hdu1874

hdu2066

数组模拟链表记录图的关系：

讲到图不能避开的技巧。以点 i 为例，设以点 i 为原点延伸出去

m 条边，建立一个以点 i 为初始点，长 m 的链表，链表存储着这 m 条边。

对边的读入：

如果当前读入一条点 u 到点 v 的边：

```
inline void Add(int u,int v)
{ next[++l]=head[u]; head[u]=l; go[l]=v; }
```

当调用某个点延伸出去的边时：

```
for (j=head[k];j;j=next[j]) {}
```

这里 j 就表示边 j。

SPFA：

几乎所有的最短路算法其步骤都可以分为两步

1.初始化

2.松弛操作

初始化： d 数组全部赋值为 INF（无穷大）；p 数组全部赋值为 s（即源点），或者赋值为-1，表示还没有知道前驱

然后 d[s]=0; 表示源点不用求最短路径，或者说最短路就是 0。将源点入队；

（另外记住在整个算法中有顶点入队了要记得标记 vis 数组，有顶点出队了记得消除那个标记）

队列+松弛操作

读取队头顶点u，并将队头顶点u出队（记得消除标记）；将与点u相连的所有点v进行松弛操作，如果能更新估计值（即令d[v]变小），那么就更新，另外，如果点v没有在队列中，那么要将点v入队（记得标记），如果已经在队列中了，那么就不用入队

以此循环，直到队空为止就完成了单源最短路的求解

SPFA可以处理负权边

定理: 只要最短路径存在，上述SPFA算法必定能求出最小值。

证明：

每次将点放入队尾，都是经过松弛操作达到的。换言之，每次的优化将会有某个点v的最短路径估计值d[v]变小。所以算法的执行会使d越来越小。由于我们假定图中不存在负权回路，所以每个结点都有最短路径值。因此，算法不会无限执行下去，随着d值的逐渐变小，直到到达最短路径值时，算法结束，这时的最短路径估计值就是对应结点的最短路径值。（证毕）

期望的时间复杂度O(ke)， 其中k为所有顶点进队的平均次数，可以证明k一般小于等于2。

判断有无负环：

如果某个点进入队列的次数超过N次则存在负环（SPFA无法处理带负环的图）

算法示例：

```
inline void spfa()
{   int l,r,i,j;
    b[l=r=1]=1; memset(f,false,sizeof(f)); f[1]=true;
    for (i=2;i<=n;i++) g[i]=2147483647; g[1]=0;
    while (l<=r)
    {   for (j=head[b[l]];j;j=next[j])
          if (g[go[j]]>g[b[l]]+cost[j])
          { g[go[j]]=g[b[l]]+cost[j];
            if (!f[go[j]]) b[++r]=go[j],f[go[j]]=true;
            }
        f[b[l]]=false; l++;
        }   for (i=1;i<=n;i++) ans+=g[i];
}
```

练习：

poj1511

图遍历

Dfs（深度优先遍历）：

深度优先搜索遍历类似于树的先序遍历。假定给定图 G 的初态是所有顶点均未被访问过，在 G 中任选一个顶点 i 作为遍历的初

始点，则深度优先搜索递归调用包含以下操作：

（1）访问搜索到的未被访问的邻接点；

（2）将此顶点的 visited 数组元素值置 1；

（3）搜索该顶点的未被访问的邻接点，若该邻接点存在，则从此邻接点开始进行同样的访问和搜索。

深度优先搜索 DFS 可描述为：

（1）访问 v0 顶点；

（2）置 visited[v0]=1；

（3）搜索 v0 未被访问的邻接点 w，若存在邻接点 w，则 DFS(w)。

Bfs（宽度优先遍历）：

广度优先搜索遍历类似于树的按层次遍历。

对于无向连通图，广度优先搜索是从图的某个顶点 v0 出发，在访问 v0 之后，依次搜索访问 v0 的各个未被访问过的邻接点 w1，w2，…。然后顺序搜索访问 w1 的各未被访问过的邻接点，w2 的各未被访问过的邻接点，…。即从 v0 开始，由近至远，按层次依次访问与 v0 有路径相通且路径长度分别为 1，2，…的顶点，直至连通图中所有顶点都被访问一次。

广度优先搜索的顺序不是唯一的。

具体描述如下：

设图 G 的初态是所有顶点均未访问，在 G 中任选一顶点 i 作为

初始点，则广度优先搜索的基本思想是：

（1）从图中的某个顶点 V 出发，访问之；并将其访问标志置为已被访问，即 visited[i]=1；

（2）依次访问顶点 V 的各个未被访问过的邻接 点，将 V 的全部邻接点都访问到；

（3）分别从这些邻接点出发，依次访问它们的未被访问过的邻接点，并使“先被访问的顶 点的邻接点”先于“后被访问的顶点的邻接点”被访问，直到图中所有已被访问过的顶 点的邻接点都被访问到。

依此类推，直到图中所有顶点都被访问完为止 。

广度优先搜索在搜索访问一层时，需要记住已被访问的顶点，以便在访问下层顶点时，从已被访问的顶点出发搜索访问其邻接点。所以在广度优先搜索中需要设置一个队列 Queue，使已被访问的顶点顺序由队尾进入队列。在搜索访问下层顶点时，先从队首取出一个已被访问的上层顶点，再从该顶点出发搜索访问它的各个邻接点。

练习：

Poj3083

*差分约束

差分约束问题是给出一些形如 x-y<=b 不等式的约束，问你是否满足有解的问题。这类问题可以转换成图论里的最短路径问题。

求解差分约束系统，可以转化成图论的单源最短路径问题。观察 xj-xi<=bk，会发现它类似最短路中的三角不等式 d[v] <= d[u] +w[u,v]，即 d[v] - d[u] <=w[u,v]。因此，以每个变量 xi 为结点，对于约束条件 xj-xi<=bk，连接一条边 E(i,j），边权为 bk。求单源最短路径，必须有一个源点，然后再求这个源点到其他所有点的最短路径，我们可以增加一个原点 s 与所有其他点相连，边权均为 0，xi - x0 <= 0。

图中任意一条最短路径既不能包含负权回路，也不会包含正权回路，所以最多包含 n-1 条边，从源点 s 出发每进行一遍松弛操作时，多生成了了从 s 出发层次为 1 的树，而最短边路径最多为 n-1，故只需要循环 n-1 次。最后计算的 d[v]即为差分不等式的一组解。

注意点:

1. 如果要求最大值想办法把每个不等式变为标准 x-y<=k 的形式,然后建立一条从 y 到 x 权值为 k 的边,变得时候注意 x-y<k =>x-y<=k-1

如果要求最小值的话,变为 x-y>=k 的标准形式，然后建立一条从 y 到 x 的 k 边，求出最长路径即可

2.如果权值为正，用 dijkstra，spfa，bellman 都可以，如果为负不能用 dj，并且需要判断是否有负环，有的话就不存在

Intervals

Time Limit: 2000MS **Memory Limit:** 65536K
Total Submissions: 22979 **Accepted:** 8671

Description

You are given n closed, integer intervals [ai, bi] and n integers c1, ..., cn.
Write a program that:
reads the number of intervals, their end points and integers c1, ..., cn from the standard input,
computes the minimal size of a set Z of integers which has at least ci common elements with interval [ai, bi], for each i=1,2,...,n,
writes the answer to the standard output.

Input

The first line of the input contains an integer n (1 <= n <= 50000) -- the number of intervals. The following n lines describe the intervals. The (i+1)-th line of the input contains three integers ai, bi and ci separated by single spaces and such that 0 <= ai <= bi <= 50000 and 1 <= ci <= bi - ai+1.

Output

The output contains exactly one integer equal to the minimal size of set Z sharing at least ci elements with interval [ai, bi], for each i=1,2,...,n.

Sample Input

```
5
3 7 3
8 10 3
6 8 1
1 3 1
10 11 1
```

Sample Output

```
6
```

解析：

设 x[i]是{i}这个集合跟所求未知集合的交集元素个数，明显最大只能是 1 ，再设 s[i] = x[0] + x[1] + …… + x[i] 明显的，s[i]表示集合{0,1,2,3,……,i}与所求未知集合的交集元素个数

那么就有 x[i] = s[i] - s[i-1]

∵0 <= x[i] <= 1 ∴0 <= s[i] - s[i-1] <= 1

由于题目求最小值，所以是最长路，用的是 a - b >= c 这种形式即有：

①s[i]-s[i-1]>=0; ②s[i-1]-s[i]>=-1;

按照题目输入 a, b, c：

表示{a,a+1,a+2,……,b}(设这个集合是 Q)与所求未知集合的交集元素个数至少为 c 。

而 s[a-1]表示{1,2,3,……,a-1}与所求未知集合的交集元素个数 s[b]表示{1,2,3,……,a-1,a,a+1,a+2,……,b}与所求未知集合的交集

元素个数。

∴Q=s[b]-s[a-1]；即可建立关系:

③s[b]-s[a-1]>=c;

但是还有一个问题 a >= 0，那么 a-1 有可能不合法 。

解决方法：所有元素+1 就可以了。实现：把③变成

s[b+1]-s[a]>=c;

练习另给文件。