最小生成树

PKU Hzyoi

01 最小生成树

最小生成树

- 无向连通图G的某一子图T是一棵树且若覆盖G中所有的顶点，则称作G的一棵生成树。
- 若图G为一张带权图，则每一棵生成树的权值即为其所采用各边边权的和。
- 在G所有的生成树中，权值最小的生成树称为最小生成树。
- 假设所有边权均不相同（相同可以设第二关键字），最小生成树有以下性质：
- 切割性质：
  - 设S为非空集的V的真子集，边$e = (u, v)$是$u \in S$且$v \in V \setminus S$的所有边中权值最小的一个。
  - 则图G的所有最小生成树均包含e。
- 回路性质:
  - 设C是图G中的任意回路，边e是C上权值最大的边。
  - 则图G的所有生成树均不包含e。

Prim

- 思路和Dijkstra差不多，代码也非常像。
- 直接上步骤：
- 1.输入：一个加权连通图，其中顶点集合为V，边集合为E；
- 2.初始化：V_{new} ={x}，其中x为集合V中的任一节点（起始点），E_{new}={}；
- 3.重复下列操作，直到V_{new}=V：

  a.在集合E中选取权值最小的边<u, v>，其中u为集合V_{new}中的元素，而v不在V_{new}集合当中，并且v∈V（如果存在有多条权值相等且最小的边，则可任意选取其中之一）；

  b.将v加入集合V_{new}中，将<u, v>边加入集合E_{new}中；
- 4.输出：使用集合V_{new}和E_{new}来描述所得到的最小生成树。
- 不加优化的Prim时间复杂度为O（$|V|^2$）

- 简单证明Prim算法
- 我们使用反证法：假设prim生成的不是最小生成树
- 1.设Prim算法生成的树为G_0
- 2.假设存在G_{min}使得$W(G_{min})<W(G_0)$，则在G_{min}中存在$<u,v>$不属于G_0
- 3.将$<u,v>$加入G_0中可得一个环，且$<u,v>$不是该环的最长边（这是因为$<u,v>\in G_{min}$）。
- 4.而如果这样的话，说明Prim算法选了更长的那一条边，这与Prim算法每次生成最短边矛盾。
- 5.故假设不成立，命题得证.

```
int Prim(int s){
    int Ans=0;
    memset(vis,0,sizeof vis);
    Rep(i,1,n) dist[i]=len[s][i];
    vis[s]=1;
    Rep(i,1,n-1){
        int v,Min=1<<30;
        Rep(j,1,n) if (!vis[j]&&dist[j]<Min)
            Min=dist[j],v=j;
        vis[v]=1;Ans+=dist[v];
        Rep(j,1,n) if (!vis[j])
            dist[j]=min(dist[j],len[v][j]);
    }
    return Ans;
}
```

原题忘了

- 小明决定修建n个饮水点。第i个位于(xi,yi)
- 对于每一个饮水点i可以通过花费w[i]元来打井，或者从其他饮水口j花费$\sqrt{(xi-xj)^2+(yi-yj)^2}$元修建一条独立的管道运输而来。请你计算出在所有饮水点都能喝到水的最小花费。

- 视所有地表以下的水为一个点，那么题目就是要求所有个点连通起来的最小花费，求最小生成树即可。
- 这个题目边数是满的n^2所以用prim会更加优秀

Kruscal

- 这也是一种基于贪心的算法。
- 也是直接上步骤:
- 1.记图G中顶集为V，边集为E;
- 2.新建图G_{new}，G_{new}的V_{new}=V，但是E_{new}={};
- 3.将E中的边按权值从小到大排序;
- 4.从权值最小的边开始遍历每条边，直至图G中所有的节点都在同一个连通块中。在遍历边的过程中，如果这条边连接的两个顶点在图G_{new}中不在同一个连通块中，那么就添加这条边到图G_{new}中。
- 连通块由并查集维护。时间复杂度为O（|E|*log|E|+α|V|）

- 证明：显然。
- 只有1个点的图即1阶图是能找到最小生成树的。
- 假设Kruskal算法对n≤k阶图适用，那么，在k+1阶图G中，我们把最短边的两个顶点a和b做一个合并操作，即把u与v合为一个点v'，把原来接在u和v的边都接到v'上去，这样就能够得到一个k阶图G'，G'的最小生成树T'可以继续使用这种方法得到。
- 然后我们用反证法证明T'+{<u, v>}是G的最小生成树。
- 如果T'+{<u,v>}不是最小生成树，最小生成树是T，即W(T)<W(T'+{<u,v>})，显然T应该包含<u,v>，否则，可以用<u,v>加入到T中，形成一个环，删除环上原有的任意一条边，形成一棵更小权值的生成树。而T-{<u,v>}，是G'的生成树。所以W(T-{<u,v>})<=W(T')，也就是W(T)<=W(T')+W(<u,v>)=W(T'+{<u,v>})，与W(T)<W(T'+ {<u,v>})产生了矛盾。于是假设不成立，T'+{<u,v>}是G的最小生成树，Kruskal算法对k+1阶图也适用。

```cpp
struct Edge{
    int u,v,w;
    bool operator <(const Edge&x)const{return w<x.w;}
}a[maxm];
int find(int x){return x==fa[x]?x:fa[x]=find(fa[x]);}
void Kruscal(){
    int Ans=0;
    sort(a+1,a+m+1);
    Rep(i,1,n) fa[i]=i;
    Rep(i,1,m){
        int u=find(a[i].u),v=find(a[i].v);
        if (u!=v){Ans+=a[i].w;fa[u]=v;}
    }
}
```

[SCOI2012]滑雪与时间胶囊

- a180285非常喜欢滑雪。他来到一座雪山，这里分布着M条供滑行的轨道和N个轨道之间的交点（同时也是景点），而且每个景点都有一编号i（1<=i<=N）和一高度Hi。a180285能从景点i 滑到景点j 当且仅当存在一条i 和j 之间的边，且i 的高度不小于j。
- 与其他滑雪爱好者不同，a180285喜欢用最短的滑行路径去访问尽量多的景点。如果仅仅访问一条路径上的景点，他会觉得数量太少。于是a180285拿出了他随身携带的时间胶囊。这是一种很神奇的药物，吃下之后可以立即回到上个经过的景点（不用移动也不被认为是a180285 滑行的距离）。请注意，这种神奇的药物是可以连续食用的，即能够回到较长时间之前到过的景点（比如上上个经过的景点和上上上个经过的景点）。
- 现在，a180285站在1号景点望着山下的目标，心潮澎湃。他十分想知道在不考虑时间胶囊消耗的情况下，以最短滑行距离滑到尽量多的景点的方案（即满足经过景点数最大的前提下使得滑行总距离最小）。你能帮他求出最短距离和景点数吗？

- 如果没有高度相等的点，那么就是一个有向无环图的最小树形图，贪心的让每一个点选入边中权值最小的就可以
- 加上了高度相等的点后，每一层都是无向边，要搞最小生成树
- 考虑这样做：先把最高层做最小生成树缩成一个点，再把这个点加入低一层做下一层的最小生成树
- 具体实现时只需kruskal排序时以边指向点的高度为第一关键字对边排序
- Prim从堆里取出节点时也以高度为第一关键字

任意点对的最小瓶颈路

- 给出一张带权无向图，求每两个顶点u和v之间的一条路径，使得路径上的最长边尽量短。
- 原图的最小生成树即为一棵最小瓶颈生成树（最大边权尽量小的生成树）。
- u到v在最小瓶颈生成树上的唯一路径即为一条u到v的最小瓶颈路。（不是唯一的一条）

经典题 货车运输

- 给定一个图。
- Q次询问，每次询问从x[i]走到y[i]的路径中，边权最小值的可能最大值是多少。
- n<=10000,m<=50000,Q<=30000

- 建出最大生成树之后问题变成了树链查询min
- 倍增树剖都可以

增量最小生成树

- 动态加入边，维护图的最小生成树T
- 考虑在上一次得到的最小生成树上，加入一条边e = (u, v)，图中恰好包含一个环。
- 根据回路性质，删除该回路上权值最大的边即可。
- 只需在加边前的最小生成树上找到u到v的唯一路径上权值最大的边，再和e比较，删除权值较大的一条。
- 由于路径唯一，用DFS或BFS找到这条路径即可。
- 进阶：LCT维护，可以做到M logN 的复杂度

次小生成树

- 把所有生成树按照权值之和从大到小排序，求排在第二位的生成树。
- 如果最小生成树不唯一，次小生成树的权值和最小生成树相同。
- 严格次小生成树指权值严格小于最小生成树的最大生成树。
- 次小生成树不会和最小生成树完全相同。
- 因此，可以枚举次小生成树中不在最小生成树中出现的边(u, v)。
- 在最小生成树上加入(u, v)后，图上会出现一个环。
- 根据回路性质，删除的边为在最小生成树u到v的路径上的最大边。
- 最后取出这样能得到的生成树的最小值即为次小生成树

一般来说基于边的Kruscal可扩展性更好

- UOJ Easy Round 1 C DZY Loves Graph
- 现在有 N 个点，进行 M 次操作，有三种情况：
  - 1 在 a, b 间添加一条长度为 i 的边，i 为操作标号。
  - 2 删除当前图中边权最大的 k 条边。
  - 3 撤销 i − 1 次操作，保证 i − 1 次操作不是撤销。
- 每次操作后，输出当前图的最小生成树大小。
- 范围：N, M ≤ 300000。

- 因为每次加入边的权值都是递增的，所以这本质就是一 个 kruskal 的过程。
- 我们用栈把对按秩合并数组进行的操作都存下来。并记录下每一个 时刻的 MST大小。
- 注：为了保证每一步的复杂度，必须使用按秩合并
- 当前为插入操作，把操作入栈。
- 当前为删除操作，若下一次操作不是撤销，那也直接退栈；否则直接输出对应的答案，不进行退栈操作。
- 效率：O(N log N)。

APIO 2013 TOLL

- 给出 N 个点 M 条边的图，每条边有权值（两两不同）；其中还有 K 条边权值未定。
- 你需要给这 K 条边定权值，然后求整张图的 MST 。
- 每个点有 Ai 个人，现在所有人都要沿着MST去 1 号点。
- 如果它们经过了某条待定边，则需要支付人数乘边权的代价。
- 求如何定权值使得代价和最大。
- 范围：$N \le 10^5$，$M \le 300000$，$K \le 20$。

- 首先如果MST确定以后，我们只需要DFS一遍树就可以统计代价了。
- 而不同的MST只有2^K种，即每一条不确定的边都有在或者不在两种可能。
- 如果一条特殊边在MST中，则它的权值不能超过:原图所有覆盖这条边的边的权值；而为了让答案最大，取这些边最小值即可。
- 所以我们可以2^K枚举每一条边选不选，优先把选的边连掉；
- 然后用kruskal求MST；对于每一条特殊边求出权值；DFS计算答案。
- 效率：O(M log M + 2^K*NK)。

- 考虑优化，显然我们可以先把这K条边全部用上，求一遍MST。
- 然后把图缩成K+1个点的图。
- 对这张图用原图的边求MST，得到K + 1个点K条边的图。
- 然后用上一个做法即可。
- 效率：O(M log M + 2^K*K^2)。

tree

- 给你一个无向带权连通图，每条边是黑色或白色。
- 让你求一棵最小权的恰好有need条白色边的生成树。
- 范围：$N \le 100000$，$M \le 200000$。

- 我们考虑将黑边和白边分别排序，现在同种颜色的边要加入最小生成树必然是按照从小到大的顺序。
- 之后我们二分x，然后令所有白边的权值加上x，对其求MST。
- 可以证明x越大，则MST中白边的数量越少；反之越多。
- 用归并排序顺次处理每条边，用并查集维护连通性即可。
- 效率：O(M log M + N log W α)。

Boruvka’s algorithm

- 初始所有点各自为一个独立的连通块。
- 进行若干轮操作，每一轮操作，令所有连通块找一条不在自己连通块里的，权值最小的出边相连。
- 因为每次操作，连通块个数至少缩小一半，所以效率 为O(M log N)。
- 用途：复杂度瓶颈在于“连通块找一条不在自己连通块里的，权值最小的出边”这个部分，如果这个操作能有数据结构或者题目性质加速，就能做到N*logN的复杂度，甚至在完全图里也能跑
- 据说随机图效率是O(M)的哦！

Zig Zag

- 有 N 个点,标号0~N-1,排成一一个环,有 M 次操作。
- 每次操作有三个参数A[i], B[i], C[i]
- 它会在这个图上加无穷多条边。
- 具体地可以用用一份代码来表示这个过程:
- AddEdge(x, y, z) 表示 x 向 y 连权值为 z 的边.

```
void Work(int a, int b, int c){
    AddEdge(a % N, b % N, c);
    Work(b, a + 1, c + 1);
}

for(int i = 0; i < M; i ++){
    Work(A[i], B[i], C[i]);
}
```

- 求连完所有边以后的MST大小
- N, M <= 10^5

- 考虑它每次的连边:
- (A, B, C)
- (B, A + 1, C + 1)
- (A + 1, B + 1, C + 2)…
- 考虑Kruskal:
- 连(B, A + 1, C + 1)时(A, B)必然联通
- 因此这条边等价于(A, A + 1, C + 1)
- 连(A + 1, B + 1, C + 2)时(B, A + 1)必然联通
- 因此这条边等价于(B, B + 1, C + 2)
- 因此一次操作可以等价于(A, B)之间的连边和相邻点之间的连边！相邻点之间的连边可以扫描线解决.
- 这样子边数就O(N+M)了.

ex

最小斯坦纳树

- 最小生成树是在给定的点集和边中寻求最短网络使所有点连通。
- 如果是只要求图中的某些点或者有某些性质的点连通（可以经过其他的点），那就是斯坦纳树

Codechef April 2012

- 给出一个 N * M 的矩阵A，第 i 行第 j 列的格子颜色为 $A_{i,j}$，权值 为 $V_{i,j}$。
- 颜色范围为 $[-1, N * M]$ 。
- 现在需要在这个矩阵里找出一个连通块。
- 要求颜色为 -1 的格子不能选，且这个联通块至少要包含 K 个不同的正数颜色，要求连通块内格子的权值和最小。
- 范围：$N, M \le 15, 1 \le K \le 7$。

- 我们先考虑这个问题的简化版本。
- 假设矩阵中只有 K 种不同的正数颜色，这就是非常经典的斯坦纳 树问题。
- 我们记 F[i][j][k] 为当前在矩阵的 (i, j) 点（或者说当前的树的树根是(i,j)），k 是一个二进制状态， 表示联通块中颜色的信息。
- 如果 k 二进制上某一位是 1 则表示当前联通块中已经包含这种颜色，否则说明没有选该颜色。

- 存在两种转移：
  - F[i][j][k] + V[x][y] → F[x][y][k]，其中 (x, y) 为与 (i, j) 相邻的格子。 这可以理解成建父亲
  - F[i][j][x] + F[i][j][y xor x] − V[i][j] → F[i][j][y]，其中 x and y = x。 这是合并不同的儿子
- 如果一个非障碍的格子 (i, j) 有正数颜色 Ai,j， 则 F[i][j][2A[i][j]−1] = V[i][j]；否则 F[i][j][0] = V[i][j]。
- 第二种转移时枚举子集，第一种因为会有环，用最短路进行转移，本题边权均为正数，所以用dijkastra
- 单次斯坦纳树的效率是 O(3^K NM + 2^K * dij(NM, NM))。

- 回到原问题，我们来考虑颜色总数大于 K 种的情况。
- 我们可以将所有的颜色随机给它一个 [1,K] 之间的数，表示把它的 颜色重标号。
- 因为现在整个矩阵的颜色不超过 K 种了，所有我们可以效仿之前 的斯坦纳树做法，求出这种重新标号下的最优解。
- 容易发现，这种重标号只可能使答案变劣，且肯定都是合法的。

- 现在我们考虑这种随机方案进行单次的正确性。
- 假设最后最优答案的联通块中含有的 K 种颜色为 A1, A2, .., AK 。
- 那么只要 A1, A2, .., AK 这 K 种颜色随机到的标号各不相同，也就 是说是 1 – K 的一个排列，那么就可以得到最优的答案。
- 令 T 为矩阵中不同的颜色数，那么单次正确率就是：
- $\frac{K!K^{T-K}}{K^T} = \frac{K!}{K^K}$
- 因为K ≤ 7，进行几百次就可以保证正确率了。

欧拉路径及回路

- 若一条路径经过每条边恰好一次，那么这条路径称作欧拉路径。
- 类似地，若一个路径经过每条边恰好一次，且这条路径的起点和终点相同，那么这条路径称作欧拉回路。

欧拉路径、回路判定方法

- 无向图：
- 若一个点的度为奇数，则称这个点为奇点。
- 若一个点的度为偶数，则称这个点为偶点。
- 若一张图有且仅有两个奇点，则这张图存在一条欧拉路径。
- 若一张图不存在奇点，则这张图存在一条欧拉回路。
- 有向图：
- 若一张图所有顶点入度与出度相等，则这张图存在一条欧拉回路。
- 若一张图有且仅有一个点入度比出度大1，一个点出度比入度大1，其他的所有点入度与出度相等，则这张图存在一条欧拉路径。

DFS求欧拉路径

- 首先判断该图是否存在欧拉路径/回路
- 对于存在欧拉回路的图，随意选择一个点作为起点。
- 否则，无向图选择一个奇点作为起点，有向图选择出度比入度大1的点作为起点。
- 从起点对图进行DFS，当回溯时将顶点加入欧拉路径。

- 从起点开始进行DFS，先找到一个包含起点的环/到终点的链
- 对于环/链上的点，若改点还有未覆盖到的出边，则以该点为起点，找到另一个包含该点的环。
- 不断重复这个操作，直到所有的边都被覆盖，最后连接所有环就是欧拉路径
- 考虑归纳，易证该算法正确

Waterloo local 2003.01.25

- 给出n个字符串。
- 若A串结尾和B串开头的字母相同，则可以把A和B连起来。
- 问是否可以把所有单词串成一串。
- 若能，给出字典序最小的一个方案。
- $n \le 1000$。

- 对于每个单词，把单词的起点和终点字母当作顶点。
- 每个单词即相当于一条从起点到终点的有向边。
- 判断这个图是否存在欧拉回路，若存在，用DFS求出这个欧 拉回路即可。
- 为保证字典序最小，先将所有字符串排序，按顺序加边即 可。 时间复杂度O(n+26)。

CF723E One-Way Reform

- 给出一个n个点m条边的无向图，你需要给每条边定向，使得有尽量多的点，入度等于出度，并构造方案

- 答案上界为该无向图中的偶点数量，考虑构造方案达到这个上界
- 建一个虚点S向所有奇点连边，这样奇点都变成了偶点，而奇点的个数一定是偶数，故S也是个偶点
- 于是新图存在欧拉回路，根据这个对边进行定向，则原图中所有偶点入度等于出度，达到上界

最小树形图

- 最小树形图是要对于一个有向图，求它的一组边权和最小的边，使得指定根可以走到其他所有点。
- 如果给定的是连通拓扑图，则我们直接对于每一个点选择它入度中边权最小的边即可。
- 而对于一般有向图，我们这样做之后，会出现一些环，怎么办呢？

朱刘算法

- 我们需要把环缩起来。
- 对于环内的边以后再也不会用到。
- 否则对于一条边(u, v)，令Min[v]表示v点的最小入度边，就 令(u, v)的边权减去Min[v]。
- 意义是如果选取了(u, v)，就需要把原先v选择的边断开，改为选 择(u, v)。
- 不断做这个过程，直到没有环就结束了。
- 效率：O(NM)。

非常感谢您的观看

THANK YOU FOR YOUR WATCHING