最短路径树

Shortest Path Tree

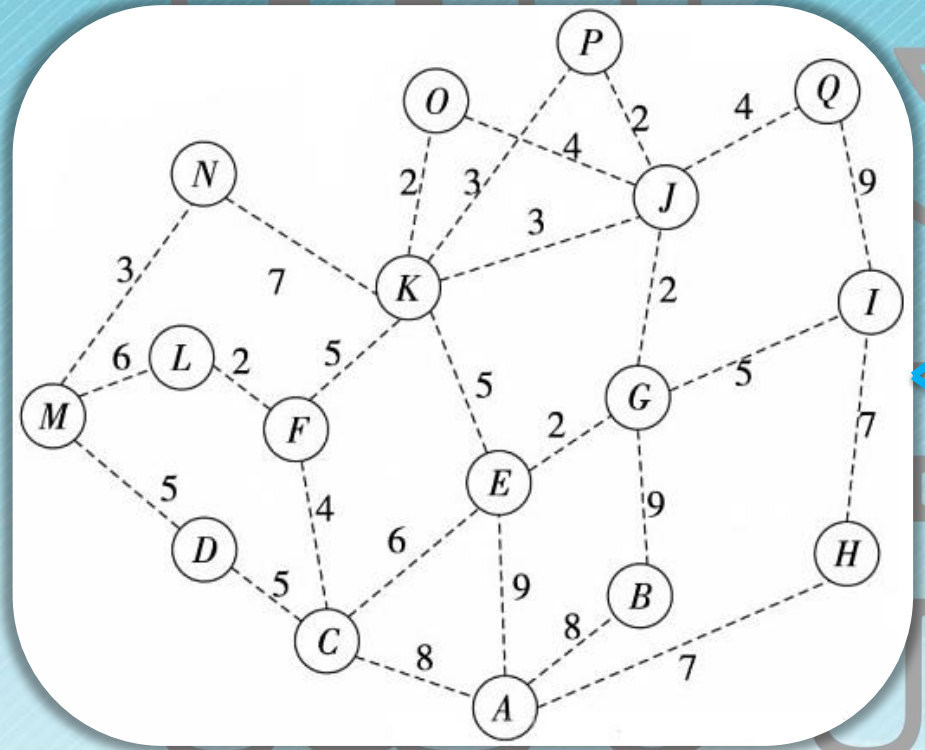
最短路径树（$Shortest\ Path\ Tree$），简称 SPT 。是一张无向连通图的生成树，该生成树满足从根节点到任意点的路径都为原图中根到任意点的最短路径。

一个连通无向图 G ，一个以顶点 v为根节点的最短路径树 T 是图 G 的满足下列条件的生成树：

树 T 中从根节点 v 到其它顶点 u 的路径距离，在图 G 中是从 v 到 u 的最短路径距离。

最短路径树 VS 最小生成树

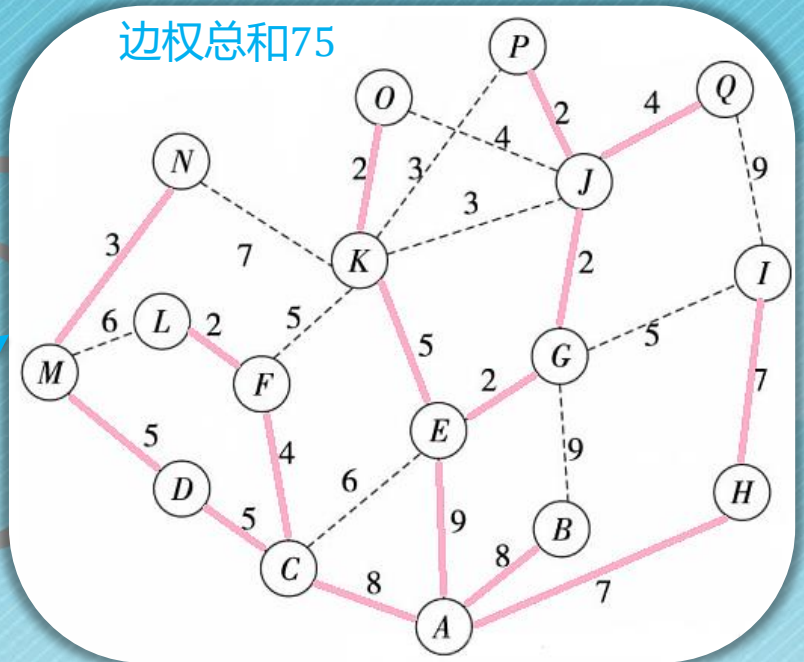
A为根节点



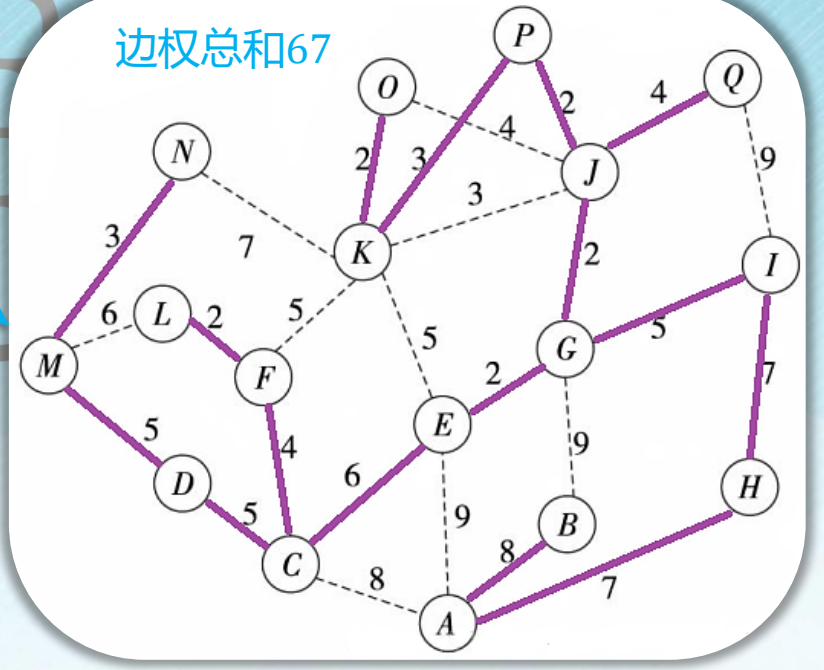
最短路径树

最小生成树

边权总和75



边权总和67



最短路径树的边权总和 ≥ 最小生成树的边权总和

例1:CF545E Paths and Trees

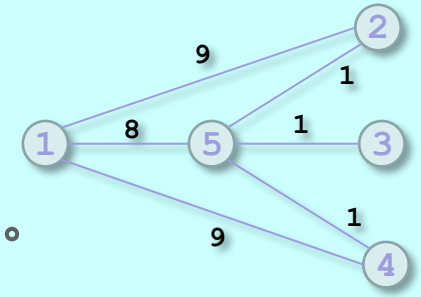
给定一张带边权的联通无向图 G，有 n个点和 m条边$(1 \le n, m \le 300000)$，指定一个顶点 u，此图中以u为根，**边权和最小**的 SPT，输出该树权值和以及每条**边的编号**。

怎么求最短路径树

根据定义，最短路径树的根节点到任一点的路径都为原图的最短路。
用$Dijkstra$等单源最短路算法，从根节点出发，跑最短路即可。

$dijkstra$算法是不断往一个集合里面加入节点，从而得到从某一个节点出发到所有节点的最短路，这个过程中共往里面加入了$n-1$个节点，对于每一个节点都是由一条边拉进来的，故运行$dijkstra$时就形成了一棵树(不会产生环)。

很多问题都需要保证最短路径树上**所有的边权总和最小**，怎么处理?
考虑贪心:
我们用$Pre[i]$记录从根到i最短路径上，最后一条边(连到i号点的边)的编号。
最短路算法进行时(松弛操作)：
设点u,v之间的边长度为$Len[k]$,编号为k
若$dis[v] > dis[u] + Len[k]$ 则 更新 $dis[v] = dis[u] + Len[k]$，$Pre[v] = k$
若$dis[v] == dis[u] + Len[k]$ 并且 $Len[k] < Len[Pre[v]]$ 则 更新 $Pre[v] = k$



例1:CF545E Paths and Trees

给定一张带边权的联通无向图 G，有 n个点和 m条边$(1 \le n, m \le 300000)$，指定一个顶点 u，此图中以u为根，**边权和最小**的 SPT，输出该树权值和以及每条**边的编号**。

我们用$Pre[i]$记录从根到i最短路径上，最后一条边(连到i号点的边)的编号。
最短路算法进行时(松弛操作)：
设点u, v之间的边长度为$Len[k]$,编号为k
若$dis[v] > dis[u] + Len[k]$ 则 更新 $dis[v] = dis[u] + Len[k]$，$Pre[v] = k$
若$dis[v] == dis[u] + Len[k]$ 并且 $Len[k] < Len[Pre[v]]$ 则 更新 $Pre[v] = k$
将$Pre[]$记录的边输出即可

例2:CF1076D Edge Deletion

给定一张带边权的联通无向图 G ，有 n个点和 m条边$(1 \leq n, m \leq 300000)$，要求删掉一些边，最多剩余 k 条边。定义好点是指删边后，1号节点到它的最短路长度仍然等于原图最短路长度的节点。

找到一种删边方案，使得最后剩余好点个数尽量多。

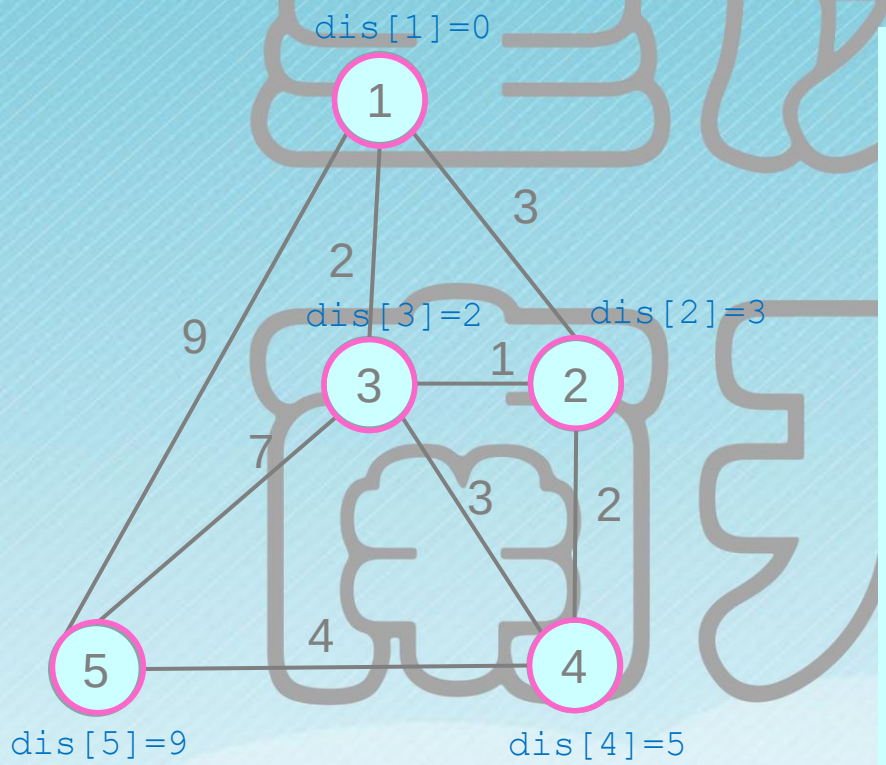
输出最后保留的边数以及这些边的编号。

第1步：1号为源点跑$dijkstra$求出最短路径树

第2步：从1号点出发，DFS遍历最短路径树，遍历$k+1$个点即可。(树中的点都是好点)这$k+1$个点恰好有k条边，并且每个点都是好点。

例3:CF1005F Berland and the Shortest Paths

给定一张带边权的联通无向图 G ，有 n个点和 m条边$(1 \le n,m \le 200000)$，边有边权，求最短路径树的方案数，并输出每种方案。



观察右图，从1号点出发用dijkstra求普通最短路径树的过程：
初始时：dis[1]=0，dis[2…5]=inf ，Fa[i]记录i的可能的父亲(即距离被谁更新)
1.一开始树中0个点
2.当前不在树中，dis最小的是1号点，1号点加入树中，以1号点为中转点，讨论：
因为dis[2]>dis[1]+3 所以 dis[2]=3，**Fa[2]=1**
因为dis[3]>dis[1]+2 所以 dis[3]=2，**Fa[3]=1**
因为dis[5]>dis[1]+9 所以 dis[5]=9，**Fa[5]=1**
3.当前不在树中，dis最小的是3号点，3号点加入树中，以3号点为中转点，讨论：
因为dis[2]==dis[3]+1 所以 **Fa[2]=1,3**
因为dis[4]>dis[3]+2 所以 dis[4]=5，**Fa[4]=3**
因为dis[5]==dis[3]+7 所以 **Fa[5]=1,3**
4.当前不在树中，dis最小的是2号点，2号点加入树中，以2号点为中转点，讨论：
因为dis[4]==dis[2]+2 所以 **Fa[4]=3,2**
5.当前不在树中，dis最小的是4号点，4号点加入树中，以4号点为中转点，讨论：
因为dis[5]==dis[4]+4 所以 **Fa[5]=1,3,4**
6.当前不在树中，dis最小的是5号点，5号点加入树中

2号点可能的父亲数为2，3号点可能的父亲数为1，4号点可能的父亲数为2，5号点可能的父亲数为3。总的方案数=2*1*2*3=12
vector<int>v[200005];v[i]记录最短路径树上，i可能的父亲们。

例4:关键公路

n个城市用m条双向公路连接，使得任意两个城市都能直接或间接地连通。其中城市编号为1..n，公路编号为1..m。任意个两个城市间的货物运输会选择最短路径，把这n*(n-1)条最短路径的和记为S。

现在你来寻找关键公路r，公路r必须满足：当r堵塞之后，S的值会变大（如果r堵塞后使得城市u和v不可达，则S为无穷大）。

n<=100, 1<=m<=3000, 1<=len<=10000

对于一个点u，如果删去的边不在它到其它点的最短路上，那么S是不会变的。考虑到两点之间的最短路不止一条，我们可以给每个点先建出一棵最短路径树出来，然后枚举最短路径树上的每条边并把它删掉，再跑一遍最短路，看S是否变大即可。

于是对**每个点**先建一个最短路径树出来，然后枚举每条边作为删除边，再跑一遍最短路看S是否变大即可。

基础习题：CF545E、 CF1076D、 CF1005F
提高习题：NKOJ8144 NKOJ1614 NKOJ8416 NKOJ8476

习题:安全路径NKOJ1614

Gremlins最近在农场上泛滥，它们经常会阻止牛们从农庄(牛棚$_1$)走到别的牛棚(牛$_i$的目的地是牛棚$_i$)。每一个gremlin只认识牛$_i$并且知道牛$_i$一般走到牛棚$_i$的最短路经。所以它们在牛$_i$到牛棚$_i$之前的最后一条牛路上等牛$_i$，当然，牛不愿意遇到Gremlins，所以准备找一条稍微不同的路经从牛棚$_1$走到牛棚$_i$，所以，请你为每一头牛$_i$找出避免$gremlin_i$的最短路经的长度。

农场上的M (2 <= M <= 200,000)条双向牛路编号为1..M并且能让所有牛到达它们的目的地，N(3 <= N <= 100,000)个编号为1..N的牛棚。

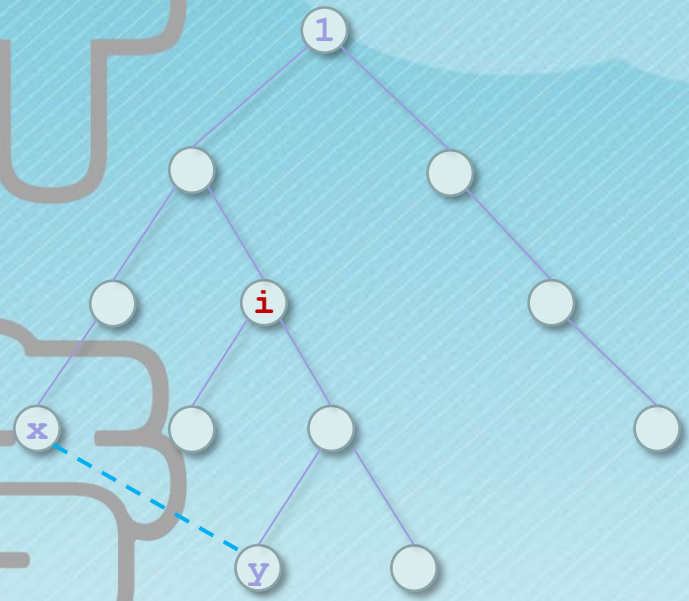
牛路i连接牛棚$a_i(1 <= a_i <= N)$和$b_i(1 <= b_i <= N)$并且需要时间$t_i(1 <= t_i <= 1,000)$通过。没有两条牛路连接同样的牛棚,所有牛路满足$a_i! = b_i$。在所有数据中，牛$_i$使用的牛棚$_1$到牛棚$_i$的最短路经是唯一的。

计算1号点到每个点的最短路线长度。但不能经过原本最短路线的最后一条边。

习题:安全路径NKOJ1614

一.求出以1号点为根的最短路径树,dis[i]记录点1到点i的最短距离。
删掉树中点i到父亲的边(即1到i路径上最后一条边)后,必须经过非树边才能到达i。贪心,显然只经过一条非树边才是最优的。

二.非树边x,y
1号点经过x,y到点i的路径长度为dis[x]+Len[x,y]+dis[y]-dis[i]
i不能为lca(x,y),因为删掉i与Ba[i]的边后,1就不再与x、y连通。



三.结论,对于非树边(x,y)
树中x到y路径上的任意一点z(z不能是lca(x,y)):
1号点出发经过边(x,y)到z的距离:
dis'[z]=dis[x]+dis[y]+Len[x,y]-dis[z]
dis[z]是一个定值,只需求最小的dis[x]+dis[y]+Len[x,y]

方法1:树链剖分
对于每条非树边(x,y),将树中x到y路径上的每一个点z(除lca(x,y))都进行下面更新操作:
dis'[z]=min(dis'[z],dis[x]+dis[y]+Len[x,y])
树链剖分即可。

习题:安全路径NKOJ1614

结论，对于非树边(x,y)，树中x到y路径上的任意一点z(z不能是lca(x,y))：

1号点出发经过边(x,y)到z的距离：dis'[z]=dis[x]+dis[y]+Len[x,y]-dis[z]

dis[z]是一个定值，只需要求最小的dis[x]+dis[y]+Len[x,y]

方法2：并查集

1.将所有非树边按Val=dis[x]+dis[y]+Len[x,y]由小到大排序

dis'[z]=min{dis[x]+dis[y]+Len[x,y]}-dis[z]

2.显然，每个点dis'[]一旦被更新，后面Val值大的不会再更新它了。

将被同一条边更新过的点合为一点，后面的边遇到它们后，直接向上跳过该点。

```
sort(E+1,E+Tot+1,cmp);    //按Val由小到大排序
for(int i=1;i<=n;i++)Fa[i]=i;
for(int i=1;i<=Tot;i++)   //共Tot条非树边
{
     int x=getFather(E[i].u),y=getFather(E[i].v);
     while(x!=y)
     {
          if(dep[x]<dep[y])swap(x,y);
          if(!Ans[x])Ans[x]=E[i].Val;//每个点只会被更新一次
          Fa[x]=getFather(Ba[x]);//Ba[x]为短路径树中x的父亲
          x=Fa[x];
     }
}
```

补充一道并查集压缩树上路径的题目：火车 NKOJ4850

A 国有n个城市，城市之间有一些双向道路相连，并且城市两两之间有唯一路径。现在有火车在城市a，需要经过m个城市。

火车按照以下规则行驶：每次行驶到还没有经过的城市中在m个城市中最靠前的。现在小A想知道火车经过这m个城市后所经过的道路数量。

N<=500000 ,M<=400000

火车 NKOJ4850

任意两点A,B的距离=dep[A]+dep[B]-2*dep[Lca_AB]

同时需要标记A,B路径中经过的点:
在计算Lca的同时,
　　将A到Lca_AB路径上的点做上访问标记
　　将B到Lca_AB路径上的点做上访问标记
但是，直接暴力标记会超时!

考虑到树上一条路径经过的点肯定是一段一段的，就可想到用并查集将一段路径合成一个点，每个点最多只能被合一次O(n)。
初始化时:
　　对于没有指定要经过的点，在并查集中的父亲设为它在树中的父亲,即fa[i]=Ba[i];
　　对于指定要经过的点，在并查集中的父亲设为它自己，即fa[i]=i;
在Lca操作时，在A,B到Lca路径上，遇到没被访问过的点，把这个点并到父亲所在并查集。

```
void dfs(int x)
{     //初始时，m个指定点的mark值为1，其它点mark值为0
      if(mark[x]==0)fa[x]=Ba[x]; else fa[x]=x;
      for(int p=Last[x];p;p=Next[p])
         if (End[p]!=Ba[x])
         {
             Ba[End[p]]=x;
             dep[End[p]]=dep[x]+1;
             dfs(End[p]);
         }
}
```

```
void GoUp(int x,int lca)
{
      if (dep[x]<dep[lca]) return;
      mark[x]=0;        //标记为已被经过
      fa[x]=Ba[x];
      GoUp(getFather(Ba[x]),lca);
}
```

本质是把经过的所有点都合并到同一个并查集里。

课后习题讲评　火车 NKOJ4850

```
for(int i=1;i<=m;i++)
    if (mark[a[i]])                    //mark[]==1 表示还没有被经过
    {   //上面也可以写成if(getFather(a[i])!=getFather(Now))
        lca=GetLca(a[i],Now);          //Now记录火车当前所在位置
        ans+=dep[a[i]]+dep[Now]-2*dep[lca];
        GoUp(a[i],lca);
        GoUp(Now,lca);
        Now=a[i];
    }
```