YCZX-Wilson_Lee 发表于 2022-10-09 22:11:23

A.一般图最小匹配

这个dp应该是比较基础的，考试时十分钟就推出来了。

显然排序后每个点和相邻点配对最优，因此设 $dp[i][j][0/1]$ 表示到第 i 个数匹配了 j 对，第 i 个数有没有用上。

转移方程：

$dp[i][j][0]=\min\{dp[i-1][j][0],dp[i-1][j][1]\}$

$dp[i][j][1]=dp[i-1][j-1][0]+a[i]-a[i-1]$

代码：

```
#include<bits/stdc++.h>
using namespace std;

#define now i&1
#define pre (i-1)&1
const int MAXN=5e3+5;
int a[MAXN];
long long dp[2][MAXN][2];
int main()
{
    int n,m;
    cin>>n>>m;
    for(int i=1;i<=n;++i) scanf("%d",&a[i]);
    sort(a+1,a+n+1);
    memset(dp[1],0x3f,sizeof(dp[1]));
    dp[1][0][0]=0;
    for(int i=2;i<=n;++i)
    {
        dp[now][0][0]=dp[pre][0][0];
        for(int j=1;j<=m;++j)
        {
            dp[now][j][0]=min(dp[pre][j][0],dp[pre][j][1]);
            dp[now][j][1]=dp[pre][j-1][0]+a[i]-a[i-1];
        }
    }
    cout<<min(dp[n&1][m][0],dp[n&1][m][1]);
    return 0;
}
```

B.重定向

分三种情况讨论：

1.$a[i]>0$ 且 $a[i+1]>0$：若 $a[i]>a[i+1]$ 删去 i。

2.$a[i]>0$ 且 $a[i+1]=0$：若 $a[i]>minnum$ 删去 i（$minnum$ 为未填数中最小的）。

3.$a[i]=0$：若 $minn[i]<minnum$ 删去 $pos[minn[i]]$（$minn[i]$ 为 i 的后缀最小值，pos 表示其对应的位置）。

代码：

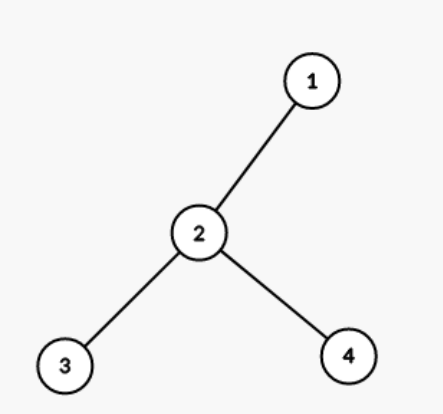
```
#include<bits/stdc++.h>
using namespace std;

const int MAXN=2e5+5;
const int INF=1e9;
int a[MAXN],b[MAXN],minn[MAXN];
bool used[MAXN];
```

```
int main()
{
        int T;
        cin>>T;
        int n;
        while(T--)
        {
                scanf("%d",&n);
                for(int i=1;i<=n;++i) used[i]=0;
                for(int i=1;i<=n;++i) scanf("%d",&a[i]),used[a[i]]=1,b[a[i]]=i;
                minn[n+1]=INF;
                for(int i=n;i>=1;--i) minn[i]=min(minn[i+1],(a[i]>0)?a[i]:INF);
                priority_queue< int,vector<int>,greater<int> >q;
                for(int i=1;i<=n;++i) if(!used[i]) q.push(i);
                int pos=n;
                for(int i=1;i<n;++i)
                {
                        if(!a[i])
                        {
                                if(minn[i]>q.top()) a[i]=q.top(),q.pop();
                                else
                                {
                                        a[i]=minn[i],pos=b[a[i]];
                                        break;
                                }
                                continue;
                        }
                        if(a[i+1])
                        {
                                if(a[i]>a[i+1])
                                {
                                        pos=i,q.push(a[i]);
                                        break;
                                }
                        }
                        else
                        {
                                if(a[i]>q.top())
                                {
                                        pos=i,q.push(a[i]);
                                        break;
                                }
                        }
                }
                for(int i=1;i<=n;++i)
                {
                        if(a[i] || i==pos) continue;
                        a[i]=q.top(),q.pop();
                }
                for(int i=1;i<=n;++i) if(i!=pos) printf("%d ",a[i]);
                putchar('\n');
        }
        return 0;
}
```

C.斯坦纳树

牛牛做法错误的情况：



当1，3，4点为询问的点集时，若求最小生成树，一定会有一条边被2次选中。

令属于 P 的点为实点，属于 P 的虚树 P' 的点为虚点，不难发现只要有一个虚点有3个及以上的分叉，就一定有边会重复统计。

当有虚点分叉数小于3时：

1.分叉数=2，直接把虚点两端的连起来，不用再管虚点了。

2.分叉数=1，直接删去虚点（此时它一定是叶子，不会对答案造成任何影响）。

当然，我们少考虑了一种情况，就是边权为 0 时，就算边被重复统计也是没有影响的。

直接用并查集维护连通块，把边权为 0 的两端加到同一个连通块里，连通块被删除当且仅当连通块内部所有点都被删除。

综上，我们从后往前删点，按上述策略维护虚点，若操作后无虚点，牛牛的做法正确，否则错误。

为避免重边的判断，连边采用set。

```
#include<bits/stdc++.h>
using namespace std;

const int MAXN=3e5+5;
struct EDGE
{
        int u,v,w;
}edge[MAXN];
set<int>s[MAXN];
int fa[MAXN],p[MAXN],tot;
int cnt[MAXN],sum=0;
bool ans[MAXN];
int getfa(int x){return fa[x]==x?x:fa[x]=getfa(fa[x]);}
void add(int x,int y){s[x].insert(y);}
void erase(int x,int y){s[x].erase(y);}
void del(int x)
{
        if(s[x].size()<=2 && !cnt[x])
        {
                --sum;
                int tmp[3],len=0;
                for(auto y:s[x]) tmp[++len]=y;
                s[x].clear();
                for(int i=1;i<=len;++i)
                {
                        for(int j=1;j<=len;++j) if(i!=j) add(tmp[i],tmp[j]);
                        erase(tmp[i],x);
                }
                for(int i=1;i<=len;++i) del(tmp[i]);
        }
}
int main()
{
        int n;
        cin>>n;
        for(int i=1;i<=n;++i) fa[i]=i;
        int u,v,w;
        for(int i=1;i<n;++i)
```

```
        {
                scanf("%d %d %d",&u,&v,&w);
                edge[i]=(EDGE){u,v,w};
                if(!w) fa[getfa(u)]=getfa(v);
        }
        for(int i=1;i<n;++i) if(edge[i].w) add(getfa(edge[i].u),getfa(edge[i].v)),
        for(int i=1;i<=n;++i) scanf("%d",&p[i]),++cnt[getfa(p[i])];
        for(int i=n;i>=1;--i)
        {
                ans[i]=(!sum);
                if(--cnt[getfa(p[i])]==0) ++sum,del(getfa(p[i]));
        }
        for(int i=1;i<=n;++i) putchar(ans[i]+'0');
        return 0;
}
```

题外话：大样例有误，正解跑出来根本对不上大样例。

D.直径

手玩 $n \leq 6$，能拿20pts（这也是本场T4满分了……）

待补……

后记：第二题的构造并不困难，实际却没想清楚（考场做法甚至过了大样例，最后喜提25pts），第三题考场打的树剖浪费太多时间，但凡对虚树稍微有些了解应该能想到上述结论。

笔者能力有限，有未述详尽或有误之处，欢迎指正。

收起 ^