Prufer 序列

算法简介

一种带编号生成树与数列之间的双射，主要用于解决组合计数问题。

Prufer 序列的性质

对一棵 n 个结点的带标号树，考虑每次取编号最小的叶子结点，将其删除。

然后将与其相邻的结点加入 Prufer 序列，直到只剩下两个结点。

显然这样操作得到的序列具有唯一性，删除到最后余下的节点中一定包含编号为 n 的结点。

接下来考虑得到的 Prufer 序列的性质。显然有 Prufer 序列长度为 $n-2$任意一个结点出现次数为其度数减一。

对一个长度为 $n-2$ 且只含 $1\sim n$ 的数列，每次找到标号最小的且不在序列中出现的点。

将它与当前序列的第一个点连一条边，然后删去序列的第一个点。易知这样得到的图唯一且最终图为树。

发现该操作对应之前寻找叶子节点删去并将其相邻节点加入序列的操作。

于是可以证明一棵 n 个结点的带标号树与一个任意长度为 $n-2$ 且只含 $1\sim n$ 的数列构成双射。

根据上述结论，不难得到 Cayley 公式：完全图 K_n 有 n^{n-2} 棵生成树。

Prufer 的编码与解码

考虑如何根据树建立 Prufer 序列，同时如何根据 Prufer 序列还原树。

首先不难利用优先队列可以 $O(n\log n)$ 实现，接下来考虑线性算法。

先考虑建立 Prufer 序列的过程，不妨先将无根树设置为以 n 为根的有根树，因为 n 一定不会被删除。

用指针维护当前编号最小的叶子节点。每次将其删除，然后如果与其相邻变为叶子节点，则考虑该结点的编号。

则如果其标号小于删除节点，则立刻将该结点删除，因为删除节点是最小叶子节点，所以该结点一定是新的最小叶子节点。

否则继续寻找下一个编号最小的叶子节点，于是可以 $O(n)$ 建立 Prufer 序列。

根据 Prufer 序列建树过程也类似，具体细节参考代码。

```
int f[MAXN],prufer[MAXN];
void prufer_encode(int n){
```

```
    static int deg[MAXN];
    _rep(i,1,n)deg[i]=0;
    _for(i,1,n)deg[f[i]]++;
    int pos=1;
    _rep(i,1,n-2){
        while(deg[pos])pos++;
        prufer[i]=f[pos];
        while(i<n-2&&!--deg[prufer[i]]&&prufer[i]<pos)
        prufer[i+1]=f[prufer[i]],i++;
        pos++;
    }
}
void prufer_decode(int n){
    static int deg[MAXN];
    _rep(i,1,n)deg[i]=0;
    _rep(i,1,n-2)deg[prufer[i]]++;
    prufer[n-1]=n;
    int pos=1;
    _for(i,1,n){
        while(deg[pos])pos++;
        f[pos]=prufer[i];
        while(i<n&&!--deg[prufer[i]]&&prufer[i]<pos)
        f[prufer[i]]=prufer[i+1],i++;
        pos++;
    }
}
```

算法练习

习题一