H - Truck History （最小生成树&Prim）

题意：给定n个字符串，任意两字符串直接的距离为相同位置不同 字符的个数。求生成n个字符串所需要最小的权值和。

思路：显然是最小生成树问题，只是距离转化一下。n个字符串看成n个结点，任选一个结点进行prim算法即可。

AC代码：

```
#include<iostream>
#include<cstdio>
#include<cstring>
#include<algorithm>
using namespace std;
const int N=2e3+5;
int a[N][N],n;
char c[N][10];
const int inf=0x3f3f3f3f;
int dis(int x,int y){ //求距离
    int ans=0;
    for(int i=0;i<7;i++)
        if(c[x][i]!=c[y][i]) ans++;
    return ans;
}
int prim(){ //prim板子
    int vis[N]={},d[N]={},ans=0;
    for(int i=1;i<=n;i++) d[i]=a[1][i];
    vis[1]=1;
    for(int i=2;i<=n;i++)
    {
        int mn=inf,p=0;
        for(int j=1;j<=n;j++)
            if(!vis[j]&&mn>d[j]) mn=d[j],p=j;
        if(mn==inf) break;
        ans+=mn,vis[p]=1;
        for(int j=1;j<=n;j++)
            if(!vis[j]&&d[j]>a[p][j]) d[j]=a[p][j];
    }
```

```
    return ans;
}
int main(){
    while(~scanf("%d",&n)&&n){
        for(int i=1;i<=n;i++)
        {
            scanf("%s",c[i]);
            for(int j=1;j<i;j++)
            {
                a[i][j]=a[j][i]=dis(i,j);
            }
        }
        printf("The highest possible quality is 1/%d.\n",prim());
    }
    return 0;
}
```

F - Agri-Net （最小生成树&kruskal)

思路：板子题。（第一次学这个算法标记一下）。思路就是对边排序，取n-1条边生成一棵权值和最小的树。生成树的过程用并查集实现。

AC代码：

```
xxxxxxxxxx
#include<cstdio>
#include<cstring>
#include<iostream>
#include<queue>
#include<cmath>
#include<algorithm>
#include<vector>
using namespace std;
const int N=1e2+5;
int n,s[N];
struct edge{
    int u,v,w;
    bool operator<(const edge&a)const{ //重载’<’
```

```
        return w<a.w;
    }
}now;
vector<edge>e;
int find(int x){
    if(s[x]!=x) s[x]=find(s[x]);
    return s[x];
}
int kruskal(){
    int ans=0;
    for(int i=1;i<=n;i++) s[i]=i;
    sort(e.begin(),e.end());//排序
    int tot=0;
    for(int i=0;i<e.size();i++){
        int fa=find(e[i].u),fb=find(e[i].v);
        if(fa==fb) continue;//防止生成环.
        tot++;
        s[fb]=fa;
        ans+=e[i].w;
        if(tot==n-1) break; //n-1条边生成一棵树.
    }
    return ans;
}
int   main(){
    while(~scanf("%d",&n)){
        e.clear();//初始化
    for(int i=1;i<=n;i++)
        for(int j=1,w;j<=n;j++)
        {
            scanf("%d",&w);
            if(i>j) e.push_back({i,j,w});//防止重边
        }
        printf("%d\n",kruskal());
    }
        return 0;
}
```