**1.最大多位整数问题**

思路：贪心，选择首位最大的字符串放在最前面，按字符串首位的大小依次输出字符串，这

样可以连接出最大整数

#include<iostream>

#include<algorithm>

using namespace std;

//按字符串首字母对字符串进行降序排序

bool cmp(string s1,string s2)

{

    return s1+s2>s2+s1;

}

int main()

{

    int n;

    cin>>n;

    string\* p=new string[n];

    for(int i=0;i<n;i++){

        cin>>p[i];

    }

    sort(p,p+n,cmp);

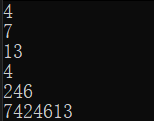
    for(int i=0;i<n;i++){

        cout<<p[i];//字符串由大到小依次连接，形成最大整数

    }

    return 0;

}



**2.Kruskal算法实现最小生成树**

思路：先将所有边按权值升序排序，从最小的边开始选择，若新选的边不会和现有的任意一条边构成环，则将该边加入。是否生成环可以利用并查集进行判断。当选取的边数达到V-1（V为顶点数）时，最小生成树生成。

#include<iostream>

#include<vector>

#include<algorithm>

using namespace std;

struct edge

{

    int u;

    int v;

    int weight;

};

edge e[20];

bool cmp(edge e1, edge e2)

{

    return e1.weight<e2.weight;

}

int parent[20]={0};

void init(int V)

{

    for(int i=1;i<=V;i++){

        parent[i]=i;

    }

}

int seek(int i)

{

    if(parent[i]==i){

        return i;

    }

    else{

        return seek(parent[i]);

    }

}

void merge(int a,int b)

{

    int x=seek(a);

    int y=seek(b);

    parent[y]=x;

}

int V,N;

int main()

{

    cout<<"请输入顶点数"<<endl;

    cin>>V;

    cout<<"请输入边数"<<endl;

    cin>>N;

    cout<<"请依次输入每条边及权重"<<endl;

    for(int i=0;i<N;i++){

       cin>>e[i].u>>e[i].v>>e[i].weight;

    }

    sort(e,e+N,cmp);

    init(V);

    int cost=0;

    int cnt=0;

    for(int i=0;i<N;i++){

        if(seek(e[i].u)!=seek(e[i].v)){

            merge(e[i].u,e[i].v);

            cost+=e[i].weight;

            cnt++;

        }

        if(cnt==N-1){

            break;

        }

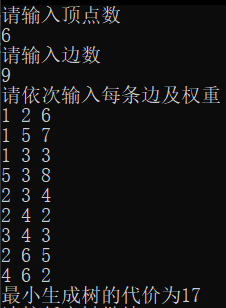
    }

    cout<<"最小生成树的代价为"<<cost<<endl;

    system("pause");

    return 0;

}



**3.跳跃问题**

思路：维护所能到达的最远位置，记为边界。数组遍历时，若更新后的最远位置大于边界，边界更新并且步数+1

