122492508@qq.com pq是不可战胜的

总分: 100 / 100

编程题 总分: 100 / 100

7-1 Forever 答案正确 得分: 20 / 20

"Forever number" is a positive integer A with K digits, satisfying the following constrains:

- the sum of all the digits of A is m;
- the sum of all the digits of $A+1$ is n; and
- the greatest common divisor of m and n is a prime number which is greater than 2.

Now you are supposed to find these forever numbers.

Input Specification:

Each input file contains one test case. For each test case, the first line contains a positive integer N (≤ 5). Then N lines follow, each gives a pair of K ($3 < K < 10$) and m ($1 < m < 90$), of which the meanings are given in the problem description.

Output Specification:

For each pair of K and m, first print in a line `Case X`, where `X` is the case index (starts from 1). Then print n and A in the following line. The numbers must be separated by a space. If the solution is not unique, output in the ascending order of n. If still not unique, output in the ascending order of A. If there is no solution, output `No Solution`.

Sample Input:

```
2
6 45
7 80
```

Sample Output:

```
Case 1
10 189999
10 279999
10 369999
10 459999
10 549999
10 639999
10 729999
10 819999
10 909999
Case 2
No Solution
```

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

struct answer {
        int n,A;
};

bool cmp(answer a,answer b) {
        if(a.n!=b.n)
                return a.n<b.n;
        else
                return a.A<b.A;
}

int gcd(int a,int b) {
        if(b==0)
                return a;
        else
                return gcd(b,a%b);
```

```
}

bool isPrime(int n) {
        if(n<=1)
                return false;
        int sqr=(int)sqrt(n*1.0);
        for(int i=2; i<=sqr; i++)
                if(n%i==0)
                        return false;
        return true;
}

int calSum(int n) {
        int sum=0;
        while(n!=0) {
                sum+=n%10;
                n/=10;
        }
        return sum;
}

int main() {
        int N,k,m;
        scanf("%d",&N);
        for(int i=1; i<=N; i++) {
                scanf("%d%d",&k,&m); //k>3 则尾数必为99
                printf("Case %d\n",i);
                vector<answer> ans;
                int tm=m-18; //只需计算剩余几位的和为tm
                int minn=pow(10,k-3),maxn=pow(10,k-2);
                //cout<<"minn="<<minn<<" maxn="<<maxn<<endl;
                for(int tA=minn; tA<maxn; tA++) {
                        //cout<<"sum="<<calSum(tA)<<endl;
                        if(calSum(tA)==tm) {
                                int A=tA*100+99;
                                int B=A+1;
                                int n=calSum(B);
                                if(isPrime(gcd(m,n))&&gcd(m,n)>2) {
                                        ans.push_back({n,A});
                                }
                        }
                }
                if(ans.size()!=0) {
                        sort(ans.begin(),ans.end(),cmp);
                        for(int i=0; i<ans.size(); i++) {
                                printf("%d %d\n",ans[i].n,ans[i].A);
                        }
                } else {
                        printf("No Solution\n");
                }
        }
        return 0;
}
```

测试点	结果	耗时	内存
0	答案正确	3 ms	488KB
1	答案正确	2 ms	516KB
2	答案正确	12 ms	352KB
3	答案正确	126 ms	416KB

7-2 Merging Linked Lists 答案正确 得分: 25 / 25

Given two singly linked lists $L^1 = a^1 \to a^2 \to \cdots \to a^{n-1} \to a^n$ and $L^2 = b^1 \to b^2 \to \cdots \to b^{m-1} \to b^m$. If $n \geq 2m$, you are supposed to reverse and merge the shorter one into the longer one to obtain a list like $a^1 \to a^2 \to b^m \to a^3 \to a^4 \to b^{m-1} \cdots$. For example, given one list being 6→7 and the other one 1→2→3→4→5, you must output 1→2→7→3→4→6→5.

Input Specification:

Each input file contains one test case. For each case, the first line contains the two addresses of the first nodes of L^1 and L^2, plus a positive N ($\leq 10^5$) which is the total number of nodes given. The address of a node is a 5-digit nonnegative integer, and NULL is represented by `-1`.

Then N lines follow, each describes a node in the format:

```
Address Data Next
```

where `Address` is the position of the node, `Data` is a positive integer no more than 10^5, and `Next` is the position of the next node. It is guaranteed that no list is empty, and the longer list is at least twice as long as the shorter one.

Output Specification:

For each case, output in order the resulting linked list. Each node occupies a line, and is printed in the same format as in the input.

Sample Input:

```
00100 01000 7
02233 2 34891
00100 6 00001
34891 3 10086
01000 1 02233
00033 5 -1
10086 4 00033
00001 7 -1
```

Sample Output:

```
01000 1 02233
02233 2 00001
00001 7 34891
34891 3 10086
10086 4 00100
00100 6 00033
00033 5 -1
```

```
#include <bits/stdc++.h>
using namespace std;
const int maxn = 100010;

struct Node {
        int address,data,next;
        int order;
} node[maxn];

bool cmp(Node a,Node b) {
        return a.order<b.order;
}

int main() {
        for(int i=0; i<maxn; i++)
                node[i].order=maxn;
        int b1,b2,n,address;
        scanf("%d%d%d",&b1,&b2,&n);
        for(int i=0; i<n; i++) {
                scanf("%d",&address);
                node[address].address=address;
                scanf("%d%d",&node[address].data,&node[address].next);
        }
        int p1=b1,cnt1=0;
        while(p1!=-1) {
                cnt1++;
                p1=node[p1].next;
        }
        int p2=b2,cnt2=0;
        while(p2!=-1) {
                cnt2++;
                p2=node[p2].next;
        }
        int cnt=cnt1+cnt2;
        if(cnt2>cnt1) { //长的链表设为b1
                swap(b1,b2);
                swap(cnt1,cnt2);
        }
        int ord=1;
        p1=b1;
        while(p1!=-1) {
                node[p1].order=ord++;
                p1=node[p1].next;
        }
        p2=b2;
        while(p2!=-1) {
                node[p2].order=ord++;
                p2=node[p2].next;
        }
        sort(node,node+maxn,cmp);
        int j=0;
        if(cnt1!=0&&cnt2!=0) {
                for(int i=0; i<cnt1; i++) {
```

```
                        printf("%05d %d ",node[i].address,node[i].data);
                        j++;
                        if(j%2==0&&j!=0&&j/2<=cnt2) {
                                printf("%05d\n",node[cnt-j/2].address);
                                printf("%05d %d ",node[cnt-j/2].address,node[cnt-j/2].data);
                                if(i!=cnt1-1)
                                        printf("%05d\n",node[i+1].address);
                                else
                                        printf("-1\n");
                        } else {
                                if(i!=cnt1-1)
                                        printf("%05d\n",node[i+1].address);
                                else
                                        printf("-1\n");
                        }
                }
        }
        if(cnt1!=0&&cnt2==0) {
                for(int i=0; i<cnt1; i++) {
                        if(i!=cnt1-1)
                                printf("%05d %d %05d\n",node[i].address,node[i].data,node[i+1].address);
                        else
                                printf("%05d %d -1\n",node[i].address,node[i].data);
                }
        }
        if(cnt1==0&&cnt2!=0) {
                for(int i=0; i<cnt2; i++) {
                        if(i!=cnt2-1)
                                printf("%05d %d %05d\n",node[i].address,node[i].data,node[i+1].address);
                        else
                                printf("%05d %d -1\n",node[i].address,node[i].data);
                }
        }
        if(cnt1==0&&cnt2==0) {
                printf("0 -1\n");
        }
        return 0;
}
```

测试点	结果	耗时	内存
0	答案正确	6 ms	1896KB
1	答案正确	6 ms	2048KB
2	答案正确	6 ms	2020KB
3	答案正确	6 ms	1952KB
4	答案正确	62 ms	3456KB

7-3 Postfix Expression 答案正确 得分: 25 / 25

Given a syntax tree (binary), you are supposed to output the corresponding postfix expression, with parentheses reflecting the precedences of the operators.

Input Specification:

Each input file contains one test case. For each case, the first line gives a positive integer N (≤ 20) which is the total number of nodes in the syntax tree. Then N lines follow, each gives the information of a node (the i-th line corresponds to the i-th node) in the format:

```
data left_child right_child
```

where `data` is a string of no more than 10 characters, `left_child` and `right_child` are the indices of this node's left and right children, respectively. The nodes are indexed from 1 to N. The NULL link is represented by -1. The figures 1 and 2 correspond to the samples 1 and 2, respectively.

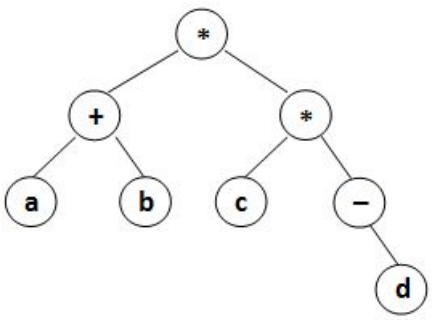

Figure 1

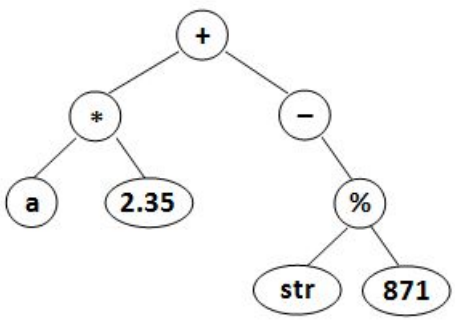

Figure 2

Output Specification:

For each case, print in a line the postfix expression, with parentheses reflecting the precedences of the operators.There must be no space between any symbols.

Sample Input 1:

```
8
* 8 7
a -1 -1
* 4 1
+ 2 5
b -1 -1
d -1 -1
- -1 6
c -1 -1
```

Sample Output 1:

```
(((a)(b)+)((c)(-(d))*)*)
```

Sample Input 2:

```
8
2.35 -1 -1
* 6 1
- -1 4
% 7 8
+ 2 3
a -1 -1
str -1 -1
871 -1 -1
```

Sample Output 2:

```
(((a)(2.35)*)(-((str)(871)%))+)
```

```
#include <bits/stdc++.h>
using namespace std;
const int maxn = 21;

int n,root;
bool isRoot[maxn];

struct Node {
        string data;
        int lchild,rchild;
} node[maxn];

string ans;
void postOrder(int root) {
        if(root==-1)
                return;
        if(node[root].lchild!=-1&&node[root].rchild!=-1) {
                ans+="(";
                postOrder(node[root].lchild);
                postOrder(node[root].rchild);
                ans+=node[root].data+")";
                return;
        }
        if(node[root].lchild==-1&&node[root].rchild!=-1){
                ans+="("+node[root].data;
                postOrder(node[root].rchild);
                ans+=")";
                return;
        }
        if(node[root].lchild==-1&&node[root].rchild==-1) {
                ans+="("+node[root].data+")";
                return;
        }
}
```

```
int main() {
	fill(isRoot,isRoot+maxn,true);
	int n;
	scanf("%d",&n);
	for(int i=1; i<=n; i++) {
		cin>>node[i].data;
		scanf("%d%d",&node[i].lchild,&node[i].rchild);
		isRoot[node[i].lchild]=false;
		isRoot[node[i].rchild]=false;
	}
	for(int i=1; i<=n; i++) {
		if(isRoot[i]) {
			root=i;
			break;
		}
	}
	postOrder(root);
	cout<<ans<<endl;
	return 0;
}
```

测试点	结果	耗时	内存
0	答案正确	3 ms	372KB
1	答案正确	2 ms	296KB
2	答案正确	3 ms	420KB
3	答案正确	2 ms	256KB
4	答案正确	2 ms	376KB

7-4 Dijkstra Sequence 答案正确 得分: 30 / 30

Dijkstra's algorithm is one of the very famous greedy algorithms. It is used for solving the single source shortest path problem which gives the shortest paths from one particular source vertex to all the other vertices of the given graph. It was conceived by computer scientist Edsger W. Dijkstra in 1956 and published three years later.

In this algorithm, a set contains vertices included in shortest path tree is maintained. During each step, we find one vertex which is not yet included and has a minimum distance from the source, and collect it into the set. Hence step by step an ordered sequence of vertices, let's call it **Dijkstra sequence**, is generated by Dijkstra's algorithm.

On the other hand, for a given graph, there could be more than one Dijkstra sequence. For example, both { 5, 1, 3, 4, 2 } and { 5, 3, 1, 2, 4 } are Dijkstra sequences for the graph, where 5 is the source. Your job is to check whether a given sequence is Dijkstra sequence or not.

Input Specification:

Each input file contains one test case. For each case, the first line contains two positive integers N_v ($\le 10^3$) and N_e ($\le 10^5$), which are the total numbers of vertices and edges, respectively. Hence the vertices are numbered from 1 to N_v.

Then N_e lines follow, each describes an edge by giving the indices of the vertices at the two ends, followed by a positive integer weight (≤ 100) of the edge. It is guaranteed that the given graph is connected.

Finally the number of queries, K, is given as a positive integer no larger than 100, followed by K lines of sequences, each contains a permutationof the N_v vertices. It is assumed that the first vertex is the source for each sequence.

All the inputs in a line are separated by a space.

Output Specification:

For each of the K sequences, print in a line `Yes` if it is a Dijkstra sequence, or `No` if not.

Sample Input:

```
5 7
1 2 2
1 5 1
2 3 1
2 4 1
2 5 2
3 5 1
3 4 1
4
5 1 3 4 2
5 3 1 2 4
```

```
2 3 4 5 1
3 2 1 5 4
```

Sample Output:

```
Yes
Yes
Yes
No
```

```
#include <bits/stdc++.h>
using namespace std;
const int maxn = 1010;
const int inf = 1000000000;

int n,m,k,st;
int g[maxn][maxn];
vector<int> v;

int d[maxn];
bool vis[maxn];
void Dijkstra(int s) {
        fill(d,d+maxn,inf);
        fill(vis,vis+maxn,false);
        d[s]=0;
        for(int i=1; i<=n; i++) {
                int u=-1,min=inf;
                for(int j=1; j<=n; j++) {
                        if(vis[j]==false&&d[j]<min) {
                                u=j;
                                min=d[j];
                        }
                }
                if(u==-1)
                        return;
                vis[u]=true;
                for(int v=1; v<=n; v++) {
                        if(vis[v]==false&&g[u][v]!=inf) {
                                if(d[u]+g[u][v]<d[v]) {
                                        d[v]=d[u]+g[u][v];
                                }
                        }
                }
        }
}

int main() {
        fill(g[0],g[0]+maxn*maxn,inf);
        int a,b,t;
        scanf("%d%d",&n,&m);
        for(int i=0; i<m; i++) {
                scanf("%d%d%d",&a,&b,&t);
                g[a][b]=g[b][a]=t;
        }
        scanf("%d",&k);
        while(k--) {
                bool flag=true;
                v.clear();
                for(int i=1; i<=n; i++) {
                        scanf("%d",&t);
                        if(i==1) {
                                st=t;
                                Dijkstra(st);
                        } else {
                                v.push_back(t);
                        }
                }
                for(int i=1; i<=n-1; i++) {
                        if(d[v[i]]<d[v[i-1]]) {
                                flag=false;
                                break;
                        }
                }
                if(flag)
                        printf("Yes\n");
                else
                        printf("No\n");
        }
        return 0;
}
```

测试点	结果	耗时	内存

测试点	结果	耗时	内存
0	答案正确	7 ms	4392KB
1	答案正确	20 ms	4352KB
2	答案正确	7 ms	4352KB
3	答案正确	539 ms	4352KB