ABC073 Editorial

writer:namonakiacc

2017/9/9

For international readers: English Editorial starts at page 4.

A: September 9

整数 N は二桁なので、N の十の位と一の位の少なくとも一方が 9 であるとき「Yes」、そうでないとき「No」を出力すればよい。

```
#include <cstdio>
int N;
int main()
{
        scanf("%d",&N);
        if(N%10==9||N/10==9)puts("Yes");
        else puts("No");
}
```

B: Theater

l_i 番目から r_i 番目までの連続した席に人が座っているとき、それらは $r_i - l_i + 1$ 人である。　条件より、同じ席に複数の人が座ることはないので、$\sum_{i=1}^{N}(r_i - l_i + 1)$ が答えとなる。よって、これを計算して出力すればよい。
それ以外にも、席数と人数が高々100000 なため、席 1 から席 100000 までのすべての席において、人がいるかいないかをフラグで持っておき、各 i において l_i から r_i までにフラグを立てていく方法でも解くことができる。

```
#include <cstdio>
int N,res,l,r;
int main()
{
        scanf("%d",&N);
        for(int i=0;i<N;i++)scanf("%d%d",&l,&r),res+=r-l+1;
        printf("%d\n",res);
}
```

C: Write and Erase

まず、どのような場合に数が最後に紙に書かれており、どのような場合に書かれていないかを考えてみる。

A_i に一度しか登場しなければ、その数は最後まで紙に残る。 A_i に二度登場すれば、その数は最後には紙に残っていない。同様に考えていくと、 A_i に偶数回登場する数は最後に紙には書かれていないが、奇数回登場する数は最後に紙に書かれているとわかる。

よって、求めるのは奇数回登場する数が何種類あるかで良いことがわかる。

N は最大で 100000、A_i は最大で 10^9 であり、 $O(N^2)$ 回かけて数えたり、 10^9 個のフラグを持つような手法はとることができず、異なる方法を考える必要がある。

A_i をソートすると、同じ数が隣り合うように並べ替えられる。これを左から順にみていくと、奇数回登場する数が何種類あるかを数えることができる。

ソート部分が $O(NlogN)$ 、 左から見ていく部分が $O(N)$ より、これで間に合う。

また、C++における set のようなデータ構造を用いて紙に書かれている数を持っておき、実際にシミュレーションしても解くことができる。

```
#include <cstdio>
#include <algorithm>
using namespace std;
int N,ptr,res;
int A[100000];
int main()
{
        scanf("%d",&N);
        for(int i=0;i<N;i++)scanf("%d",&A[i]);
        sort(A,A+N);
        while(ptr<N)
        {
                int cc=A[ptr],f=0;
                while(ptr<N&&A[ptr]==cc)f++,ptr++;
                res+=f%2;
        }
        printf("%d\n",res);
}
```

D: joisino's travel

N が 200 以下なので、町と町の間の距離はワーシャルフロイド法で求めて問題ない。

そして、 r_i の並べ方だが、こちらも R が 8 以下なので、すべての順列について試してみることができる。

順列の生成方法はいろいろ考えられるが、深さ優先探索を用いても十分間に合う。

また、C++における next_permutation のような標準ライブラリを用いて解くのもよいだろう。

```
#include <cstdio>
using namespace std;
#define INF (1<<29)

int N,M,R;
int d[201][201];
int r[9];
int A,B,C;
int res;
bool used[9];
```

```
void dfs(int c,int las,int dist)
{
        if(c==R+1)
        {
                if(res>dist)res=dist;
                return;
        }
        for(int i=1;i<=R;i++)if(!used[i])
        {
                used[i]=true;
                if(las==-1)dfs(c+1,i,0);
                else dfs(c+1,i,dist+d[r[las]][r[i]]);
                used[i]=false;
        }
}


int main()
{
        scanf("%d%d%d",&N,&M,&R);
        for(int i=1;i<=N;i++)for(int j=1;j<=N;j++)if(i!=j)d[i][j]=INF;
        for(int i=1;i<=R;i++)scanf("%d",&r[i]);
        for(int i=1;i<=M;i++)
        {
                scanf("%d%d%d",&A,&B,&C);
                if(d[A][B]>C)d[A][B]=d[B][A]=C;
        }
        for(int k=1;k<=N;k++)
                for(int i=1;i<=N;i++)
                        for(int j=1;j<=N;j++)
                                if(d[i][j]>d[i][k]+d[k][j])
                                        d[i][j]=d[i][k]+d[k][j];
        res=INF;
        dfs(1,-1,0);
        printf("%d\n",res);
}
```

A: September 9

N is a two-digit number,so you can get 100 points by output "Yes" when ten's digit or one's digit (or both of them) of N is 9, "No" otherwise.

```
#include <cstdio>
int N;
int main()
{
        scanf("%d",&N);
        if(N%10==9||N/10==9)puts("Yes");
        else puts("No");
}
```

B: Theater

When one group occupies Seat l_i to r_i , there are $r_i - l_i + 1$ people in the group.
No seat is occupied by more than one person, so $\sum_{i=1}^{N}(r_i - l_i + 1)$ is the answer.
So, you can get 200 points by output it.
And,it is OK to simulate it because the number of seat and audiences is no more than 100000.

```
#include <cstdio>
int N,res,l,r;
int main()
{
        scanf("%d",&N);
        for(int i=0;i<N;i++)scanf("%d%d",&l,&r),res+=r-l+1;
        printf("%d\n",res);
}
```

C: Write and Erase

First,thinking about which type of number remains on the paper,and which doesn't.
If a number appears once in A_i , it will remain. If appears twice, it will not remain.
Thinking as such, you can understand that numbers which appears in A_i even-numbers time will not remain,and which did in A_i odd-numbers time will remain.
So, it is OK to count the variety of number which appears in A_i odd-numbers time.
By sorting A_i ,we can make A_i such that same numbers occupies continuous locations in array.So, you can solve it by looking this array from left or right.
And, it is OK to solve by simulating this.

```
#include <cstdio>
#include <algorithm>
using namespace std;
int N,ptr,res;
int A[100000];
int main()
{
        scanf("%d",&N);
```

```
        for(int i=0;i<N;i++)scanf("%d",&A[i]);
        sort(A,A+N);
        while(ptr<N)
        {
                int cc=A[ptr],f=0;
                while(ptr<N&&A[ptr]==cc)f++,ptr++;
                res+=f%2;
        }
        printf("%d\n",res);
}
```

D: joisino's travel

N is no more than 200,so there is no problem to calculate distances of cities by Warshall-Floyd algorithm.
And, R is no more than 8,so it is possible to try in all pattern of permutations.
There are many ways to make permutations, but it is OK too in Depth First Search.

And, of course it is ok to use like next_permutation in C++ STL.

```
#include <cstdio>
using namespace std;
#define INF (1<<29)

int N,M,R;
int d[201][201];
int r[9];
int A,B,C;
int res;
bool used[9];

void dfs(int c,int las,int dist)
{
        if(c==R+1)
        {
                if(res>dist)res=dist;
                return;
        }
        for(int i=1;i<=R;i++)if(!used[i])
        {
                used[i]=true;
                if(las==-1)dfs(c+1,i,0);
                else dfs(c+1,i,dist+d[r[las]][r[i]]);
                used[i]=false;
        }
}


int main()
{
        scanf("%d%d%d",&N,&M,&R);
        for(int i=1;i<=N;i++)for(int j=1;j<=N;j++)if(i!=j)d[i][j]=INF;
        for(int i=1;i<=R;i++)scanf("%d",&r[i]);
        for(int i=1;i<=M;i++)
        {
                scanf("%d%d%d",&A,&B,&C);
```

```
                if(d[A][B]>C)d[A][B]=d[B][A]=C;
        }
        for(int k=1;k<=N;k++)
                for(int i=1;i<=N;i++)
                        for(int j=1;j<=N;j++)
                                if(d[i][j]>d[i][k]+d[k][j])
                                        d[i][j]=d[i][k]+d[k][j];
        res=INF;
        dfs(1,-1,0);
        printf("%d\n",res);
}
```