МЕЂУНАРОДНИ ПРОГРАМЕРСКИ ТУРНИР
24 новембар, 2012, Шумен, Бугарска
Јуниори

Задатак B1. РАВНОТЕЖА

Дате су теразије са два таса и n тегова чије су масе a_1, a_2, …, a_n. Постављамо сваки од датих тегова на теразије, један за другим, тако да у сваком моменту леви тас није никад тежи од десног таса. У сваком кораку, бирамо један тег који још није постављен на теразије, и смештамо га или на леви тас или на десни тас. Понављамо ове кораке све док не употребимо све тегове. Напишите програм **`balance`**, који израчунава број начина на који можемо то урадити.

Улаз

У првом реду стандардног улаза дат је цео број n ($0 < n < 10$). У другом реду се налази n целих бројева: a_1, a_2, …, a_n ($0 < a_1, a_2, \ldots, a_n < 1000$).

Излаз

Ваш програм мора да испише један број на стандардни излаз – захтевани број начина.

Пример

Улаз

```
3
1 2 4
```

Излаз

```
15
```

МЕЂУНАРОДНИ ПРОГРАМЕРСКИ ТУРНИР
24 новембар, 2012, Шумен, Бугарска
Јуниори

Задатак B2. МАКСИМАЛНА СУМА

Дат је низ од *n* кутија. У свакој кутији се налази неколико лопти. На свакој лопти је написан један цео број. Бирамо неке (1, 2, ...,, или све) кутије и узимамо једну лопту из сваке изабране кутије, чувајући редослед кутија. Потом поређамо све узете лопте у један ред, према редоследу кутија. Понекад, бројеви који су написани на узетим лоптама, могу формирати неопадајући низ. Напишите програм **`maxsum`**, који израчунава шта може бити највећа сума ових бројева.

Улаз

У првом реду се налази вредност броја *n*. Потом следи *n* редова, и сваки ред се односи на једну кутију. Сваки од ових редова почиње са бројем лопти у кутији, а потом следе бројеви написани на лоптама у тој кутији.

Излаз

Један цео број једнак максималној суми описаној горе у формулацији задатка.

Ограничења

$0 < n < 500$. У свакој кутији се не налази више од 50 лопти, али се налази макар једна лопта. Цео број, који је написан на лопти, припада сегменту од 1 до 1000.

Пример

Улаз

```
10
3 2 2 4
2 1 2
3 3 7 10
4 5 5 1 1
1 3
1 2
3 1 9 1
1 5
7 8 1 1 1 1 2 1
1 3
```

Излаз

```
25
```

Објашњење примера

Низ узетих лопти је 2 + 2 + 3 + 5 + 5 + 8. Из кутија број 5, 6, 7 и 10 није ништа узето.

МЕЂУНАРОДНИ ПРОГРАМЕРСКИ ТУРНИР
24 новембар, 2012, Шумен, Бугарска
Јуниори

Задатак B3. ПРОШИРИВАЊЕ КАНАЛА

У земљи Waterland, постоји n језера (нумерисаних од 1 до n) и m канала између њих. Позната је ширина сваког канала (у метрима). Кретање каналима се може извести у оба смера. Познато је да чамац ширине један метар може доспети до ма ког језера, почевши од језера са редним бројем 1.

Написати програм **`channels`**, који израчунава минимални број канала које треба проширити, тако да чамац ширине k метара може путовати између свака два језера (чамац се може померати од једног језера до другог, ако је његова ширина мања или једнака од ширине канала који повезује језера).

Улаз

У првој линији стандардног улаза су дати цели бројеви n и m ($1 < n \leq 1000$, $1 < m \leq 100000$).

У наредних m линија су дата три цела броја, i, j и w, који указују да постоји канал ширине w ($1 \leq w \leq 200$) између језера, i и j ($1 \leq i, j \leq n$).

У последњој линији је дат цео број k ($1 \leq k \leq 200$).

Излаз

У једином реду стандардног излаза испишите један цео број: минимални број канала који треба проширити.

Пример

Улаз

```
6 9
1 6 1
1 2 2
1 4 3
2 3 3
2 5 2
3 4 4
3 6 2
4 5 5
5 6 4
4
```

Излаз

```
2
```

Resenja:

1.

Задатак може да се реши проналажењем свих могућности за постављање тегова под условима описаним у формулацији задатака. Стога, генеришемо све пермутација бројева a1, a2, ... За сваку такви пермутацију покушавамо да стави другу тежину на леви или десни тас, уз проверу да ли укупне тежине на левом тасу нису веће него са десне стране.Предложено решење је да се користити рекурзивна реализација трагања тегова.

```
#include<iostream>
#include<algorithm>
using namespace std;

int a[10], n, ans = 0, lpan, rpan;

void set(int num)
{
   if(num == n)
   {
      ans++;
      return;
   }
   if (lpan + a[num] <= rpan)
   {
      lpan += a[num];
      set(num + 1);
      lpan -= a[num];
   }
   rpan += a[num];
   set(num + 1);
   rpan -= a[num];
}

int main()
{
   cin >> n;
   for(int i=0; i<n; i++)
     cin >> a[i];
   sort(a, a+n);
   do
   {
      lpan=0; rpan=0;
      set(0);

   }while(next_permutation(a,a+n));
   cout << ans << endl;
   return 0;
}
```

2.

```
#include<iostream>
#include<iomanip>
using namespace std;
const int N=1001;
const int M=101;
int n;
```

```
int a[N][M],s[N][M];
int main()
{

 a[0][0]=1;
 cin >> n;
 for(int i=1;i<=n;i++)
 { cin >> a[i][0];
   for(int j=1;j<=a[i][0];j++) cin >> a[i][j];
 }

 for(int j=1;j<M;j++) s[1][j]=a[1][j];

 for(int i=2;i<=n;i++)
 for(int j=1;j<=a[i][0];j++)
 {
   for(int ik=0;ik<i;ik++)
   for(int jk=1;jk<=a[ik][0];jk++)
   if(a[i][j]>=a[ik][jk])
    if(s[i][j]<s[ik][jk]+a[i][j]) s[i][j]=s[ik][jk]+a[i][j];
 }

 int r=0;
 for(int i=0;i<=n;i++)
  for(int j=0;j<=a[i][0];j++)
   if(r<s[i][j]) r=s[i][j];
 cout << r << endl;
}
```

3.

1 начин

Нађимо максимално покривајуће стабло T_{max} графа језера G и повежимо га каналима. После израчунавамо колко канала из T_{max} су ужи од дате вредности K. Нека је број таквих канала q. Ако се ови канали прошире, чамац ширине K може да прође између свака два језера. Штавише, ако уклонимо све канале уже од K, граф ће се разбити на $q+1$ компоненти повезаности и минимални број канала који ће повезати све компоненте јесте q.

2 начин

Тражимо број компоненти повезаности графа језера G_K и канала ширине барем K.

Нека је тај број $q+1$. Пошто је град G повезан, постојаће q тесних канала, који ће при проширивању до ширина K и додавањем у G_K повезати компоненте.

Проналажење компоненти повезаности у G_K може да се обави неким алгоритмом за обилазак графа – *BFS* или *DFS*.

```
#include <algorithm>
#include <cstdio>
#include <map>
#include <vector>
```

```
#include <queue>
#include <time.h>
#include <limits.h>
using namespace std;

const int MaxVertex=1001;
int E[MaxVertex][MaxVertex], D[MaxVertex], Pi[MaxVertex];
bool Marked[MaxVertex];
int N,M,K;


void input()
{
 int u,v,w;
 scanf("%d %d", &N, &M);
 for(int i=1; i<=N; i++)
  for (int j=1; j<=N; j++)
   E[i][j]=0;
 for(int i=1; i<=M; i++)
 {
  scanf("%d %d %d", &u, &v, &w);
  E[u][v]=w;
  E[v][u]=w;
 }
 scanf("%d", &K);
}

void CreateMST(int s)
{
 int u,v,w,cnt;
 for(int i=1; i<=N; i++)
  { D[i]=0; Pi[i]=-1; Marked[i]=false; }
 D[s]=INT_MAX; // MaxSpaningTree !!
 cnt=1; // koliko cvorova je u T, pocetak s je u T
 while (cnt<=N) {
  w=0; u=0;
  for(int i=1; i<=N; i++)
    if ((D[i]>w) && (!Marked[i]))
      {w=D[i]; u=i;}
  Marked[u]=true;
//   if (u != s) printf("%d %d %d\n",u, Pi[u], D[u]);
  for(int v=1; v<=N; v++)
    if ((D[v]<E[u][v]) && (!Marked[v]))
      {D[v]=E[u][v]; Pi[v]=u;}
  cnt++;
 }
// uredjeni par <v,Pi(v)> je u T,  sa tezinom d[v], za v != s
}
```

```
void SolveP2()
{
  int l;
  l=0;
  for (int v=1; v<=N; v++)
    if ((Pi[v]>0) && (D[v]<K))
      l++;
//  printf("Short_chanels = %d\n",l);
  printf("%d\n",l);
}

main()
{
  input();
  CreateMST(1);
  SolveP2();
}
```