```
}

int main()
{
  scanf("%d %d %lf", &m, &n, &d);
  for(int x=0;x<m;x++)
  for(int y=0;y<n;y++)
   b[x*x+y*y]=1;

  double e=1e10;
  int L=(m-1)*(m-1)+(n-1)*(n-1);
  int k0=0;
  for(int k=1;k<=L;k++)
   if(b[k])
    {double v=fabs(d-sqrt((double)k));
     if(v<e) {e=v; k0=k;}
    }

  double res=sqrt((double)k0);
  print(res);
}
```

Емил Келеведжиев

Задача B2. ОКРЪЖНОСТИ

В равнината са дадени N окръжности ($0 < N < 10000$). Никои две окръжности нито се пресичат, нито се допират, но някои могат да съдържат вътре в себе си други окръжности. Напишете програма **`circ`**, която извежда дължината на най-дългата последователност от вложени една в друга окръжности.

На първия ред на стандартния вход е записано числото N. Следват N реда, всеки съдържащ по три цели числа, разделени с интервали. Тези числа задават съответно координатите (абсциса и ордината) на центъра и радиуса на поредната окръжност. Координатите и радиусите са в диапазона от –1000 до 1000.

ПРИМЕР
Вход

```
5
1 1 1
20 20 13
15 15 5
15 15 2
20 20 1
```

Изход

```
3
```

РЕШЕНИЕ

Входните данни се прочитат в масив с елементи `c[i]`, съответстващи на дадените окръжностти чрез дефинираната в програмата структура `cir`. Функцията `create_graph()` намира за всяка окръжност `c[i]` номерата `j` на окръжностите, които се съдържат в нея, и ги записва последователно във вектора `c[i].next`. Едновременно се образува и обратната

релация – за всяка окръжност c[j] се определят номерата i на окръжностите, които тя съдържа и те се записват последователно във вектора c[j].prev.

Функцията topsort() извършва топологична сортировка на получения ориентиран ацикличен граф. Използва се стек за съхранение на такива върховете от графа (би могло да се използва опашка или множество), които в текущия момент от работата на алгоритъма нямат наследници, т.е. за тях променливата nv е равна на нула. На всяка стъпка в цикъла while се изважда от стека по един връх на графа, неговият номер се добавя в масива t и в стека се вкарват следващи върхове от графа, които са останали без наследници след премахването на влизащите дъги в предишно извадения връх от стека. Така накрая в масива t се нареждат според топологичната сортировка върховете на графа. Едновременно с този масив, функцията topsort() формира масива ti, които ще служи за обратно индексиране на t.

Решението на задачата се пресмята от функцията dp(), която по принципите на динамичното оптимиране пробягва върховете на графа според наредбата им зададена в масива t[i] и запълва стойности в масива p[i]. Тези стойности показват големината на най-дългия път по последователни дъги от графа, който се реализира от първия до i-тия връх, според топологичната наредба. Накрая резултатът се получава от най-голямата стойност, записана в масива p и добавяне на 1.

```
#include<cstdio>
#include<vector>
#include<stack>
using namespace std;

struct cir
{
 int x,y,r;
 vector<int> next, prev;
 int nv;
};
const int N=10001;
cir c[N];
int p[N], t[N], ti[N];
int n;

void create_graph()
{
 for(int i=1;i<=n;i++)
 for(int j=1;j<=n;j++)
 if(i!=j)
 {
    if(c[j].r < c[i].r)
    if
     (
      (c[i].x-c[j].x)*(c[i].x-c[j].x)+
      (c[i].y-c[j].y)*(c[i].y-c[j].y)<
       c[i].r*c[i].r+c[j].r*c[j].r-2*c[i].r*c[j].r
     )
   {
   c[i].next.push_back(j);
   c[j].prev.push_back(i);
   }
 }
 for(int i=1;i<=n;i++) c[i].nv=c[i].next.size();
}
```

```
void topsort()
{
 stack<int> s;
 for(int i=1;i<=n;i++)
  if(c[i].nv==0) {s.push(i); c[i].nv=-1;}
 int it=-1;
 while(!s.empty())
 {
  it++; t[it]=s.top();
  ti[s.top()]=it;
  int s0=s.top();
  s.pop();
  for(int j=0; j<c[s0].prev.size(); j++)
   c[c[s0].prev[j]].nv--;
  for(int i=1;i<=n;i++)
   if(c[i].nv==0) {s.push(i); c[i].nv=-1;}
 }
}
void dp()
{
 for(int i=0;i<n;i++)
 {
  p[i]=0;
  int vmax=0;
   for(int j=0;j<c[t[i]].next.size();j++)
    {
      int v=p[ti[c[t[i]].next[j]]];
      vmax=max(vmax,v);
    }
  if(!c[t[i]].next.empty()) p[i]=vmax+1;
 }
 int res=0;
 for(int i=0;i<n;i++) res=max(res,p[i]);
 printf("%d\n",res+1);
}
int main()
{
 scanf("%d",&n);
 for(int i=1;i<=n;i++)
  {scanf("%d %d %d", &c[i].x, &c[i].y, &c[i].r);}
 create_graph();
 topsort();
 dp();
}
```

Емил Келеведжиев

Задача B3. ТАБЛИЦА

Дадена е таблица от M реда и N стълба ($0 < M < 1000$, $0 < N < 1000$), съдържаща във всяка клетка по едно цяло число в диапазона от –100 до 100. Разглеждаме всички подтаблици с размери m на n, съставени от последователни редове и стълбове на дадената таблица ($0 < m < M$, $0 < n < N$). Напишете програма **tab**, която намира максималната сума на числата, записани в подтаблица от разглеждания вид.