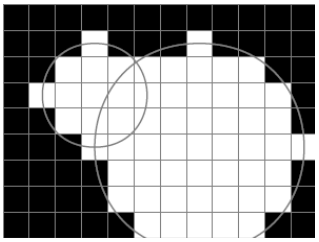
Zadatak 1 – Krugovi (3 sec, 256MB)

Andrija je napisao program koji crta n bijelih krugova na crnom monohromatskom ekranu rezolucije wxh piksela, gdje gornji lijevi ugao ima koordinate (0, 0). Krug sa centrom u tački (x_c, y_c) i poluprečnikom r sadrži sve tačke koje zadovoljavaju uslov $\sqrt{(x-x_c)^2+(y-y_c)^2} \le r^2$.



Dio kruga koji ne staje ekran se odrezuje (vidi sliku). Ako neka tačka pripada unutrašnjosti bar dva kruga, ona je bijele boje. Vaš zadatak je da izračunate koliko ima piksela crne boje poslije crtanja n krugova.

Ulaz: U prvom redu ulaza su tri cijela broja: w, h i n ($1 \le w$; $h \le 20\,000$; $1 \le n \le 100$). Sljedećih n redova sadrže po tri cijela broja: x_i, y_i, r_i ($0 \le x_i < w$; $0 \le y_i < h$; $0 \le r_i \le 40\,000$), koji označavaju koordinate centra i poluprečnik i-tog kruga.

Izlaz: U jedinom redu izlaza štampati broj preostalih crnih tačaka.

Napomena: Slika odgovara drugom primjeru.

Test primjeri:

Ulaz	Izlaz
5 3 2 1 1 1 3 1 1	6
12 9 2 3 3 2 7 5 4	51

Rješenje:

```
#include <cmath>
#include <cstdio>

#include <algorithm>
#include <iostream>
#include <utility>
#include <vector>

using namespace std;

long long dist( long long x1, long long y1, long long x2, long long y2 )
{
  return (x1 - x2) * (x1 - x2) + (y1 - y2) * (y1 - y2);
}

int main()
{
  //freopen("circles.in", "r", stdin);
  //freopen("circles.out", "w", stdout);

  int rx, ry, n;
  cin >> rx >> ry >> n;
  long long *x = new long long[n];
  long long *y = new long long[n];
  long long *r = new long long[n];
```

```
  for (int i = 0; i < n; i++)
  {
    cin >> x[i] >> y[i] >> r[i];
    r[i] = r[i] * r[i];
  }

  long long ans = 0;
  for (int cx = 0; cx < rx; cx++)
  {
    vector <pair <int, int> > events;
    for (int i = 0; i < n; i++)
    {
      if (dist(x[i], y[i], cx, y[i]) > r[i])
        continue;
      long long dy = max(0ll, (long long)sqrtl(1.0 * (r[i] - (x[i] - cx) * (x[i] -
cx))) - 10);
      while (dist(x[i], y[i], cx, y[i] + dy + 1) <= r[i])
        dy++;
      events.push_back(make_pair(max(y[i] - dy, 0ll), +1));
      events.push_back(make_pair(min(y[i] + dy + 1, (long long)ry), -1));
    }
    sort(events.begin(), events.end());
    int cur = 0, sum = 0, pos = 0;
    for (int i = 0; i < (int)events.size(); i++)
    {
      if (cur != 0)
        sum += (events[i].first - pos);
      pos = events[i].first;
      cur += events[i].second;
    }
    ans += sum;
  }
  cout << (long long)rx * (long long)ry - ans << endl;

  delete x;
  delete y;
  delete r;

  return 0;
}
```

Zadatak 2 – Par (2 sec, 256MB)

Za date prirodne brojeve a i b odrediti brojeve x i y, takve da je NZS(a, b) = NZS(x, y), HZD(a, b) = NZD(x, y) i y − x je minimalno. NZS je najmanji zajednički sadržalac a NZD je najveći zajednički djelilac.
Ulaz: U prvom redu ulaza su dva broja a i b ($1 \leq a, b \leq 10^9$).
Izlaz: Štampati dva prirodna broja x i y ($1 \leq x \leq y$) koja zadovoljavaju uslove zadatka.

Test primjeri

Ulaz	Izlaz
3 4	3 4
1 12	3 4

Rješenje:

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

```
#ifdef WIN32
    #define LLD "%I64d"
#else
    #define LLD "%lld"
#endif

template <typename T>
T abs(T x) {
    return x > 0 ? x : -x;
}

int gcd(int a, int b) {
    while (b) {
        a %= b;
        std::swap(a, b);
    }
    return a;
}

long long lcm(int a, int b) {
    return 1LL * a / gcd(a, b) * b;
}

long long power(long long x, int p) {
    long long ans = 1;
    for (int i = 0; i < p; i++)
        ans *= x;
    return ans;
}

void factorize(long long number, std::vector <long long> &primes) {
    primes.clear();
    for (long long i = 2; i * i <= number; ++i) {
        int cnt = 0;
        while (!(number % i))
            ++cnt, number /= i;
        if (cnt)
            primes.push_back(power(i, cnt));
    }
    if (number > 1)
        primes.push_back(number);
}

long long bestAns = 0, bestX, bestY;
long long x = 1, y = 1;

void go(int pos, std::vector <long long> &primes) {
    if (pos == (int)primes.size()) {
        if (bestAns > abs(x - y)) {
            bestAns = abs(x - y);
            bestX = x;
            bestY = y;
        }
        return;
    }

    x *= primes[pos];
    go(pos + 1, primes);
    x /= primes[pos];
```

```
    y *= primes[pos];
    go(pos + 1, primes);
    y /= primes[pos];
}

int main() {
    freopen("gcm.in", "r", stdin);
    freopen("gcm.out", "w", stdout);
    int a, b;
    scanf("%d%d", &a, &b);

    int d = gcd(a, b);
    std::vector <long long> primes;
    factorize(lcm(a, b) / d, primes);

    bestAns = abs(b - a) / d;
    bestX = a / d, bestY = b / d;
    go(0, primes);

    if (bestX > bestY)
        std::swap(bestX, bestY);
    printf(LLD" "LLD"\n", bestX * d, bestY * d);
    return 0;
}
```

Zadatak 3 – Sadržaoci

Dat je niz prirodnih brojeva. Pronađite dužinu najdužeg rastućeg podniza u kojem je svaki broj sadržalac prethodnog broja.

Ulaz: U prvom redu nalazi se cijeli broj N ($1 \leq N \leq 100000$), dužina niza. U svakom od sljedećih N redova nalazi se po jedan prirodan broj manji od 1000000, elementi datog niza.

Izlaz: U prvi i jedini red štampaj dužinu najdužeg rastućeg podniza u kojem je svaki broj sadržalac prethodnog broja.

Test primjeri

Ulaz	**Ulaz**
5	5
1	3
2	9
3	6
4	3
5	15
Izlaz	**Izlaz**
3	2

Rješenje:

Idemo po nizu slijeva nadesno, i tražimo koji je najduži podniz koji završava na trenutnom elementu. Prethodni element podniza mogao je biti bilo koji broj koji se pojavio prije u nizu, a da je djelilac broja na kojem smo trenutno. Provjeravanje za svaki broj je li djelilac bi nas dovelo do složenosti $O(N^2)$, što je presporo. Bolje je provjeravati samo za djelioce trenutnog broja, koji se mogu brzo generisati uz pomoć Eratostenovog sita. Takođe, dok prolazimo kroz niz, u odvojenom nizu na mjestu i pamtimo koji je najduži podniz koji završava brojem i.

```
#include <cstdio>
#include <vector>

#define maxn 100000
#define maxm 1000001

using namespace std;

int n, a[maxn];
int sito[maxm];

int dp[maxm];

vector< pair< int, int > > v;
int sol;

void load() {
      scanf( "%d", &n );
      for( int i = 0; i < n; ++i ) scanf( "%d", &a[i] );
}

void init_sito() { // eratostenovo sito, ali za svaki broj pamtimo koji je najmanji
prosti faktor
      for( int i = 2; i < maxm; ++i )
            if( sito[i] == 0 )
                  for( int j = i; j < maxm; j += i )
                        if( sito[j] == 0 ) sito[j] = i;
}

void gen( int m, int x, int k ) { // rekurzivno generisemo sve djelioce broja m
      if( k == v.size() ) {
            if( x != m ) {

                  if( dp[x] > sol ) sol = dp[x];

            }
            return;
      }

      gen( m, x, k+1 );

      for( int i = 0; i < v[k].second; ++i ) {
            x *= v[k].first;
            gen( m, x, k+1 );
      }
}
```

```
void rastavi( int x ) { // rastavi svaki broj na faktore
	v.clear();
	for(;x>1;) {
		int p = sito[x], k = 0;
		for(; x%p == 0; ++k ) x /= p;
		v.push_back( make_pair( p, k ) );
	}
}

void solve() {
	for( int i = 0; i < n; ++i ) {
		sol = 0;
		rastavi( a[i] );
		gen( a[i], 1, 0 );
		if( sol+1 > dp[ a[i] ] ) dp[ a[i] ] = sol+1;
	}

	sol = 0;
	for( int i = 0; i < maxm; ++i )
		if( sol < dp[i] ) sol = dp[i];
	printf( "%d\n", sol );

}

int main(void) {
	init_sito();
	load();
	solve();
	return 0;
}
```