Практикум з розв’язування задач

1. Задача 1 (III етап Всеукраїнської олімпіади з інформатики 2013-2014 н.р.**)**

A. "Все, Степан! Ти мене дістав!"

Input file name:

bubble.in

Output file name:

bubble.out

Time limit:

100 ms

Memory limit:

256 M

Степан нещодавно відпочивав у Японії і привіз звідти нову жувальну гумку. На першій парі в університеті він поділився гумкою зі своїм товаришем. Дочекавшись моменту, коли лектор повернувся до дошки, на рахунок "три - чотири" хлопці дружньо почали надувати бульбашки. Відомо, що Степан надуває бульбашку до максимально можливого розміру за час t_1, після чого бульбашка миттєво лопається, і Степан починає надувати бульбашку заново з тією ж швидкістю. Товариш Степана робе те ж саме за час t_2.

Весь цей час викладач настільки захоплений доведенням теореми, що взагалі нічого не чує. І тільки коли обидві бульбашки лопнуть одночасно, викладач почує шум і обернеться. І тоді вже точно студентам попаде на горіхи, а більше усього тому, хто приніс на пару жувальні гумки.

Визначте, скільки часу хлопці можуть насолоджуватись надуванням бульбашок, не замічені викладачем.

Наприклад, якщо $t_1 = 2$, $t_2 = 3$, то буде відбуватись наступне:

Степан надуває бульбашку з моменту часу $t = 0$ до моменту часу $t = 2$, потім бульбашка лопається, і він надуває бульбашку заново - з моменту часу
$t = 2$
до моменту часу
$t = 4$
, а потім ще раз - з моменту часу
$t = 4$
до
$t = 6$
.

Товариш Степана надуває бульбашку з $t = 0$ до $t = 3$ і ще раз з $t = 3$ до $t = 6$.

В момент часу $t = 6$ бульбашки лопаються одночасно в обох студентів, викладач

повертається і каже: "Все, Степан! Ти мене дістав!".

Формат вхідних даних: Перший рядок вхідного файлу містить два цілих числа t_1, t_2 ($1 \le t_1, t_2 \le 10^9$).

Формат вихідних даних: Вихідний файл повинен містити одне ціле число - час, протягом якого Степан з товаришем можуть насолоджуватись надуванням бульбашок.

Приклад вхідних та вихідних даних:

bubble.in

bubble.out

2 3

6

1 16

16

Підхід

Програмний код

Тести

Підбір в циклі з перевіркою остачі від ділення

```
nsk=a;
```

```
while (nsk%a!=0||nsk%b!=0)nsk++;
```

Зараховано 63.6%

Підбір в циклі з перевіркою остачі від ділення

```
for (int i=1;i<=a;i++)
```

```
if(a%i==0&&b%i==0)nsd=i;
```

```
nsk=a*b/nsd;
```

Зараховано 63.6%

Алгоритм Евклідіа

```
while (b!=0)
```

```
{t=a%b;
```

```
a=b;
```

```
b=t;
```

```
}
```

Зараховано 100%

Алгоритм Евклідіа

```
while (b!=0)
```

```
{a=a%b;
```

```
swap(a,b);
```

```
}
```

```
Зараховано 100%
```

http://www.e-olimp.com.ua/problems/7239

2. Задача 2 (IV етап Всеукраїнської олімпіади з інформатики 2013-2014 н.р.**)**

Відрізки (Роман Рубаненко, Роман Фурко)

Умова. Петрик дуже любить іграшки у формі геометричних фігур. Нещодавно він помітив, що серед його іграшок немає жодного трикутника. Це дуже засмутило Петрика, тому він пішов до найближчого магазину, щоб придбати новісінький трикутник. В магазині Петрику сказали, що всі трикутники вже давно розкупили, але в наявності є N прямих відрізків.

Відрізки пронумеровані послідовними натуральними числами, починаючи з одиниці. Відрізок номер i характеризується двома числами — довжиною Li та ціною Ci . Петрик дуже розумний, тому знає, що бажаний трикутник він може скласти з трьох відрізків. Більше того, наш герой знає, що трикутник можливо скласти лише з таких відрізків, що довжина будь-якого з них має бути строго меншою за сумарну довжину інших двох. Отже, хлопчик вирішив придбати рівно три таких відрізки. Звичайно, він хоче заощадити якомога більше коштів на морозиво, тому хоче витратити якнайменше на покупку відрізків для свого трикутника.

Завдання. Напишіть програму segments , яка за інформацією про відрізки визначить мінімальну вартість трьох відрізків, з яких хлопчик зможе скласти трикутник, або

визначить, що це зробити неможливо.

Вхідні дані. В першому рядку вхідного файла segments.dat записано єдине число N — кількість відрізків. Далі в N рядках записана інформація про самі відрізки. Кожен такий рядок містить відповідні Li (1 Li 10^9) та Ci . Ціни утворюють перестановку чисел від 1 до N, тобто є попарно різними натуральними числами, не більшими за N.

Вихідні дані. Вихідний файл segments.sol має містити єдине число — мінімальну вартість трьох відрізків, з яких можна скласти трикутник, або «–1» (лапки для наочності) в тому випадку, якщо вибрати рівно три такі відрізки неможливо.

Оцінювання. Набір тестів складається з 4 блоків, для яких додатково виконуються такі умови:

1 N 100.

210 < N 3000.

3000 < N 10^4 .

о 30 балів: 10^4 < N 10^5 .

Приклади вхідних та вихідних даних.

segments.dat

segments.sol

4

1 1

2 2

3 3

4 4

9

3

3 1

5 3

10 2

-1

Підхід

Програмний код

Тести

Розглянемо найпростіший алгоритм розв’язання задачі. Перебиратимемо всі трійки відрізків.

Для кожної трійки треба перевірити можливість існування відповідного трикутника. Якщо три-

кутник існує, підрахуємо сумарну вартість трьох відрізків, а з усіх знайдених вартостей вибе-

ремо найменшу. Складність алгоритму — O(N).

```
for ( int
```

```
for ( int
```

for	(	int

if	(l[i]<l[j]+l[k] && l[j]<l[i]+l[k] && l[k]<l[i]+l[j])

if	(c[i]+c[j]+c[k]<m)m=c[i]+c[j]+c[k];

А от швидше розв’язання. Розглядатимемо всі відрізки в порядку від найкоротших до найдовших

найбільша сумарна довжина двох відрізків (з числа перших k відрізків) із загальною вартістю c.

Тепер розглянемо оптимальне розв’язання. Воно базується на такій нескладній властивості:

Властивість. Якщо серед натуральних чисел L1 , L2 , ..., Lk немає трьох таких, що утворюють сто

#pragma	comment	(linker,

#include	<algorithm>

#include	<memory.h>

#include	<cstdio>

```
#include <iostream>

#include <cmath>

#include <string>

#include <cassert>

#include <map>

#include <set>

#include <vector>

#include <queue>

using namespace std;

#define prev privet1

#define next privet2

#define y1 privet3
```

```
#define rank privet4

#define left privet5

#define right privet6

#define y0 privet7

const double pi = 3.14159265358979323

const int INF = 1 << 30, ENOUGH_V

long long a[ENOUGH_VALUE_OF_N

int main()

{

freopen( "segments.dat" ,

freopen( "segments.sol" ,

int n, i, j, u;
```

```
scanf( "%d", &n);

long long len;

int cost;

for (i = 1; i <= n; i++)

{

scanf( "%lld%d", &len, &cost);

if (cost <= ENOUGH_VALUE_OF_N) a[cost] = len;

}

n = min(n, ENOUGH_VALUE_OF_N);

int ans = INF;

for (i = 1; i <= n; i++)

for (j = i + 1; j <= n; j++)
```

```
for (u = j + 1; u <= n; u++)

if (i + j + u < ans && a[i] < a[j] + a[u] && a[j] < a[i] + a[u] & a[u] < a[i]

if (ans == INF) ans = -1;

printf( "%d\n" , ans);

}
```

**http://nvk26.lutsk.ua/cgi-bin/new-client?contest_id=3 – II етап (районний) Всеукраїнської олімпіади з інформатики 2012-2013 н.р. (логін user400-user430;
пароль 400-430).**

**http://nvk26.lutsk.ua/cgi-bin/new-client?contest_id=11 – III етап (обласний) Всеукраїнської олімпіади з інформатики 2012-2013 н.р. (логін user400-user430;
пароль 400-430).**

**http://nvk26.lutsk.ua/cgi-bin/new-client?contest_id=12 – тренувальний тур по підготовці до I V етапу Всеукраїнської олімпіади з інформатики 2012-2013 н.р. (логін user400-user414;
пароль 400-414).**

http://nvk26.lutsk.ua/cgi-bin/new-client?contest_id=15 – Підготовка до олімпіади 2013 (логін user400-us er430; паро

ль 400-430).

http://nvk26.lutsk.ua/cgi-bin/new-client?contest_id=22 - Тренувальний турнір до міської олімпіади з інформатики (програмування) 2013 (м.Луцьк)

http://nvk26.lutsk.ua/cgi-bin/new-client?contest_id=25 - II етап Всеукраїнської учнівської олімпіади з інформатики (м.Луцьк) 2013-2014н.р.

http://nvk26.lutsk.ua/cgi-bin/new-client?contest_id=26 - Волинська обласна Інтернет-о лімпіада 2013

http://nvk26.lutsk.ua/cgi-bin/new-client?contest_id=27 - Тренувальний турнір до обласної олімпіади 2014

http://nvk26.lutsk.ua/cgi-bin/new-client?contest_id=28 - III етап Всеукраїнської учнівської олімпіади з інформатики (м.Луцьк) 2013-2014 н.р.

http://nvk26.lutsk.ua/cgi-bin/new-client?contest_id=29 - Підготовка до IV етапу олімпіади з інформатики (м.Луцьк) 2013-2014н.р.