Управління освіти і науки
Чернігівської обласної державної адміністрації

Чернігівський обласний інститут післядипломної
педагогічної освіти імені К.Д. Ушинського

ЗБІРНИК ЗАДАЧ ТА РОЗВ'ЯЗКІВ II ЕТАПУ ВСЕУКРАЇНСЬКОЇ УЧНІВСЬКОЇ ОЛІМПІАДИ З ІНФОРМАТИКИ

2018-2019 навчального року

Чернігів – 2018

Автори:
Бондаренко С. М., Зуб В. В., Коваленко О. І., Хрол Н. П., Нємкевич Д. О., Солонко Н. В., Потьомкін Л. Є.

Рецензенти:
Горошко Ю. В., завідувач кафедри інформатики і обчислювальної техніки Національного університету «Чернігівський колегіум» імені Т. Г. Шевченка, доктор педагогічних наук, професор

Покришень Д. А., завідувач кафедри природничо-математичних дисциплін та інформаційно-комунікаційних технологій в освіті Чернігівського обласного інституту післядипломної педагогічної освіти імені К.Д.Ушинського, кандидат педагогічних наук, доцент

Загальна редакція:
Літош Ю. М., старший викладач кафедри природничо-математичних дисциплін та інформаційно-комунікаційних технологій в освіті Чернігівського обласного інституту післядипломної педагогічної освіти імені К.Д. Ушинського

Баранова О. Є., методист відділу природничо-математичних дисциплін Чернігівського обласного інституту післядипломної педагогічної освіти імені К. Д. Ушинського

Смірнова О. М., методист відділу природничо-математичних дисциплін Чернігівського обласного інституту післядипломної педагогічної освіти імені К. Д. Ушинського

Збірник задач та розв’язків ІІ етапу Всеукраїнської учнівської олімпіади з інформатики 2018-2019 навчального року/ С. М. Бондаренко, В. В. Зуб, О. І. Коваленко та ін.; за заг. ред. Ю. М. Літоша, О. Є. Баранової, О. М. Смірнової. – Чернігів: ЧОІППО імені К. Д. Ушинського, 2019 – 29 с.

Збірник містить задачі та розв’язки ІІ етапу Всеукраїнської учнівської олімпіади з інформатики, що проводилась у Чернігівській області у 2018-2019 навчальному році.

Матеріали розраховані на вчителів інформатики, слухачів курсів підвищення кваліфікації. Можуть бути використані учнями 7-11 класів для підготовки до олімпіад.

Рекомендовано до друку
вченою радою Чернігівського обласного інституту
післядипломної педагогічної освіти імені К. Д. Ушинського
(протокол № 1 від 28.02.2019 р.)

ЗМІСТ

Задача A – Зачіска Леді …….. 4

Задача B – Пароль ……...… 7

Задача C. – Леді і частота послідовності …… 12

Задача D – Максимальний прямокутник ……... 15

Задача E – Друзі Степана ……... 23

Задача F – Квіти ……... 24

Задача G – Факультатив з інформатики …… 25

Задача H – Цукерки ……. 26

Задача I – Степан і похід в магазин ……. 28

Задача A - Зачіска Леді

Назва вхідного файлу: **standard input**

Назва вихідного файлу: **standard output**

Ліміт часу: **1 second**

Ліміт використання пам'яті: **256 megabytes**

Всім відомо, що Леді слідкує за тим, які зачіски зараз у тренді. Найпопулярніша зачіска цього року - дві косички з вплетеною стрічкою у кожну. Мода непередбачувана, тому дуже важливо, щоб стрічки були різних кольорів. У Леді є n червоних та m жовтих стрічок. Кожного ранку Леді буде намагатися зробити зачіску з двома різними стрічками, проте, якщо такої можливості не буде, то дві косички з однаковими стрічками також непогано виглядають. Використовувати одну й ту саму стрічку 2 рази вона не може.

Формат вхідних даних:
Дано два числа n та m ($1 \le n, m \le 100$).

Формат вихідних даних:
Виведіть два числа - максимальну кількість днів, яку Леді буде мати зачіску з двома різними стрічками та кількість днів після цього, яку Леді буде мати зачіску з однаковими стрічками, поки вони не закінчаться або стрічка не буде мати пари.

Приклади

standard input	standard output
3 1	1 1
2 3	2 0
7 3	3 2

Ідея розв'язання задачі

Хрол Н. П., ***учителя інформатики загальноосвітньої спеціалізованої школи I-III ступенів фізико-математичного профілю № 12 м. Чернігова***

Леді зможе мати зачіску з двома різними стрічками min(n,m) днів, а зачіску з двома однаковими стрічками (max(n,m)-min(n,m))/2 днів.

```
#include <iostream>
using namespace std;
int main()
{
   int n,m;
   cin>>n>>m;
   if (n<m)    cout<<n<<" "<<(m-n)/2;
      else      cout<<m<<" "<<(n-m)/2;
   return 0;
}
```

Ідея розв'язання задачі

Солонка Н. В., ***учня 11 класу ліцею № 15 м. Чернігова***

```
#include <iostream>

using namespace std;

int main(int argc, char** argv) {
        int n, m, ans1, ans2;
        cin >> n >> m;
        if (n >= m) {
                ans1 = m;
                ans2 = (n - m) / 2;
        }
        if (m >= n) {
                ans1 = n;
                ans2 = (m - n) / 2;
        }
        cout << ans1 << " " << ans2;
        return 0;
}
```

Ідея розв'язання задачі

Зуба В. В., ***учителя математики, директора Прилуцької загальноосвітньої школи І-ІІІ ступенів № 7 Прилуцької міської ради Чернігівської області;***

Бондаренка С. М., ***учителя математики та інформатики Прилуцької загальноосвітньої школи І-ІІІ ступенів № 7 Прилуцької міської ради Чернігівської області***

```
  var n,m,k,l:longint;
begin
  read(n,m);
  if n>m then l:=m else l:=n; {Вибір меншого з вхідних чисел – кількість
днів з різними стрічками}
  k:=abs(m-n) div 2; {Кількість днів з однаковими стрічками}
  write(l,' ',k);
end.
```

Ідея розв’язання задачі

Коваленко О. І., ***учителя інформатики та математики спеціалізованої загальноосвітньої школи №2 І-ІІІ ступенів з поглибленим вивченням іноземних мов м. Чернігова***

Для розв'язку цієї задачі нам треба:

1). Зчитати дані.

2). За допомогою першого while, знайти кількість днів, в які ми заплітаємо різні стрічки.

3). Знайти, яка стрічка в нас залишалася після while, для цього знайдемо максимум між кількістю двох стрічок.

4). За домопогою другого while, знайдемо кількість днів, в які ми заплітаємо однакові стрічки.

```
#include <iostream>
using namespace std;
int main()
{
   int n,m,ans = 0; // оголошення зміних
   cin >> n >> m; // зчитування даних
   while(n > 0 && m > 0){ // знайдемо кількість днів, в які ми будемо
заплітати різні стрічки
      ans++; // додаємо до відповіді 1
      n--; // віднімаємо від кількості червоних стрічок одну
      m--; // віднімаємо від кількості синіх стрічок одну
   }
   cout << ans << " "; // виводимо кількість днів, в які вона буде заплітати
різні стрічки
   ans = 0; // обнуляємо відповідь
   m = max(n,m); // знаходимо кількість стрічок, які залишилися після
перших дій
   while(m > 0 && m - 2 >= 0){ //// знайдемо кількість днів, в які ми
будемо заплітати однакові стрічки
      ans++; // додаємо до відповіді 1
```

```
        m -= 2; // віднімаємо від кількості стрічок, що залишилися, два
    }
    cout << ans; // виводимо другу відповідь
    return 0;
}
```

Задача B - Пароль

Назва вхідного файлу:	**standard**	**input**
Назва вихідного файлу:	**standard**	**output**
Ліміт часу:	**1 second**	
Ліміт використання пам'яті:	**256 megabytes**	

Журі олімпіади вирішило, що пароль учасників олімпіади буде складатись з семи цифр та спеціального символу, який записується в кінці пароля. Цей символ визначається спеціальним методом, який полягає в наступному:

1. Помножте кожну цифру пароля на відповідну вагу. Вага першої цифри становить 2, вага другої цифри становить 7, третьої - 6, четвертої - 5, п'ятої - 4, шостої - 3, сьомої - 2.
2. Додайте усі отримані добутки.
3. Знайдіть залишок від ділення на 11.
4. Замініть залишок на велику літеру: 0 = «**J**», 1 = «**A**», 2 = «**B**», 3 = «**C**», 4 = «**D**», 5 = «**E**», 6 = «**F**», 7 = «**G**», 8 = «**H**», 9 = «**I**», 10 = «**Z**».

Наприклад, у нас є сім цифр паролю «6830907», тоді маємо 2 • 6 + 7• 8 + 6 • 3 + 5 • 0 + 4 • 9 + 3 • 0 + 2 • 7 = 12 + 56 + 18 + 0 + 36 + 0 + 14 = 136. Число 136 при діленні на 11, дає залишок 4, якій відповідає літера «**D**». Отже, фактично пароль повинен бути «6830907D».

Напишіть програму, яка за даними сімома цифрами пароля, визначить літеру, яку потрібно дописати в кінець пароля.

Формат вхідних даних:

У першому рядку задано сім цифр пароля.

Формат вихідних даних:

Виведіть в першому рядку одну літеру, яку потрібно дописати до пароля.

Приклади

standard input	standard output
6830907	D

Ідея розв'язання задачі

Хрол Н. П., ***учителя інформатики загальноосвітньої спеціалізованої школи I-III ступенів фізико-математичного профілю № 12 м. Чернігова***

Нємкевич Д. О., ***учениця 10 класу загальноосвітньої спеціалізованої школи I-III ступенів фізико-математичного профілю № 12 м. Чернігова***

Будемо зберігати в двох константах: масив ***B*** – вагу кожної цифри, ***s*** – послідовність літер, якими потрібно замінити залишок від ділення на 11.

Зчитуємо послідовно 7 символів в змінну ***a***, переводимо її на цифри, враховуючи вагу кожного символу. Знаходимо суму цифр в змінну ***ans***, та знаходимо залишок від ділення на 11.

З рядка ***s*** виводимо відповідну літеру ***s[ans]***.

```
#include <bits/stdc++.h>
using namespace std;
const string s = "JABCDEFGHIZ";
int B[7] = {2, 7, 6, 5, 4, 3, 2};
int main()
 {
   char a;
   int ans = 0;
   for (int i = 0; i < 7; i++)
 {
      cin >> a;
      ans += B[i]*(a-'0');
   }
   ans %= 11;
   cout << s[ans];
   return 0;
}
```

Ідея розв'язання задачі

Солонка Н. В., ***учня 11 класу ліцею № 15 м. Чернігова***

```
#include <iostream>

using namespace std;
```

```
int main(int argc, char** argv) {
	int n, i;
	int arr[7];
	char temp;
	for (i = 0; i < 7; i++) {
		cin >> temp;//зчитування паролю по цифрі
		arr[i] = int(temp) - 48;//Перевод символу цифри у
		//відповідне їй число
	}
	//Виконання першого пункту умови:
	arr[0] *= 2;
	arr[1] *= 7;
	arr[2] *= 6;
	arr[3] *= 5;
	arr[4] *= 4;
	arr[5] *= 3;
	arr[6] *= 2;
	n = 0;
	for (i = 0; i < 7; i++)
		n += arr[i];
	n %= 11;
	switch (n) {
	case(0):
		temp = 'J';
		break;
	case(1):
		temp = 'A';
		break;
	case(2):
		temp = 'B';
		break;
	case(3):
		temp = 'C';
		break;
	case(4):
		temp = 'D';
		break;
	case(5):
		temp = 'E';
		break;
	case(6):
		temp = 'F';
		break;
	case(7):
```

```
                temp = 'G';
                break;
        case(8):
                temp = 'H';
                break;
        case(9):
                temp = 'I';
                break;
        case(10):
                temp = 'Z';
                break;
        }
        cout << temp;
        return 0;
}
```

Ідея розв’язання задачі

Зуба В. В., ***учителя математики, директора Прилуцької загальноосвітньої школи І-ІІІ ступенів № 7 Прилуцької міської ради Чернігівської області;***

Бондаренка С. М., ***учителя математики та інформатики Прилуцької загальноосвітньої школи І-ІІІ ступенів № 7 Прилуцької міської ради Чернігівської області***

```
  const a:array[1..7] of byte = (2,7,6,5,4,3,2);
       b:array[0..10]of char = ('J','A','B','C','D','E','F','G','H','I','Z');
  var i,n,x:longint; s:int64;
begin
  read(n);
  s:=0;
  for i:=7 downto 1 do {Знаходження суми добутків}
    begin
      x:=n mod 10;
      s:=s+x*a[i];
      n:=n div 10;
    end;
  write(b[s mod 11]); {Виведення результату}
end.
```

Ідея розв’язання задачі

Коваленко О. І., ***учителя інформатики та математики спеціалізованої загальноосвітньої школи №2 І-ІІІ ступенів з поглибленим вивченням іноземних мов м. Чернігова***

Для розв'язку цієї задачі нам треба:
1). Зчитати цифри.
2). Додати до результату першу та останню цифру.
3). Додати через цикл всі інші числа до результату.
4). Знайти, яку букву нам треба вивести.

```
#include <iostream>
using namespace std;
int main()
{
    int n; //
    cin >> n; // зчитування данних
    int ans = (n / 1000000) * 2; n %= 1000000; ans += (n % 10) * 2; n /= 10; //
додаємо до відповіді перше та останнє число
    for(int i = 3; i <= 7; i++){ // за допомогою цикла додамо до відповіді всі
інші значення
        ans += (n % 10) * i; // додаємо значення
        n /= 10; // зменшуємо наше число
    }
    ans %= 11; // знаходимо залишок від ділення
    if(ans == 0) cout << "J"; // якшо наш залишок дорівнює 0, виводимо J
    else{
        if(ans == 10){ // якшо наш залишок дорівнює 10, виводимо Z
            cout << "Z";
            return 0;
        }
        char b = 'A' - 1; // змінна, якій присвоїмо результат
        b += ans; // змінюємо результат на потрібну букву
        cout << b; // виводимо її
    }
    return 0;
}
```

Задача C - Леді і частота послідовності

Назва вхідного файлу: **standard input**

Назва вихідного файлу: **standard output**

Ліміт часу: **1 second**

Ліміт використання пам'яті: **256 megabytes**

Леді отримала у подарунок послідовність цілих чисел. У послідовності цілих чисел кількість разів, яку повторюється ціле число, називають частотою цілого числа. З урахуванням послідовності цілих чисел, допоможіть Леді порахувати найвищу частоту.

Формат вхідних даних:

У першому рядку задано число N ($1 \le N \le 10000$) - кількість цілих чисел у послідовності. Кожен з наступних рядків містить одне ціле число послідовності A_i ($1 \le A_i \le 1000$).

Формат вихідних даних:

Виведіть в першому рядку одне ціле число - найвищу частоту послідовності.

Приклади

standard input	standard output
12 1 2 5 6 3 7 11 345 754 2 5 2	3
7 2 4 6	2

7 7 2 4	

Ідея розв'язання задачі

Хрол Н. П., ***учителя інформатики загальноосвітньої спеціалізованої школи І-ІІІ ступенів фізико-математичного профілю № 12 м. Чернігова***

Нємкевич Д. О., учениця 10 класу загальноосвітньої спеціалізованої школи І-ІІІ ступенів фізико-математичного профілю № 12 м. Чернігова

В елементі масиву ***A[i]*** будемо зберігати не число, а кількість повторів числа ***i*** в цьому масиві.

Найбільший елемент масиву ***A*** і буде відповіддю задачі.

```
#include <bits/stdc++.h>
using namespace std;
int A[1010];
int main() {
   int n, t;
   cin >> n;
   for (int i = 0; i < n; i++)
  {
      cin >> t;
      A[t]++;
   }
   int ans = -1;
   for (int i = 1; i <= 1000; i++)
      ans = max(ans, A[i]);
   cout << ans;
   return 0;
}
```

Ідея розв'язання задачі

Солонка Н.В., ***учня 11 класу ліцею № 15 м. Чернігова***

```
#include <iostream>

using namespace std;

int main(int argc, char** argv) {
        int n, i, j, counter, ans = 1;
```

```
        cin >> n;
        int *arr = new int[n];
        for (i = 0; i < n; i++)
                cin >> arr[i];
        for (i = 0; i < n - 1; i++) {
                counter = 1;
                for (j = 1; j < n - i; j++) {
                        if (arr[i] == arr[j + i])
                                counter++;
                }
                if (counter > ans)
                        ans = counter;
        }
        cout << ans;
        delete[]arr;
        return 0;
}
```

Ідея розв'язання задачі

Зуба В. В., ***учителя математики, директора Прилуцької загальноосвітньої школи І-ІІІ ступенів № 7 Прилуцької міської ради Чернігівської області;***

Бондаренка С. М., ***учителя математики та інформатики Прилуцької загальноосвітньої школи І-ІІІ ступенів № 7 Прилуцької міської ради Чернігівської області***

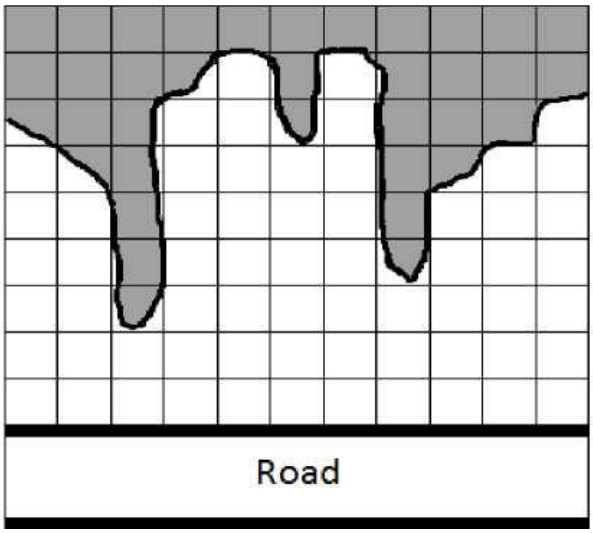
```
var n, i, l, max: longint;  a:array[1..1000]of longint;  {Таблиця частоти даних чисел}
begin
read(n);
for i:=1 to n do
  begin  read(l);   inc(a[l]);  end; {Введення вхідних даних у таблицю}
max:=0;
for i:=1 to 1000 do
  if a[i]>max then max:=a[i]; {Пошук найбільшого числа в таблиці}
write(max);
end.
```

```
#include <iostream>
#include <vector>
using namespace std;
int main()
{
   int n;
   cin >> n; // зчитуємо кількість чисел
   vector <int> col(1001, 0); // створюємо масив на 1001 елемент і заповнюємо його 0
   for(int i = 0; i < n; i++){ // за домопогою циклу зчитуємо числа
      int f;
      cin >> f; // зчитуємо число
      col[f]++; // в нашому масиві до цього числа додаємо 1
   }
   int mx = 0;
   for(int i = 1 ; i <= 1000; i++){
      mx = max(mx,col[i]); // знаходимо максимальне число серед 1000
   }
   cout << mx; // виводимо максимум
   return 0;
}
```

Задача D - Максимальний прямокутник

Назва вхідного файлу:	**standard input**
Назва вихідного файлу:	**standard output**
Ліміт часу:	**1 second**
Ліміт використання пам'яті:	**256 megabytes**

Загальновідомо, що будівництво готелю біля морського узбережжя є дуже вигідним. Ось чому компанія Ledy Invest купила ділянку землі на узбережжі Чорного моря (як показано на малюнку) і хотіла б побудувати найбільший готель. З різних причин готель повинен бути з прямокутним підвалом. Саме тому компанія шукає того, хто зможе знайти прямокутник максимальної площі, який може бути намальовано на землі. З цією метою місцевість була розділена на N стовпчиків рівних квадратів (білим кольором на малюнку). Стовпчики позначаються послідовно 1, 2, ..., N зліва направо, а прямокутник повинен складатися з цілого числа таких квадратів. Тоді для кожного стовпця було пораховано кількість цілих квадратів у ньому. Напишіть програму, яка знайде площу максимального прямокутника на місцевості, яка може складатися з квадратів.



Формат вхідних даних:

Перший рядок містить ціле число ціле число N ($1 \le N \le 1,000,000$).

У наступному рядку задано N цілих чисел $D_1, D_2, ..., D_N$, де D_i ($1 \le D_i \le 15,000$) - число квадратів у стовпці i.

Формат вихідних даних:

Виведіть одне число - площу максимального прямокутника.

Приклади

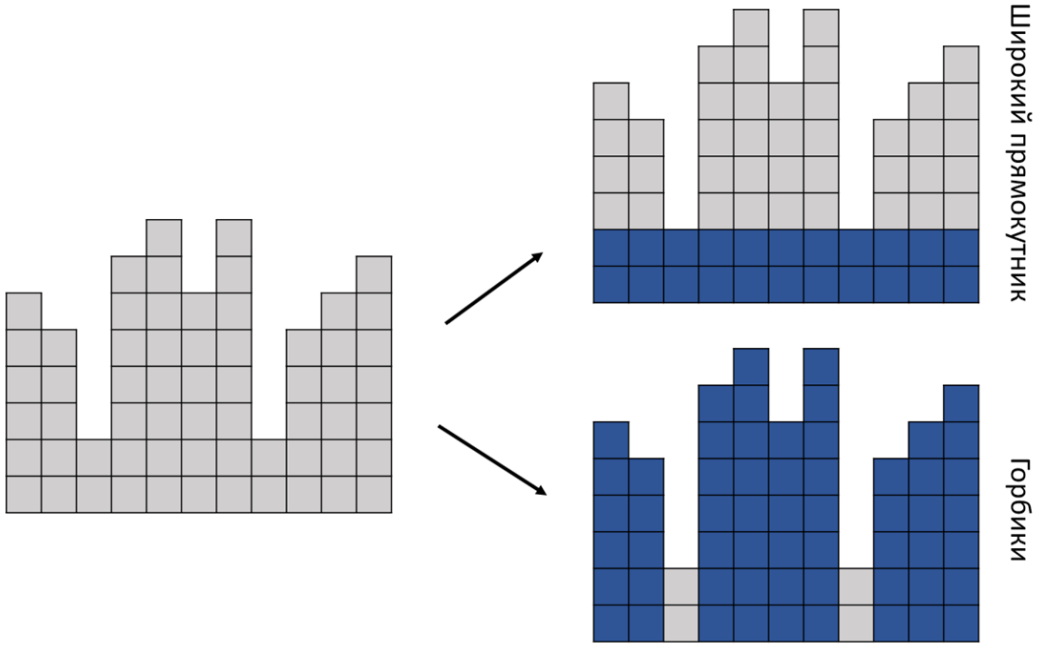
standard input	standard output
11 6 5 2 7 8 6 8 3 5 6 7	24

Ідея розв'язання задачі

Хрол Н. П., ***учителя інформатики загальноосвітньої спеціалізованої школи І-ІІІ ступенів фізико-математичного профілю № 12 м. Чернігова***

Потьомкіна Л. Є., ***випускника ПНЛ №145 м. Києва***

Розділяй і володарюй



- Відповідь до задачі на відрізку стовпців це максимум з:
  - площі **широкого прямокутника**
  - відповіді для кожного з **горбиків**
- Реалізуємо функцію int answer(int L, int R), котра буде:
  - обчислювати площу **широкого прямокутника**
  - викликати себе рекурсивно на відрізках **горбиків**

  Де $[L; R]$ це даний відрізок стовпців

Реалізація

Структури даних

unordered_map<int, vector<int>> level	**Дерево Відрізків на вхідному масиві**
level[height] буде містити відсортований масив індексів всіх стовпців с висотою height	Дає можливість швидко шукати мінімум на будь-якому відрізку стовпців
Побудова за $O(n)$	Побудова за $O(n \log n)$
Доступ за $O(1)$	Пошук за $O(\log n)$

Примітка: можна було використати ***RMQ*** *замість* ***Дерева Відрізків****, однак асимптотика б не змінилась, а реалізація б ускладнилась.*

Крок 1

- Площа **широкого прямокутника** = висота × ширина
- Ширина = $R - L + 1$
- Висота = мінімум на $[L; R]$
- Мінімум шукаємо за допомогою **Дерева Відрізків**

Асимптотика: $O(\log n)$

Крок 2

- Аби знайти **горбики** треба знайти індекси всіх мінімумів на $[L; R]$
- Аби знайти значення мінімума ми скористались **Деревом Відрізків** у Кроці 1
- Аби знайти <u>всі</u> індекси використовуємо levels[height]
- Тепер обмежимо індекси лише до тих, які лежать <u>всередині нашого відрізка</u>
  - Для швидкого обмеження використовуємо бінарний пошук:
  - int from = lower_bound(all(mins), L) – mins.begin();
    int to = --upper_bound(all(mins), R) – mins.begin();

Асимптотика: $O(\log n)$

Крок 3

- Ітеруємо по mins від from до to та викликаємоanswer на цих відрізках
- Помітимо, що answer буде викликана не більше n разів
  - *Підказка: кожен виклик відповідає єдиному **широкому прямокутнику**, так само як і кожен стовпець*

Підсумкова асимптотика: $O(n \log n)$

Лістінг програми Потьомкіна Л.Є., випускника ПНЛ №145 м. Києва):

```
#pragma GCC optimize("O3")
#include <bits/stdc++.h>
#define all(x) (x).begin(), (x).end()
#define int lint

using namespace std;
typedef long long lint;
unordered_map<int, vector<int>> level;
int *a, n;

struct node {
   node *left, *right;
   int value;
   node(int v): value(v), left(0), right(0) {}
   node(node *l, node *r):  left(l),  right(r) {
```

```
        value = min(l->value, right->value);
    }
} *root;

node *build(int l, int r) {
    if (l == r) return new node(a[l]);
    int m = (l+r)/2;
    return new node(build(l, m), build(m+1, r));
}

int get_min(node *cur, int ql, int qr, int l, int r) {
    if (ql == l && qr == r) return cur->value;
    int m = (l+r)/2;
    if (qr <= m) return get_min(cur->left, ql, qr, l, m);
    if (m < ql) return get_min(cur->right, ql, qr, m+1, r);
    return min(get_min(cur->left, ql, m, l, m),
               get_min(cur->right, m+1, qr, m+1, r));
}

int answer(int l, int r) {

    int height = get_min(root, l, r, 0, n-1);
    int ans = height * (r-l+1);

    const auto& mins = level[height];
    int from = lower_bound(all(mins), l) - mins.begin();
    int to = --upper_bound(all(mins), r) - mins.begin();
    int prv = l-1;

    for (int i = from; i <= to; i++) {
        int curr = mins[i];
        if (prv+1 <= curr-1)
            ans =  max(ans, answer(prv+1, curr-1));
        prv = curr;
    }

    if (prv+1 <= r)
        ans = max(ans, answer(prv+1, r));

    return ans;
}


signed main() {
```

```
        cin.tie(0); cout.tie(0); ios_base::sync_with_stdio(0);
    cin >> n;
    a = new int[n];

    for (int i = 0; i < n; i++) {
        cin >> a[i];
        level[a[i]].push_back(i);
    }

    root = build(0, n-1);
    cout << answer(0, n-1);
    return 0;
}
```

Ідея розв'язання задачі

Солонка Н.В., ***учня 11 класу ліцею № 15 м. Чернігова***

```
#include <iostream>

using namespace std;

int main(int argc, char** argv) {
        int n, i, j, tempcounter;
        long long tempsquare, square = 0;
        cin >> n;
        int *arr = new int[n];
        for (i = 0; i < n; i++)
                cin >> arr[i];
        for (i = 0; i < n; i++) {
                tempcounter = arr[i];
                while (tempcounter > arr[i - 1] || i == 0) {
                        j = 0;
                        tempsquare = 0;
                        while (arr[i + j] >= tempcounter&& j + i < n) {
                                tempsquare += tempcounter;
                                j++;
                        }
                        if (tempsquare > square)
                                square = tempsquare;
                        if (i + j == n)
                                break;
                        else tempcounter = arr[i + j];
                }
        }
```

```
        delete[]arr;
        cout << square;
        system("pause");
        return 0;
}
```

Ідея розв'язання задачі

Зуба В. В., ***учителя математики, директора Прилуцької загальноосвітньої школи І-ІІІ ступенів № 7 Прилуцької міської ради Чернігівської області;***

Бондаренка С. М., ***учителя математики та інформатики Прилуцької загальноосвітньої школи І-ІІІ ступенів № 7 Прилуцької міської ради Чернігівської області***

```
var a:array[0..1000001] of longint;
    i,n,x,y:longint;    s:int64;
Begin
 read(n);
 For i:=1 to n do read(a[i]);
 For i:=1 to n do
 Begin
  If a[i]=a[i-1] Then Continue; {Якщо ідуть підряд однакові числа}
  x:=i;y:=i; {Координати}
  While (a[x]>=a[i]) and (x>=1) do dec(x); {Пошук вліво чисел більших чи
рівних a[i]}
  While (a[y]>=a[i]) and (y<=n) do inc(y); ); {Пошук вправо чисел більших
чи рівних a[i]}
  If (y-x-1)*a[i]>s Then s:=(y-x-1)*a[i]; {Пошук площі прямокутника та її
порівняння з попередньо знайденою}
 End;
 Writeln(s);
End.
```

Ідея розв'язання задачі

Коваленко О. І., ***учителя інформатики та математики спеціалізованої загальноосвітньої школи №2 І-ІІІ ступенів з поглибленим вивченням іноземних мов м. Чернігова***

* Наша задача полягає в знаходженні прямокутника найбільшої площі.

* Для цього використаємо такий підхід:

* Якщо поточний блок висоти H, тоді будемо намагатися побудувати найдовший прямокутник цієї висоти.

* Для цього нам будуть підходити всі блоки, які не нижчі за нього.

* Коли зустрінемо блок висоти h (h < H), тоді перерахуємо відповідь (оберемо максимум між площею поточного прямокутника та поточної відповіддю).

* Після перерахунку відповіді можемо зменшити висоту прямокутника до h та продовжити побудову прямокутника.

* (так як всі попердні блоки були висоти вищої за h, то ми можемо продовжувати побудову прямокутника с тією самою довжиною, але нижчим).

* Цю операція будемо повторювати послідовно для кожного блоку, до тих пір поки поточна висота більша за висоту попередньго блоку,

* адже коли висота стає менша за попередню, то ми можемо сказати що прямокутник цієї висоти вже обраховувася на попередніх кроках.

* Після цього в результаті отримаємо оптимальну відповідь.

```
#include <iostream>
#include <vector>

using namespace std;

int main(int argc, char** argv) {

    int n;
    long long best_square = 0;

    cin >> n;
    vector <int> a(n);

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


    for (int i = 0; i < n; i++) {
        int height = a[i];
        int length = 0;
        /*Якщо поточна висота більша за висоту попердньго блоку
         * (інакше ми вже проглядали прямокутники цієї висоти)
         *
         */
        while (i == 0 || height > a[i - 1]) {
```

```
                    /* Знаходимо підпослідовність прямокутників які НЕ нижчі за поточну висоту
                     * Зупинимося, коли зустрінемо перший нижчий, або кінець масиву
                     */
                    while (i + length < n && a[i + length] >= height) {
                        length ++;
                    }

                    /* Знаходимо площу поточної послідовності та порівнюємо з максимумом */
                    long long cur_square = static_cast<long long>(length) * height;
                    best_square = max(cur_square, best_square);

                    /* Якщо ми зупинилися в кінці масиву => виходимо з циклу
                     * інакше зменшуємо висоту та продовжуємо побудову послідовності
                     */
                    if (i + length >= n)
                        break;
                    else height = a[i + length];
                }
            }
            cout << best_square;
            return 0;
        }
```

Задача E – Друзі Степана

Степан повернувся з міжнародної олімпіади школярів з програмування (IOI) і привіз з собою N різнокольорових каменів в якості сувенірів. Степан зовсім не жадний хлопчик, тому вирішив поділитися камінням зі своїми друзями. Кожному другу Степан віддав рівно один камінь. Виявилося, що у самого Степана залишився теж тільки один камінь. Визначте, скільки ж у нього друзів?

Вхідні дані:

У першому рядку вхідного файлу задано число N ($1 \le N \le 100$).

Вихідні дані:

Виведіть у вихідний файл одне число - кількість друзів Степана.

Пояснення до прикладу:

Степан привіз 2 каменя, один з яких залишився у нього. Отже, другий камінь Степан віддав своєму єдиному другу.

Приклади

Вхідні дані	Результат роботи
2	1

Ідея розв'язання задачі

Зуба В. В., ***учителя математики, директора Прилуцької загальноосвітньої школи І-ІІІ ступенів № 7 Прилуцької міської ради Чернігівської області;***

Бондаренка С. М., ***учителя математики та інформатики Прилуцької загальноосвітньої школи І-ІІІ ступенів № 7 Прилуцької міської ради Чернігівської області***

Оскільки кожному другу Степан дає по одному каменю і в нього залишається 1 камінь, то кількість друзів на 1 менше за кількість каменів.

```
  var n:integer;
begin
  read(n);
  write(n-1);
end.
```

Задача F – Квіти

Ромчик вирішив пересадити квіти. У нього є a – літрових, b – 1,5-літрових, c – 2-літрових, d – 2,5 літрових вазонів з квітами. Кожну квітку потрібно пересаджувати у вазон на 0,5 л більший. Допоможіть Ромчику обчислити скільки вазонів потрібно купити, щоб пересадити квіти? Вазони можна використовувати повторно.

Вхідні дані:

Цілі числа a, b, c, d – кількість вазонів з квітами ($0 \le n \le 10000$).

Вихідні дані:

Ціле число k – кількість вазонів, які необхідно придбати.

Приклад:

Вхідні дані a, b, c, d	Результат роботи z
1 0 0 3	4

```
  var a,b,c,d,k:longint;
begin
  read(a,b,c,d);
  k:=d;
  if c>d then k:=k+(c-d);
  if b>c then k:=k+(b-c);
  if a>b then k:=k+(a-b);
  write(k);
end.
```

Задача G – Факультатив з інформатики

Степан вирішив додатково позайматись програмуванням і записався на факультатив з інформатики. Він записав у щоденник час початку заняття (години, хвилини і секунди) та час закінчення (години, хвилини і секунди). Тепер його цікавить скільки годин, хвилин і секунд він буде задіяний на факультативі?

Вхідні дані:

У першому рядку записані три цілих числа A, B, C ($0 \le A \le 23$, $0 \le B \le 59$, $0 \le C \le 59$) - час початку заняття в годинах, хвилинах і секундах. У другому рядку записані три цілих числа G, H, S ($0 \le G \le 23$, $0 \le H \le 59$, $0 \le S \le 59$) - час закінчення заняття в годинах, хвилинах і секундах.

Вихідні дані:

Три цілих числа - тривалість заняття в годинах, хвилинах і секундах.

Обмеження: Число N від 0 до 100 включно. Число N від 0 до 10000 включно. Результат роботи методу не перевершує 2000000000.

Приклади:

Вхідні дані	Результат роботи
12 30 10 13 50 15	1 20 5

Ідея розв'язання задачі

Зуба В. В., ***учителя математики, директора Прилуцької загальноосвітньої школи І-ІІІ ступенів № 7 Прилуцької міської ради Чернігівської області;***

Бондаренка С. М., ***учителя математики та інформатики Прилуцької загальноосвітньої школи І-ІІІ ступенів № 7 Прилуцької міської ради Чернігівської області***

Зрозуміло, що відповідь отримується шляхом віднімання від часу закінчення факультативу часу його початку. Переведемо все в секунди. З умови не зрозуміло, чи відбувається факультатив в межах однієї доби, тому варто передбачити, що факультатив закінчується наступної доби.

```
  var h1,m1,s1,h2,m2,s2:int64;
begin
  read(h1,m1,s1,h2,m2,s2);
  h1:=(h2*3600+m2*60+s2)-(h1*3600+m1*60+s1);
  if h1<=0 then h1:=h1+24*3600;
  h2:=h1 div 3600;
  m2:=h1 mod 3600 div 60;
  s2:=h1 mod 60;
  write(h2,' ',m2,' ',s2);
end.
```

Задача Н – Цукерки

На столі стоять три вази з цукерками. В лівій вазі лежить А цукерок, в середній вазі - В цукерок, в правій вазі відповідно С цукерок. Степан з'їдає одну цукерку з лівої вази, потім - одну цукерку з середньої вази, потім з правої, середньої, лівої, середньої, правої, середньої і т.д. (зліва направо, потім наліво, знову направо і т.д.). Якщо Степан хоче взяти цукерку з якоїсь вази, а там її немає, то він засмучується і йде спати. Визначте скільки цукерок з'їсть Степан.

Вхідні дані:

Дано 3 цілих невід'ємних числа A, B, C (A+B+C ≤ 2*109) кількість цукерок відповідно в лівій, середній та правій вазі.

Вихідні дані:

Одне число - кількість цукерок, які з'їв Степан.

Приклади:

Вхідні дані	Результат роботи
3 3 3	7

Ідея розв’язання задачі

Зуба В. В., ***учителя математики, директора Прилуцької загальноосвітньої школи І-ІІІ ступенів № 7 Прилуцької міської ради Чернігівської області;***

Бондаренка С. М., ***учителя математики та інформатики Прилуцької загальноосвітньої школи І-ІІІ ступенів № 7 Прилуцької міської ради Чернігівської області***

Спробуємо розв'язати задачу "напряму" і організуємо циклічне ходіння Степана від однієї вази (a[1]) до іншої (a[3]) з поїданням (-1) "цукерок" в них.

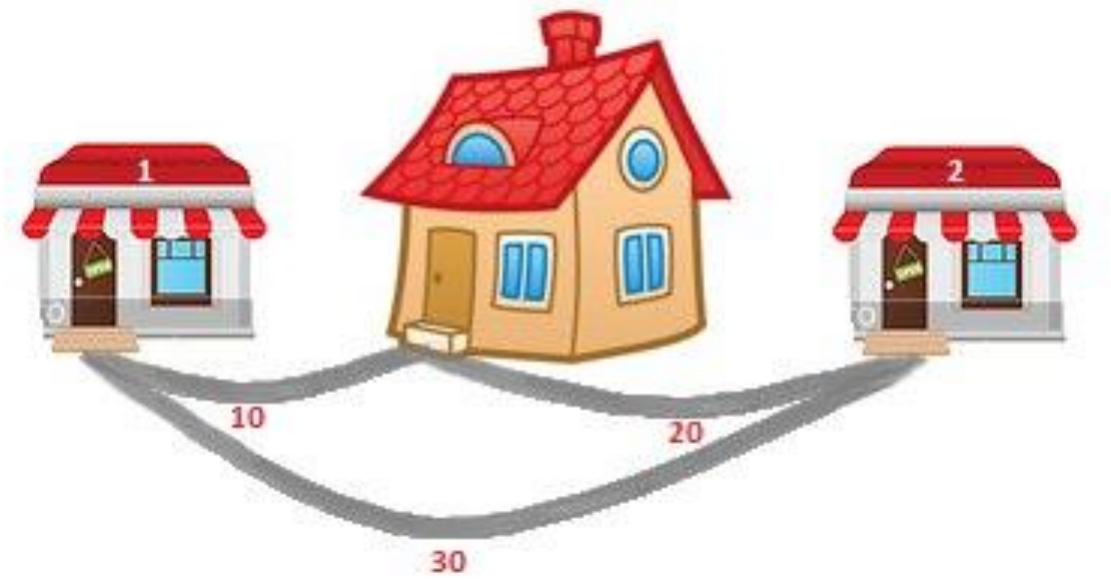
```
  var a:array[1..3]of longint;i,k,f:longint;
begin
  read(a[1],a[2],a[3]);
  k:=0;i:=1;f:=-1;
  while 1=1 do
    begin
      if a[i]>0 then begin inc(k);dec(a[i]);end else break;
      if i<>2 then f:=-f;
      i:=i+f;
    end;
  write(k);
end.
```

Такий код хоч і не раціональний, але дає досить швидко заробити 72 бали із 100. Щоб отримати максимальний бал потрібно застосувати математику. Аналіз прикладу та умови показує, що цукерки в середній вазі поїдаються в 2 рази швидше. Також аналіз показує, що за один прохід туди-назад Степан поїдає 4 цукерки. Кількість таких проходів можна знайти, визначивши найменше серед чисел A, B/2, C. Після визначеної кількості проходів принаймні в одній з ваз залишиться не більше 1 цукерки.

```
  var a,b,c,k,min:longint;
begin
  read(a,b,c);
  if a=0 then begin write(0);exit;end;
  min:=a;
  if b div 2 <min then min:=b div 2;
```

```
  if c<min then min:=c;
  k:=min*4;
  a:=a-min;
  b:=b-2*min;
  c:=c-min;
  if a=0 then begin write(k);exit;end;
  if b=0 then begin write(k+1);exit;end;
  if c=0 then begin write(k+2);exit;end;
  write(k+3);
end.
```

Задача I – Степан і похід в магазин



Сьогодні Степан чекає в гості свого друга Василя. Щоб підготуватися до зустрічі, Степану необхідно відвідати два магазини, розташованих поряд з його будинком. Від будинку до першого магазину веде доріжка довжини d1 метрів, а до другого магазину веде доріжка довжини d2 метрів. Також існує доріжка, безпосередньо сполучає два магазини один з одним, довжиною d3 метрів. Допоможіть Степану обчислити мінімальну відстань, яку йому буде потрібно пройти, щоб відвідати обидва магазини і повернутися додому. Степан завжди стартує зі свого будинку. Він повинен відвідати обидва магазини, переміщаючись тільки за наявними трьома доріжками, і повернутися назад додому. При цьому його абсолютно не збентежить, якщо йому доведеться відвідати один і той же магазин або пройти по одній і тій же доріжці більше одного разу. Єдине його завдання - мінімізувати сумарну пройдену відстань.

Вхідні дані:

У першому рядку вхідних даних знаходяться три цілих числа d1, d2, d3 ($1 \le d1, d2, d3 \le 108$) - довжини доріжок.

- d1 - довжина доріжки, з'єднує будинок Степана і перший магазин;

- d2 - довжина доріжки, з'єднує будинок Степана і другий магазин;
- d3 - довжина доріжки, що з'єднує два магазина.

Вихідні дані:

Виведіть мінімальну кількість метрів, яку доведеться пройти Степану, щоб відвідати обидва магазини і повернутися додому.

Приклади:

Вхідні дані	Результат роботи
10 20 30	60

Ідея розв'язання задачі

Зуба В. В., учителя математики, директора Прилуцької загальноосвітньої школи І-ІІІ ступенів № 7 Прилуцької міської ради Чернігівської області;

Бондаренка С. М., учителя математики та інформатики Прилуцької загальноосвітньої школи І-ІІІ ступенів № 7 Прилуцької міської ради Чернігівської області

Існує всього три маршрути, якими може рухатися Степан:

1. дім – магазин1 – магазин 2 – дім;
2. дім – магазин2 – магазин1 – дім;
3. дім – магазин1 – дім – магазин2 – дім.

Необхідно визначити довжину кожного маршруту, та вибрати найкоротший.

```
  var d1,d2,d3,l1,l2,l3,min:longint;
begin
  read(d1,d2,d3);
  l1:=d1+d2+d3;
  l2:=(d1+d2)*2;
  l3:=(d1+d3)*2;
  min:=(d2+d3)*2;
  if l1<min then min:=l1;
  if l2<min then min:=l2;
  if l3<min then min:=l3;
  write(min);
end.
```