Управління освіти і науки
Чернігівської обласної державної адміністрації

Ніжинський державний університет
імені Миколи Гоголя

ЗБІРНИК ЗАДАЧ ТА РОЗВ'ЯЗКІВ ІІ ЕТАПУ ВСЕУКРАЇНСЬКОЇ УЧНІВСЬКОЇ ОЛІМІПАДИ З ІНФОРМАТИКИ

2015-2016 НАВЧАЛЬНОГО РОКУ

Ніжин – 2016

УДК 371.32: 004(072)
ББК 22.183.4р10
Х22

Рекомендовано Вченою радою Ніжинського державного університету імені Миколи Гоголя (НДУ ім. М. Гоголя)
Протокол №3 від 29.09.2016 р.

Рецензенти:

Віра М.Б., доцент кафедри вищої математики НДУ ім. Миколи Гоголя, канд. фіз.- мат. наук

Лісова Т.В., доцент кафедри прикладної математики, інформатики і освітніх вимірювань НДУ ім. Миколи Гоголя, канд. фіз.- мат. наук

Рожовець Н.М., учитель математики та інформатики Ніжинської ЗОШ І-ІІІ ст. №15, учитель-методист

Збірник задач та розв'язків ІІ етапу Всеукраїнських учнівських олімпіад з інформатики 2015-2016 навчального року /укл. В. М. Харченко. – Ніжин: НДУ імені Миколи Гоголя, 2016. – 54 с.

Збірник допоможе вчителям та учням, які готуються до участі у Всеукраїнській учнівській олімпіаді з інформатики (програмування), або ж факультативно займаються програмуванням. Він буде корисним і для студентів фізико-математичних факультетів, що вивчають курси «Програмування», «Програмування складних алгоритмів» або беруть участь у АСМ-олімпіадах.

ЗМІСТ

ВСТУП

Даний посібник містить завдання ІІ етапу Всеукраїнської учнівської олімпіади з інформатики за 2015-2016 н.р. та ідеї розв'язання задач. Він є логічним продовженням методичних рекомендацій, які були опубліковані в Чернігівському обласному інституті післядипломної педагогічної освіти імені К.Д. Ушинського за минулі роки.

Збірник допоможе вчителям та учням, які готуються до участі у Всеукраїнській учнівській олімпіаді з інформатики (програмування), або ж факультативно займаються програмуванням. Він буде корисним і для студентів фізико-математичних факультетів, що вивчають курс «Програмування складних алгоритмів» або беруть участь у АСМ-олімпіадах.

8-11 клас
On-line варіант

Задача А-Цукерки

Обмеження часу: 100 мс

Обмеження пам'яті: 256 M

Петрик та Василько дуже люблять цукерки. Та вони їх не їдять, а колекціонують. Петрик полюбляє цукерки з малюнком ведмедя на обгортці, а Василько полюбляє з малюнком лисиці. У Петрика є непотрібні йому цукерки (обгортки цукерок не подобаються Петрикові). І у Василька є непотрібні йому цукерки. Також Петрик і Василько полюбляють обмінюватись непотрібними цукерками. Вони обмінюють всі свої непотрібні, на всі непотрібні цукерки товариша.

Вхідні дані:

дано 2 числа N, K $(1 \le N, K \le 10^9)$ – кількість непотрібних цукерок відповідно Петрика і Василька.

Вихідні дані:

одне число – кількість цукерок, які отримав Петрик під час обміну.

Приклади

Вхідні дані	Результат роботи
1 2	2
2 3	3

Розв'язання

Задача відноситься до тих, які читати довше, ніж писати код програми. Слід зчитати кількість непотрібних цукерок Петрика і Василька, а вивести – кількість непотрібних цукерок Василька. Зауважимо, що числа можуть бути більшими за 32767, а тому вибирати доцільно тип LongInt.

Код програми на Free Pascal:

```
var n,k:longint;
begin
readln(n,k);
writeln(k);
end.
```

Задача B-Звичайна арифметика

Обмеження часу: 100 мс

Обмеження пам'яті: 256 M

Дано два числа (a і b) та операція (словом), котру необхідно виконати:

plus – додати ці числа

minus – відняти друге число від першого

div - знайти цілу частину від ділення першого числа на друге

mod – знайти остачу від ділення першого числа на друге

mult – перемножити дані числа

Напишіть програму, яка за даними числами та операцією, обчислює результат.

Вхідні дані:

дано 2 числа $a, b\ (-2*10^9 \le a, b \le 2*10^9)$ та операцію, яка записана через один пробіл.

Вихідні дані:

одне число – результат виконаної операції.

Приклади

Вхідні дані	Результат роботи
3 4 plus	7
12 5 div	2
9 4 mod	1
4 5 minus	-1
7 3 mult	21

Розв'язання

Дана задача на просте використання рядкових величин. Особливістю її є те, що при перевірці слова-операції слід враховувати прогалину, яка відокремлює число від тексту. За умовою задачі числа a і b належать типу LongInt, проте для перестраховки візьмемо тип `Int64`, який дозволяє зберігати числа до $2^{63} - 1$.

Код програми на Free Pascal:

```
var a,b:int64;
       t:string;
begin
 readln(a,b,t);
 if t=' plus' then writeln(a+b);
 if t=' minus' then writeln(a-b);
 if t=' mult' then writeln(a*b);
```

```
 if t=' div' then writeln(a div b);
 if t=' mod' then writeln(a mod b);
end.
```

Задача С-Шкідлива речовина

Обмеження часу: 100 мс

Обмеження пам’яті: 256 М

В одному з боксів хімічної лабораторії, де ведуться наукові дослідження, відбувся витік шкідливої речовини. Коли ця речовина сягає зовнішніх стін лабораторії, вмикається попереджувальна сирена.

Відомо, що лабораторія складається з ***n*** боксів по горизонталі та ***m*** боксів по вертикалі, які мають нумерацію ***(i, j)***, де ***i*** – номер рядка, а ***j*** – номер боксу в рядку ***(1 ≤ i ≤ n, 1 ≤ j ≤ m)***. Витік шкідливої речовини відбувся у боксі з номером ***(x, y)*** і протягом 1 хвилини поширюється у сусідні бокси через спільні з ним стіни.

Вхідні дані:

у першому рядку містяться розміри лабораторії ***n, m (1 ≤ n, m ≤ 10^9)***, у другому – нумерація боксу ***x, y (1 ≤ x ≤ n, 1 ≤ y ≤ m)***, де відбувся витік шкідливої речовини.

Вихідні дані:

одне число, яке визначає мінімальну кількість хвилин, через яку увімкнеться попереджувальна сирена.

Приклади

Вхідні дані	Результат роботи
5 5 2 3	1
5 5 3 3	2

Розв'язання

Спочатку складається враження, що для розв'язання даної задачі потрібно застосувати двовимірні масиви. Проте це не так. Дану задачу можна розв'язати за допомогою операторів розгалуження і стандартних змінних.

На рис. 1 схематично зобразимо лабораторію й пронумеруємо бокси. Витік шкідливої речовини у боксі з координатами (x,y) позначимо зафарбованим прямокутником, а напрями поширення цієї речовини – стрілками.

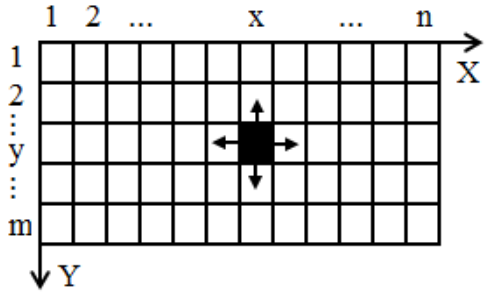


Рис. 1

Очевидно, що мінімальний шлях розповсюдження речовини до меж лабораторії відбуватиметься по вертикалі, або по горизонталі від вказаного боксу. Тому слід знайти відстань від зафарбованого боксу до лівої (змінна `xl`), правої (змінна `xp`), верхньої (змінна `yv`) та нижньої межі (змінна `yvn`). Залишається серед даних величин знайти найменшу – це і буде часом розповсюдження шкідливої речовини.

Враховуючи обмеження, які накладені на вхідні дані, слід використовувати тип LongInt. Якщо ж використовувати тип Integer або Word – програма не пройде всі тести.

Код програми на Free Pascal:

```
var n,m,x,y,xl,xp,yv,yvn,minx,
    miny:longInt;
begin
 readln(n,m);
 readln(x,y);
 xl:=x-1;
 xp:=n-x;
 yv:=y-1;
 yvn:=m-y;
 if xl<xp
 then minx:=xl
 else minx:=xp;
 if yv<yvn
 then miny:=yv
 else miny:=yvn;
 if minx<miny
 then writeln(minx)
 else writeln(miny);
end.
```

Задача D-Гра

Обмеження часу: 500 мс

Обмеження пам’яті: 256 M

У парку встановили новий атракціон, який був зроблений у вигляді закритого круглого приміщення, куди заходив гравець. У приміщенні вимикалося світло і по колу на стінах з’являлися білі і чорні квадрати. Гравець за один крок гри може прибирати два квадрати однакового кольору, притискаючи до них свої долоні. Послідовність прибирання квадратів різних кольорів не має значення, тобто спочатку можна прибрати всі квадрати одного кольору, а потім іншого, або у будь-якій іншій послідовності.

Гравець виграє приз, якщо у результаті залишиться один квадрат білого кольору, і залишиться без призу, якщо залишиться лише один квадрат чорного кольору. У разі, коли не залишиться жодного квадрату, то у нього з’являється можливість зіграти ще раз.

Допоможіть гравцю якнайшвидше визначитися з перспективами 5 сеансів гри.

Вхідні дані:

у першому рядку міститься число ***w***, яке визначає початкову кількість квадратів білого кольору в першій грі, у другому рядку – число ***b***, яке визначає початкову кількість квадратів чорного кольору в першій грі ***(1 ≤ w, b ≤ $10^{1000000}$)***. У третьому та четвертому рядках – числа ***w*** та ***b***, що визначають початкову кількість квадратів відповідно білого та чорного кольорів в другій грі. Аналогічно для решти (всього 10 рядків – по 2 на кожну з 5 ігор).

Вихідні дані:

для кожної з 5 ігор в окремому рядку вивести ‘Lose’, якщо гравець не отримає приз, ‘Win’, якщо гравець отримає приз, та ‘Again’, якщо він зможе зіграти ще раз.

Приклади

Вхідні дані	Результат роботи
1 2	Win Lose

2 1 10 100 1001 1001 1411 2015	Again Win Win

Розв'язання

Оскільки у вхідних даних вказано, що числа $1 \leq w, b \leq 10^{1000000}$, то спочатку спадає на думку використати довгу арифметику. Уважне читання умови й обмежень ресурсів дає підстави засумніватися в початковій ідеї. В умові говориться, що можна прибрати парну кількість білих та чорних квадратів. Черговість такого усунення квадратів не важлива. З [1, с. 10] відомо, що парними називаються числа які діляться на 2. Число буде ділитися націло на 2, якщо його остання цифра ділиться націло на 2. Згідно умови задачі та наведених прикладів очевидно, що якщо число w є непарним, то гравець

виграє, причому станеться це незалежно від парності числа b. Автомат виграє лише тоді, коли число b – непарне, а число w є парним. Коли ж числа w і b є парними, то гравцю дається можливість зіграти ще раз. Перевіряти на парність будемо не числа, а їх останні цифри.

Враховуючи гранично можливу кількість цифр заданих чисел, у задачі для їх зчитування доречно використати тип AnsiString. Оскільки про нього не йде мова у підручниках з інформатики, то зупинимося на ньому більш детально. Тип AnsiString використовує кодування ANSI. Рядок символів розміщується в динамічній пам’яті. Для одного символу відводиться 1 байт. У статичній пам’яті компілятор створює покажчик розміром в 4 байта, в який при створенні рядка заноситься адреса його початку. Покажчик до створення рядка і після його видалення містить nil – ознаку відсутності даних. У рядку AnsiString може бути не більше $2^{31} = 2147483648$ символів.

Код програми на Free Pascal:

```
var w,b:ansistring;
    i:longint;
begin
 for i:=1 to 5 do
  begin
   readln(w);
   readln(b);
   if ((ord(w[length(w)])-ord('0'))
mod 2=1) then writeln('Win')
   else
    if ((ord(b[length(b)])-ord('0'))
mod 2=1) and ((ord(w[length(w)])-
ord('0')) mod 2=0)
    then writeln('Lose')
    else writeln('Again');
  end;
end.
```

Задача Е-Черга

Обмеження часу: 500 мс

Обмеження пам’яті: 8 М

Перед відкриттям кас, де продаються квитки на залізничні потяги, вишикувалася

величезна черга. Після відкриття кас виявилося, що працюють не всі вікна і час обслуговування обмежений. Самі покупці вирішили розподілитися по працюючих касах таким чином: не порушуючи послідовності людей у попередньо зайнятій черзі, до першого вікна переходить перша група покупців, до другого – наступна і т.д. Розподіл черги на групи повинен враховувати час роботи кас.

Вхідні дані: у першому рядку міститься кількість людей у черзі N $(1 \le N \le 3*10^6)$, кількість працюючих вікон K $(1 \le K \le 10^9)$, час роботи кас t. У другому – час обслуговування $a_1, a_2, \ldots, a_n$ $(1 \le a_i \le t \le 10^9)$ кожної людини у черзі.

Вихідні дані: одне число, яке визначає максимальний час, необхідний для обслуговування покупців у межах часу роботи кас t.

Приклади

Вхідні дані	Результат роботи
5 3 4 2 3 2 1 4	3

Вхідні дані	Результат роботи
5 3 2 1 1 1 1 1	2
5 3 5 1 1 1 1 1	5

Розв'язання

Подивившись на формат вхідних даних, складається враження, що найпростіше використати масиви. Проте обмеження пам'яті не дозволяє цього зробити для граничних значень задачі.

У програмі будемо використовувати змінні n, k і t для зберігання кількості людей, кількості кас та час роботи кас відповідно. Змінну *a* використовуватимемо для зберігання часу обслуговування наступної людини, j – для зберігання кількості груп людей до кас, а *i* – як лічильник циклу уведення даних. Саме тому, врахувавши вхідні обмеження, оголосимо їх тип даних LongInt. У змінній b будемо зберігати час обслуговування поточної людини групи. Оскільки додавання кількох чисел порядку 10^9 приводить до переповнення

змінної типу LongInt, то використаємо 64 розрядний тип Int64. Змінна max буде зберігати максимальний час обслуговування груп, а тому в неї тип повинен бути також Int64.

Оскільки в групах не повинна порушуватися загальна черга, то їх формування здійснюється таким чином: формується група тих, чий час не більший за час роботи кас і відправляється до першої каси, потім формується друга група і так далі. Це стає очевидним із аналізу третього набору вхідних даних та результату. Бо якби спочатку підходили до всіх вільних кас, а потім за ними ставали наступні, то час обслуговування черги у третьому наборі був би 2, а не 5. У сформованих групах вибирається максимальний час – він і є відповіддю задачі.

Особливістю розв'язання є те, що час обслуговування людини може бути більшим за час роботи кас. У даному випадку час обслуговування черги слід вивести як час роботи кас.

Код програми на Free Pascal:

```
var n,k,t,i,j,a:longint;
```

```
    b,max:int64;
begin
 readln(n,k,t);
 max:=-1;
// Формуваннння першої групи
 j:=1;i:=1;
 b:=0;
 pr:=false;
 read(a);
{ Якщо час обслуговування більший за
час роботи кас}
 if a>=t then writeln(t)
 else
 begin
{ Додаємо час до часу групи, що
формується}
 b:=b+a;
 if b>max then max:=b;
 for i:=2 to n do
  begin
     read(a);
{ Перевірка чи не перевищує час
групи час роботи кас}
     if b+a<=t then
      begin
       b:=b+a;
```

```
        if b>max then max:=b;
       end
      else
       begin
        b:=0;
//Формування наступної групи
        inc(j);
        j:=j mod (k+1);
{ Якщо кількість груп більша за
кількість кас, що працюють }
        if j=0 then break
        else
         if b+a<=t then
          begin
           b:=b+a;
           if b>max then max:=b;
          end;
       end;
       if max=t then break;
  end;
//Виведення результату
 writeln(max);
 end;
end.
```

8-11 клас
Паперовий варіант

Зауважимо, що запасний паперовий варіант був сформований на основі задач із ресурсу www.e-olymp.com [6]. Тому їх можна перевірити, відправивши розв'язок за вказаною адресою.

Задача А – Пиріжки

Пиріжок у шкільній їдальні коштує a гривень та b копійок. Знайдіть, скільки гривень та копійок заплатить Петрик за n пиріжків.

Вхідні дані: три натуральних числа a, b, n ($0 \leq a,\ b,\ n \leq 100$).

Вихідні дані: через пропуск два числа – вартість покупки у гривнях та копійках.

Приклад:

Вхідні дані	Вихідні дані
1 25 2	2 50

Розв'язання

Переведемо вартість покупки с в копійки, а потім знайдемо кількість витрачених гривень

за n пиріжків (`c div 100`) та копійок (`c mod 100`). Перевірити розв'язання можна, викликавши задачу №7336 ресурсу [6].

Код програми на Free Pascal:

```
var a,b,n,c,:longint;
begin
 readln(a,b,n);
 c:=(a*100+b)*n;
 writeln(c div 100,' ',c mod 100);
end.
```

Задача B – Постійна сума цифр

Знайти кількість двозначних чисел, які не змінюють свою суму цифр при множені числа на однозначне ціле число N ($N = 0..9$).

Вхідні дані: ціле число N ($0 \leq N \leq 9$).

Вихідні дані: відповідь до задачі.

Приклад:

Вхідні дані	**Вихідні дані**
2	10

Пояснення: При множенні двоцифрових чисел на 2 не змінюють суму цифр такі числа: 18 27 36 45 54 63 72 81 90 99.

Розв'язання

Дана задача має №7338 на ресурсі [6]. Оскільки слід перевірити тільки числа від 10 до 99, то використаємо повний перебір. Будемо знаходити суму двоцифрового числа та порівнювати її із сумою цифр даного числа, але помноженого на деяке число n. Якщо ці суми рівні, то збільшуватимемо лічильник таких чисел на 1.

Щоб знайти суму цифр, потрібно "брати" цифри по одній і додавати їх одну до одної, а потім використану цифру "відкидати". На мові Free Pascal для цього використовують операції ділення націло (div) на 10 та знаходження залишку від цілочисельного ділення на 10 (mod). Перша операція використовується для вилучення останньої цифри з числа, а друга – для виділення останньої цифри числа.

Здається, що можна знаходити суму цифр двоцифрового числа через оператор

```
Sum:=n div 10+ n mod 10;
```

Проте вже 12*9=108, а тому даний оператор не застосовний для чисел, що є добутком двоцифрового числа на цифру. Отже,

для знаходження суми цифр будемо використовувати стандартний алгоритм:

```
sum:=0;
while n<>0do
 begin
//Додавання до суми останньої цифри
  sum:=sum+ n mod 10;
//Вилучення з числа останньої цифри
  n:=n div 10; end;
```

Код програми на Free Pascal:

```
var i,l,n:longint;
function sum(n:longint):longint;
var sum1:longint;
begin
 sum1:=0;
 while n<>0do
   begin
    sum1:=sum1+ n mod 10;
    n:=n div 10;
   end;
 sum:=sum1;
end;
begin
 readln(n);
 l:=0;
 for i:=10 to 99 do
```

```
  if sum(i)=sum(i*n)
   then l:=l+1;
writeln(l);
end.
```

Задача C – Молоко та пиріжок

Учням першого класу призначають додаткову склянку молока та пиріжок, якщо першокласник важить менше **30** кг. У перших класах школи навчається n учнів. Склянка молока має об'єм **200** мл, а замовлені упаковки молока – **0,9** л. Визначити кількість додаткових пакетів молока та пиріжків, необхідних щодня.

Вхідні дані: у першому рядку задано ціле число **n** ($\mathbf{0 < n \leq 100}$). У наступному рядку знаходяться **n** додатних чисел – маси кожного першокласника.

Вихідні дані: в одному рядку вивести два цілих числа – кількість додаткових пакетів молока та пиріжків, необхідних щодня.

Приклад:

Вхідні дані	**Вихідні дані**
20	2 9

30 37 31 25 32 29 35 40 28 25 30 34 26 23 20 22 21 30 38 33	

Розв'язання

Задача знаходиться під №7365 на ресурсі [6]. За умовою потрібно організовувати циклічне введення N даних. Будемо зберігати кількість учнів, які легші за 30 кг у змінній p. У блоці введення зразу ж порівнюємо введену вагу із 30 кг: якщо вона менша, то до лічильника додаємо 1. Після введення всіх даних використаємо змінну n для зберігання необхідної для учнів кількості молока. Щоб визначити кількість пакетів молока, слід значення n цілочисельно поділити на 900 (вміст пакету) і проконтролювати чи залишок від ділення не більший за 0. Якщо залишок відрізняється від 0, то збільшимо кількість пакетів на 1.

Код програми на Free Pascal:

```
var n,i,m,p:longint;
    a:real;
begin
readln(n);
```

```
p:=0;
for i:=1 to n do
  begin
   read(a);
   if a<30 then inc(p);
  end;
n:=p*200;
m:=n div 900;
if (n mod 900)>0 then inc(m);
writeln(m,' ',p);
end.
```

Задача D – Цукерки

Іванко дуже любить цукерки. Цукерок у нього є багато і зберігає він їх у спеціальних скриньках. Всього у Іванка є N цукерок (N - парне) і S однакових скриньок. В одну скриньку поміщається не більше як N/2 цукерок. Іванкові стало цікаво, скількома ж способами він може розкласти цукерки по скриньках. Допоможіть йому знайти відповідь на його питання.

Зверніть увагу, що всі цукерки однакові, тому має значення тільки кількість цукерок в кожній зі скриньок. Тобто, два розклади цукерок по скриньках вважаються різними, якщо хоч в одній зі скриньок кількість цукерок в першому розкладі відрізняється від кількості цукерок в другому розкладі (у тій самій скриньці).

Вхідні дані: в єдиному рядку задано два числа N та S ($2 \leq N \leq 1000$ – кількість цукерок, N – парне; $2 \leq S \leq 1000$ – кількість скриньок).

Вихідні дані: єдине число – кількість різних можливих розкладів цукерок по скриньках.

Приклад:

Вхідні дані	**Вихідні дані**
4 3	6

Розв'язання

Задача знаходиться під №65 на ресурсі [6]. Як вказано у класифікації, вона відноситься до комбінаторних задач.

Особливістю даної задачі є те, що комбінації з повтореннями в загальноосвітніх школах не вивчають. Проте можна використати підхід, який запропоновано при виведенні даної комбінаторної схеми в [2]. Перефразуємо дану задачу таким чином: є N однакових цукерок, як можна поділити їх на S частин так, щоб у кожній з них було не більше $\frac{N}{2}$ цукерок? Використаємо для поділу $S-1$ перегородку, які поділять цукерки на S частин. Знайдемо спочатку загальну кількість розподілів цукерок по таких частинах, в тому числі й тих, коли в одній частині знаходяться всі цукерки. Тоді задачу можна сформулювати так: Скільки є способів із $N+S-1$ об'єктів вибрати $S-1$ об'єкт. Це можна зробити C_{N+S-1}^{S-1} способами. Відкинемо із даної кількості ті способи, які не задовольняють вказані в умові задачі обмеження. Це буде тоді, коли в один із ящиків поклали не менше ніж $\frac{N}{2}+1$ цукерку, тобто, $\frac{N}{2}+1$ цукерку закріпимо за окремим ящиком, а решту $\frac{N}{2}-1$ цукерку розподілимо аналогічно

описаному вище по всіх ящиках. Причому, більше половини цукерок можна класти лише в один ящик із S. Тому загальна фомула для отримання відповіді задачі запишеться:

$$C_{N+S-1}^{S-1} - sC_{\frac{N}{2}+S-2}^{S-1}.$$

Якщо використовувати стандартні типи Free Pascal, навіть 20 розрядний `qword`, то програма проходить лише 43% запропонованих тестів. Таку кількість тестів проходить і оптимізований алгоритм обчислення комбінацій. Використання функцій max і min без оператора розгалуження зменшує час виконання програми на деяких тестах. Код такої програми на Free Pascal вказано нижче:

```
var n,s,m,k,l, i,j:integer;
    c,p,p1,f,f1:qword;
function max(a,b:integer):integer;
begin
max:=(abs(a-b)+a+b) div 2;
end;
function min(a,b:integer):integer;
begin
 min:=(-abs(a-b)+a+b) div 2;
end;
```

```
function comb(nn,kk:integer):qword;
var ii:integer;
begin
 comb:=1;
 if kk>(nn-kk) then kk:=nn-kk;
 for ii:=1 to kk do
 comb:=comb*(nn-ii+1) div ii;
end;
begin
 readln(n,s);
 k:=max(n,s-1);
 l:=min(n,s-1);
 k:=k+l;
 p:=comb(k,l);
 k:=max((n div 2)-1,s-1);
 l:=min((n div 2)-1,s-1);
 k:=k+l;
 f1:=comb(k,l);
 p:=p-s*f1;
 writeln(p);
end.
```

Очевидно, що вказана вище програма не може знайти навіть C_{62}^{30}, а граничні значення N і S дорівнюють 1000. Тому при розв'язуванні даної задачі доречно використовувати

алгоритми так званої "довгої арифметики". Опис ідей та багатьох алгоритмів подано в [5]. Під кожне із довгих чисел будемо відводити масив із 1000 елементів. Кожен елемент масиву може містити чотиризначне число. У загальному випадку програма буде оперувати 4000 розрядними числами. У нульовій комірці буде зберігатися кількість комірок, які відводяться на дане число. У першій комірці – нижні розряди числа, а у k-ій – старші розряди даного числа. Запропонований нижче варіант програми буде проходити 100% тестів з [6].

```
const osn=10000;
{У комірках зберігаємо 4-и значні
числа}
      max_d=1000;
{Максимальна кількість комірок
числа}
type ar=array[0..max_d] of integer;
var a,b,c,d:ar;
    i,j,n,k,s:word;
    ch:char;
    dv:int64;
procedure readlong(var x:ar);
```

```
{Процедура зчитування "довгого
числа"}
var ii:word;
begin
 fillchar(x,sizeof(x),0);
 repeat
  read(ch);
 until ch in ['0'..'9'];
 while ch in ['0'..'9'] do
 begin
  for i:=x[0] downto 1 do
   begin
{"Протягання" старшої цифри із x[i]
в x[i+1]}
x[i+1]:=x[i+1]+(LongInt(x[i])*10)
div osn;
x[i]:=(LongInt(x[i])*10) mod osn;
   end;
  x[1]:=x[1]+ord(ch)-ord('0');
//Зміна довжини числа
  if x[x[0]+1]>0 then inc(x[0]);
  read(ch);
 end;
 if(x[1]=0)and(x[0]=0) then x[0]:=1;
//Зберігання 0 як довгого числа
end;
```

```
procedure writelong(x:ar);
//Процедура запису "довгого числа"
var ii:word;
    s1,s2:string;
begin
//Виведення 0, а не "0000"
 if (x[0]=1)and(x[1]=0) then
writeln(x[1])
 else
 begin
 str(osn div 10,s1);
 write(x[x[0]]);
 for ii:=x[0]-1 downto 1 do
  begin
   str(x[ii],s2);
//Доповнення запису числа 0-ми
   while length(s2)<length(s1) do
s2:='0'+s2;
   write(s2);
  end;
  writeln;
 end;
end;
procedure Rizn(var x:ar; y:ar;
sp:integer);
```

```
{Знаходження різниці "довгих чисел"
x і y при зсувові sp та збереженні
результату в x}
var ii,jj:integer;
begin
 for ii:=1 to y[0] do
 begin
  dec(x[ii+sp],y[ii]);
  jj:=ii;
  while (x[jj+sp]<0)and(jj<=x[0]) do
   begin
    inc(x[jj+sp],osn);
    dec(x[jj+sp+1]);
    inc(jj);
   end;
 end;
 ii:=x[0];
 while(ii>1)and(x[ii]=0) do dec(ii);
 x[0]:=ii;
end;
procedure Mul_kor(kk:Longint; y:ar;
var z:ar);
{Множення "довгого числа" y на
"коротке" kk із записом результату в
z}
var ii:integer;
```

```
begin
 z:=y;
 fillchar(z,sizeof(z),0);
 z[0]:=y[0];
 if kk=0 then inc(z[0])
 else
  begin
   for ii:=1 to y[0] do
    begin
z[ii+1]:=(LongInt(y[ii])*kk+z[ii])
div osn;
z[ii]:=(LongInt(y[ii])*kk+z[ii]) mod
osn;
    end;
    if z[y[0]+1]>0
     then z[0]:=y[0]+1
     else z[0]:=y[0];
  end;
end;
procedure mult_long(x,y:ar; var
z:ar);
{Множення "довгого числа" x на
"довге" y із записом результату в z}
begin
 if x[0]<y[0] then
 begin
```

```
  z:=x;
  x:=y;
  y:=z;
 end;
fillchar(z,sizeof(z),0);
z[0]:=x[0];
for i:=1 to x[0] do
for j:=1 to y[0] do
begin
dv:=LongInt(x[i])*y[j]+z[i+j-1];
inc(z[i+j],dv div osn);
z[i+j-1]:=dv mod osn;
end;
z[0]:=x[0]+y[0];
while (z[0]>1)and(z[z[0]]=0) do
dec(z[0]);
end;
procedure rd_l(yy:word; var z:ar);
{Перетворення "короткого числа" yy в
"довге" із записом результату в z}
 var kk:word;
 begin
  fillchar(z,sizeof(z),0);
  kk:=1;
  while yy>0 do
   begin
```

```
    z[kk]:=yy mod osn;
    yy:=yy div osn;
    kk:=kk+1;
   end;
   z[0]:=kk-1;
 end;
function More(x,y:ar; zsuv:integer):
byte;
{Реалізація функції, яка повертає 0
якщо x>y, 1 – y>x, 2 при y=x при
зсувові на zsuv позицій}
var ii:integer;
begin
 if x[0]>(y[0]+zsuv)
  then More:=0
  else
   if x[0]<(y[0]+zsuv)
    then More:=1
    else
     begin
      ii:=x[0];
      while (ii>zsuv)and(x[ii]=y[ii-
zsuv]) do dec(ii);
      if ii=zsuv
       then
        begin
```

```
          More:=0;
          for ii:=1 to zsuv do
           if x[ii]>0 then exit;
          More:=2;
         end
       else More:=Byte(x[ii]<y[ii-
zsuv]);
     end;
end;
function FindBin(var zal:ar; y:ar;
sp:integer):LongInt;
{Підбір цифр результату методом
поділу відрізка пополам}
var down,up:word;
//Нижня та верхня межа інтервала
    cc:ar;
//Допоміжне багаторозрядне число
begin
 down:=0;
 up:=osn;
 while up-1>down do
  begin
   Mul_kor((Up+Down) div 2,y,cc);
   case More(zal,cc,sp) of
   0: down:=(down+up) div 2;
   1: Up:=(down+up) div 2;
```

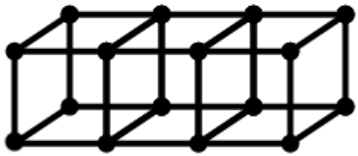
```
   2: begin Up:=(down+up) div 2;
down:=up; end;
   end;
  end;
  Mul_kor((Up+Down) div 2,y,cc);
  if More(zal,cc,0)=0
  then Rizn(zal,cc,sp)
  else
   begin
    Rizn(cc,zal,sp);
    zal:=cc;
   end;
   FindBin:=(Up+Down) div 2;
end;
procedure MakeDel(x,y:ar; var
cil,zal:ar);
{Знаходження результату ділення
багаторозрядних чисел }
var sp:integer;
begin
 zal:=x;
 sp:=x[0]-y[0];
 if More(x,y,sp)=1 then dec(sp);
 cil[0]:=sp+1;
 while sp>=0 do
  begin
```

```
    cil[sp+1]:=findbin(zal,y,sp);
    dec(sp);
   end;
end;
procedure LongDivLong(x,y:ar; var
cil,zal:ar);
{Цілочисельне ділення двох
багаторозрядних чисел x і y й
повернення цілої частини та залишку}
begin
 fillchar(cil,sizeof(cil),0);
 fillchar(zal,sizeof(zal),0);
 case More(x,y,0) of
 0: MakeDel(x,y,cil,zal);
 1: zal:=a;
 2: cil[1]:=1;
 end;
end;
function comb(nn,kk:word):ar;
{Обчислення C_nn^kk як багаторозрядного
числа}
var ii,jj,ff:integer;
    x,y,aa,zz,cil,zal:ar;
begin
 if kk>(nn-kk) then kk:=nn-kk;
 fillchar(x,sizeof(x),0);
```

```
 fillchar(y,sizeof(y),0);
 x[0]:=1;
 x[1]:=1;
 for ii:=1 to kk do
 begin
  Mul_kor(nn-ii+1,x,y);
  rd_l(ii,aa);
  LongDivLong(y,aa,cil,zal);
  x:=cil;
 end;
 comb:=x;
end;
begin
 readln(n,s);
 fillchar(b,sizeof(b),0);
 fillchar(a,sizeof(a),0);
 Mul_kor(s,comb((n div 2)+s-2,s-
1),b);
 a:=comb(n+s-1,s-1);
 Rizn(a,b,0);
 writelong(a);
 readln;
end.
```

Задача E – Сірникова модель

Професор Самодєлкін вирішив змайструвати об’ємну модель кубиків з сірників, використовуючи сірники для ребер кубиків. Довжина ребра кожного кубика дорівнює одному сірнику.



Для побудови моделі трьох кубів в нього пішло 28 сірників.

Яку найменшу кількість сірників потрібно Самодєлкіну для побудови моделі з N кубиків.

Усі числа задачі не перевищують $2 \cdot 10^9$.

Вхідні дані: одне число N – кількість кубиків.

Вихідні дані: одне число – кількість сірників.

Приклад:

Вхідні дані	**Вихідні дані**
3	28

Розв'язання

Задача має №3 на ресурсі [6], а розв'язання подане за [3]. У класифікації сказано, що вона відноситься до задач на моделювання та просту математику. Для знаходження потрібної формули використаємо метод динамічного програмування.

Слідом за [3] розіб'ємо розв'язування задачі на 3 підзадачі:

1. Знайдемо формулу формування одного ряду квадратів (1D).
2. Знайдемо формулу формування квадратів на площині (2D).
3. Використовуючи розв'язання двох попередніх підзадач знайдемо формулу для просторового випадку.

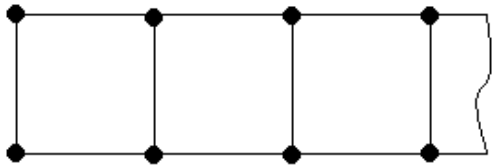


Рис. 2

При формуванні одного ряду квадратів необхідно знайти мінімальне число сірників, щоб скласти N квадратів зі стороною в 1 сірник (див. рис. 2).

Для зберігання результату використаємо масив S, кожна комірка якого буде містити кількість сірників для побудови відповідного квадрата. Щоб побачити закономірність, заповнимо даний масив даними, що подані у таблиці:

N	0	1	2	3	4
S	0	4	3	3	3

Очевидно, що S[0]:=0; S[1]:=4; S[i]:=3 (для $i > 1$). Тоді загальна кількість сірників, які необхідно використати для побудови лінії із N квадратів може бути обчислена за формулою:

$$Sum := 4 + 3 * (N - 1).$$

Можна записати таку функцію:

```
function D1(NN:integer):qword;
begin
 if NN=0
 then D1:=0
```

```
 else D1:=4+3*(NN-1);
end;
```

Перша підзадача розв’язана.

Дізнаємося, яку мінімальну кількість сірників слід використати для побудови квадратів на площині (рис.3).

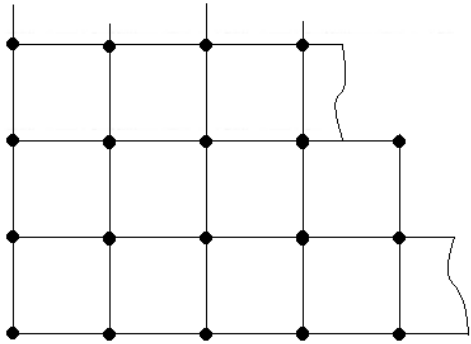


Рис. 3

При розв’язанні другої підзадачі слід уточнити до якої фігури (многокутника чи прямокутника) потрібно добудовувати щоб використовувати мінімальну кількість сірників. Зрозуміло із рис.4, що чим більше сусідніх квадратів, тим менше необхідно сірників для їх побудови. Так, при побудові 4 квадратів у вигляді перевернутої літери Г потрібно 13 сірників, а будуючи квадрат 2х2 – 12 сірників.

Найкращий варіант, коли при побудові квадрата є два сусіди.

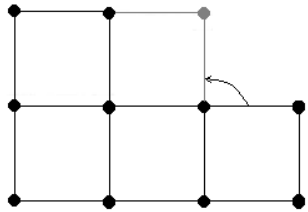


Рис. 4

Знову сформуємо допоміжну таблицю для кращого виявлення закономірності.

N	1	2	3	4	5	6	7	8	9
S	4	7	10	12	15	17	20	22	24

Оскільки у такому вигляді дана таблиця не дає повного уявлення про закономірність, то спробуємо подати її з використання рядка розмірності плоскої фігури.

N	1	2	3	4	5	6	7	8	9
Розмір	1x1	2x1	2x2	2x2	3x2	3x2	3x3	3x3	3x3
S	4	3	3	2	3	2	3	2	2

Проаналізуємо побудовану таблицю. Бачимо що при N=1 S[1]:=4, а наступні

значення залежать від ряду, який формується. Коли при формуванні фігури додається новий ряд, то при побудові квадрата використовуються 3 сірники, а потім – 2 сірники до кінця ряду.

Загальна ідея конструювання сірникової моделі на площині полягає в тому що, якщо вже побудована фігура $A \times B$, то слід будувати фігуру $(A + 1) \times B$, а потім $(A + 1) \times (B + 1)$. Побудуємо ще одну таблицю з використанням лінійного випадку.

N для D2	1	2	3	4	5	6	7	8	9
N для D1	1	1	1	2	1	2	1	2	3
Розмір	1x1	2x1	2x2	2x2	3x2	3x2	3x3	3x3	3x3
S	4	3	3	2	3	2	3	2	2

Тоді ідея побудови алгоритму полягає в організації циклу від 1 до N. У циклі слід збільшувати спочатку значення A, а потім B і проходити по масиву S з використанням функції D1. Але для D1 слід починати з

початкового значення і йти до найбільшого з них

Наперед визначимо для формули знаходження розв'язку підзадачі D2 змінні: N_{pk} – кількість повних квадратів, N_{dk} – додаткові числа, які прив'язані до підзадач з меншою розмірністю. Сама формула запишеться:

$N_{pk} = \left[\sqrt{N} - 1\right]$;

Якщо $N \leq N_{pk} \cdot (N_{pk} + 1)$,

то $N_{dk} = N_{pk} + 1$

інакше $N_{dk} = 0$.

$NN := \left(N_{pk} + N_{dk}\right)$;

$Sum := 4 + 3 * NN + 2 * (N - NN)$.

Для тривимірного випадку ідея мінімалізації кількості сірників при побудові кубів також полягає у бажанні побудувати якомога більший куб. При побудові проміжних варіантів будемо здійснювати такі переходи в конструкції:

$A \times B \times C \rightarrow (A + 1) \times B \times C \rightarrow (A + 1) \times (B + 1) \times C \rightarrow (A + 1) \times (B \times 1) \times (C + 1)$.

При написанні формули знаходження кількості сірників при побудові N кубиків слід враховувати кількість повних кубів, квадратів та ліній, а також враховувати додаткові числа, які прив'язані до підзадач з меншою розмірністю.

У наступному коді на мові Free Pascal c1 – кількість повних кубів, c2 – кількість повних квадратів, c3 – кількість повних ліній з певним корегуванням наближених обчислень, а s – кількість сірників затрачених на побудову моделі:

```
c1:=trunc(power(n,1/3)+0.5);
c2:=trunc(sqrt(n/c1)+0.5);
c3:=trunc(n/(c1*c2));
s:=3*c1*c2*c3+ 2*(c1*c2+c1*c3+c2*c
3)+ (c1+c2+c3);
```

Проте така формула видає правильні результати лише в граничних випадках. Якщо ж таких випадків немає, то слід виконати корекцію кількості сірників на площині. Така корекція виконана за допомогою змінної h в операторі розгалуження.

Код програми на Free Pascal:

```
uses math;
var c1,c2,c3,s,h,d1,d2,n:int64;
begin
readln(n);
c1:=trunc(power(n,1/3)+0.5);
c2:=trunc(sqrt(n/c1)+0.5);
c3:=trunc(n/(c1*c2));
s:=3*c1*c2*c3+2*(c1*c2+c1*c3+c2*c3)+
(c1+c2+c3);
h:=n-c1*c2*c3;
if h<>0 then
 begin
  d1:=trunc(sqrt(h)+0.5);
  if d1*d1>h then dec(d1);
  d2:=trunc(h/d1);
  s:=s+3*d1*d2+2*(d1+d2)+1;
  h:=h-d1*d2;
  if h<>0 then s:=s+3*h+2;
 end;
 writeln(s);
end.
```

ДЖЕРЕЛА:

1. Алгоритмы: построение и анализ, 2-е изд./ Т. Х. Кормен, И. И. Лейзерсон, Р. Л. Ривест, К. Штайн. – М.: Изд-ский дом «Вильямс», 2005. – 1296 с.
2. Виленкин Н. Я. Комбинаторика. / Н. Я. Виленкин, А. Н. Виленкин, П. А. Виленкин. – М.: ФИМА, МЦНМО, 2010. – 400 с.
3. Зайцев Я. Динамическое программирование. Спичечная модель. [Електронний ресурс] / Я. Зайцев. // – Режим доступу: habrahabr.ru/post/113720/ (дата звернення: 21.10.2015). – Назва з екрану.
4. Мерзляк А.Г. Математика: Підручник для 6 класу. / А.Г. Мерзляк, В.Б. Полонський, М.С. Якір. – Х.: Гімназія, 2006. – 304 с.
5. Окулов С. М. Программирование в алгоритмах. – М.: БИНОМ. Лаборатория знаний, 2002. – 341 с.
6. www.e-olymp.com [Електронний ресурс]