# **Й.А.Ентін, вчитель інформатики Дніпровського ліцею інформаційних технологій при ДНУ імені Олеся Гончара**

# **Дніпропетровські обласні олімпіади МАНУМ з інформатики – приклади задач**

Подано коротку інформацію про обласні олімпіади МАНУМ Дніпропетровської області з інформатики, які проводяться на базі Дніпровського ліцею інформаційних технологій при ДНУ імені Олеся Гончара. Для ілюстрацій представлено приклади двох задач та їх розв’язання – авторські та учнівські.

**ВСТУП**

З 2015 року в Дніпропетровській області проводяться обласні олімпіади Малої академії наук України. Інформатика – один із предметів, з якого проводяться ці олімпіади.

До проведення цієї олімпіади в учнів був один шлях до обласної олімпіади: шкільна-районна-міська олімпіади. Звичайно, певна кількість обдарованих учнів не потрапляла таким шляхом до обласної олімпіади. Олімпіада МАН дозволила розпаралелити шлях до обласної олімпіади.

Організуючи олімпіаду МАН, слід було впоратися з кількома проблемами.

1. Часи проведення цієї олімпіади та міських предметних олімпіад збігаються. Отже, проводити олімпіаду МАН у вихідні було нерозумно.

2. Організація олімпіадних змагань у навчальні дні обмежує їх тривалість і складність задач. Та недоцільно давати і надто прості задачі: тоді переможці олімпіади МАН не будуть конкурентоспроможними на обласній олімпіаді.

Було прийнято наступні рішення.

1. Олімпіада складається з двох турів, тривалість туру – дві астрономічні години.

2. Перший тур – заочний, проводиться в середині листопаду. Його переможці запрошуються до участі в другому, очному, який проводиться через чотири тижні після першого і є очним. Переможці олімпіади визначаються за підсумками другого туру (перемога в першому надає тільки право участі у другому турі). Другий тур проводиться після міської олімпіади.

3. Тур складається із трьох задач. Звичайно, задачі другого туру складніші від задач першого.

Мови програмування олімпіади – C++ (консольний додаток Visual Studio 2010 або пізніших версій, CodeBlocks 13.12), C# (консольний додаток Visual Studio 2010 або пізніших версій), Mono C#. Free Pascal. Учасники одержують можливість відправляти файли рішень на сайт e-olymp для перевірки.

У першому турі можуть брати участь усі, хто бажає і зареєструвався на сайті Дніпровського ліцею інформаційних технологій при ДНУ імені Олеся Гончара. Олімпіади проводяться на базі цього ліцею. Поки що учасників небагато (не більше від 50). Але результати виступів переможців олімпіад МАН на обласних олімпіадах успішні: протягом 2015-2018 рр. усі, хто одержав право участі в обласній олімпіаді через олімпіаду МАН, стали призерами обласної олімпіади, Тричі переможці олімпіади МАН брали участь у Всеукраїнській олімпіаді з програмування, двічі вони вибороли призові дипломи Всеукраїнської олімпіади.

Далі - аналіз алгоритмів та програм розв’язування двох задач другого туру олімпіади 2016 року. По кожній із цих задач тільки один учень одержав максимальні бали. Корисно порівняти підходи учнів та автора задач до їх розв’язувань.

## **Д.А.Байнак[[1]](#footnote-1), Й.А.Ентін**

## **Алгоритми розв’язування задачі «Сходи з чотирьох сходинок (stair4)».**

## **Умова задачі.**

Є **n** однакових за розмірами брусів у формі прямих призм із квадратом в основі. Довжина сторони основи та висота брусу такі, що він може бути сходинкою або її частиною, якщо покласти його горизонтально бічною гранню. Скільки варіантів сходів (sum) із чотирьох сходинок можна побудувати з **n** брусів? Правила побудови: 1) підйом по сходах – зліва направо; 2) висота сходинки - довжина сторони квадрата основи призми, якою є брус; 3) ширина сходинки – кількість усіх брусів на її рівні; 4) між брусами немає щілин; 5) не може бути двох сходинок однакової ширини.

Приклади сходів із чотирьох сходинок для **n**=12 показані на малюнках. Для малюнка зліва 12=1+2+4+5 (складаємо ширини сходинок від найвищої до найнижчої). Малюнок справа демонструє варіант 12=1+2+3+6. Інших варіантів сходів із 12 брусів, що задовольняють умові задачі, немає.

**Технічні умови**

10≤**n**≤30000. Значення n ввести зі стандартного пристрою введення. Значення sum вивести на стандартний пристрій виведення.

**Приклади**

**n** sum

12 2

73 2178

345 272916

**Авторське розв’язання**

Почнемо з підрахунку на папері кількості варіантів розв’язань у залежності від кількості брусів. Очевидно, що коли брусів менше від десяти, розв’язання немає: десять – найменше число, яке є сумою чотирьох різних натуральних чисел.

**Таблиця 1. Зв’язок кількості брусів із кількістю розв’язань задачі**.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Висота  сходів | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 |
| Кількість наборів довжин сходинок | 1 | 1 | 2 | 3 | 5 | 6 | 9 | 10 | 14 | 19 |

Для того, щоб зміст таблиці 1 було легше зрозуміти, додано таблицю 2, де наведено всі можливі варіанти побудови сходів, які відповідають умові задачі для заданої кількості брусів n. Окремо виділено набори варіантів для заданої пари перших чисел у наборі. Для зручності ширѝни сходинок у кожному з наборів розташовані за зростанням (це відповідає спуску по сходах згори донизу). Таблиця виявилася великою, тому її довелося розподілити на дві частини: перша – для n від 10 до 14, друга – для n від 15 до 19. Якщо скласти по вертикалі кількості варіантів, записані у вузьких клітинах таблиці 2, одержимо кількості наборів довжин сходинок, представлені в таблиці 1.

**Таблиця 2. Зв’язок кількості брусів із кількостями варіантів побудови сходів для різних наборів ширин сходинок**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Кількість  брусів | 10 |  | 11 |  | 12 |  | 13 |  | 14 |  |
| Варіанти формування сходів (широка  клітина – представлення варіантів, вузька – їх кількість для даної пари перших чисел). | 1 2 3 4 | 1 | 1 2 3 5 | 1 | 1 2 3 6 | 2 | 1 2 3 7 | 2 | 1 2 3 8 | 3 |
|  |  |  |  | 1 2 4 5 | 1 2 4 6 | 1 2 4 7 |
|  |  |  |  |  |  | 1 3 4 5 | 1 | 1 2 5 6 |
|  |  |  |  |  |  |  |  | 1 3 4 6 | 1 |
|  |  |  |  |  |  |  |  | 2 3 4 5 | 1 |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
| Кількість брусів | 15 |  | 16 |  | 17 |  | 18 |  | 19 |  |
| Варіанти формування  сходів (широка клітина – представлення варіантів, вузька – їх кількість для даної пари перших чисел). | 1 2 3 9 | 3 | 1 2 3 10 | 4 | 1 2 3 11 | 4 | 1 2 3 12 | 5 | 1 2 3 13 | 5 |
| 1 2 4 8 | 1 2 4 9 | 1 2 4 10 | 1 2 4 11 | 1 2 4 12 |
| 1 2 5 7 | 1 2 5 8 | 1 2 5 9 | 1 2 5 10 | 1 2 5 11 |
| 1 3 4 7 | 2 | 1 2 6 7 | 1 2 6 8 | 1 2 6 9 | 1 2 6 10 |
| 1 3 5 6 | 1 3 4 8 | 2 | 1 3 4 9 | 3 | 1 2 7 8 | 1 2 7 9 |
| 2 3 4 6 | 1 | 1 3 5 7 | 1 3 5 8 | 1 3 4 10 | 3 | 1 3 4 11 | 4 |
|  |  | 1 4 5 6 | 1 | 1 3 6 7 | 1 3 5 9 | 1 3 5 10 |
|  |  | 2 3 4 7 | 2 | 1 4 5 7 | 1 | 1 3 68 | 1 3 6 9 |
|  |  | 2 3 5 6 | 2 3 4 8 | 2 | 1 4 5 8 | 2 | 1 3 7 8 |
|  |  |  |  | 2 4 5 6 | 1 4 6 7 | 1 4 5 9 | 2 |
|  |  |  |  |  |  | 2 3 4 9 | 3 | 1 4 6 8 |
|  |  |  |  |  |  | 2 3 5 8 | 1 5 6 7 | 1 |
|  |  |  |  |  |  | 2 3 6 7 | 2 3 4 10 | 3 |
|  |  |  |  |  |  | 2 4 5 7 | 1 | 2 3 5 9 |
|  |  |  |  |  |  | 3 4 5 6 | 1 | 2 3 6 8 |
|  |  |  |  |  |  |  |  | 2 4 5 8 | 2 |
|  |  |  |  |  |  |  |  | 2 4 6 7 |
|  |  |  |  |  |  |  |  | 3 4 5 7 | 1 |

Таблиця 2 наочно показує, звідки взялися дані, представлені в таблиці 1. Але головне –зміна кількостей варіантів сходів з однаковою парою перших чисел! Тут ми одержуємо ключ до побудови алгоритму розв’язування. На жаль, не вдалося знайти рекурентну формулу, яка описувала б залежність кількості варіантів сходів для довільної сходинки в залежності від кількостей варіантів для менших кількостей брусів. Допомогло спостереження за змінами кількостей варіантів сходів з однаковою парою перших чисел при зростанні кількостей брусів. Перелічимо основні помічені закономірності, що дозволили побудувати алгоритм розв’язання задачі.

1. Кількість варіантів сходів з найменшими двома першими числами 1 2 визначається по-різному для парних та непарних n. Якщо n парне, кількість таких варіантів дорівнює цілій (з відкиданням дробної частині) частці від ділення n-7 на 2. Якщо ж n непарне, кількість таких варіантів дорівнює цілій (з відкиданням дробної частині) частці від ділення n-9 на 2.

2. Для парної кількості брусів при сталому першому числі в сумі зміна другого числа дає кількість варіантів, меншу на два від попередньої. Наступна зміна другого числа дає зменшення суми варіантів на 1, далі – знову на 2, ці зменшення чергуються.

3. Якщо змінюються і перше, і друге числа, кількість варіантів буде на два менше, ніж у першого варіанту попереднього набору. Приклад: при n=18 маємо найбільше варіантів для ширин найвищих сходинок 1 та 2 (5). Для ширин 1 та 3 їх 3=5-2. Для ширин 1 та 4 їх 2 (3-1), а далі віднімаємо 2 і одержимо 0 – таких варіантів більше немає. Для ширин найвищих сходинок 2 та 3 маємо 3 варіанти (5-2=3), для ширин 2 та 4 – один варіант, для ширин 3 та 4 – один варіант (1=3-2).

4. Для непарної кількості брусів маємо такі відмінності від спостереженого для парної: а) найбільша кількість варіантів для ширин найвищих сходинок 1 та 2 залежить від n як цілі частина частки від ділення n-9 на 2; 2) при зміні другого числа в сумі ширин кількість варіантів для наступної пари перших двох ширин зменшується на 1, а при наступній зміні другої ширини 0 - на 2. Всі інші закономірності – такі ж, як для парного значення n.

Далі представлено програми авторського розв’язання цієї задачі двома мовами програмування. Коментарі записано курсивом (для Free Pascal - у фігурних дужках, для CodeBlocks – після пари слешів).

**Програма мовою Free Pascal**

var

n,i,x,k,v,sum:int64;{*Опис змінних*}

Begin

sum:=0;

writeln('n=');

readln(n);{*Задано значення* n}

writeln; {*Технічна вставка – щоб числа не напливали одне на одне*.}

if n mod 2=0

then begin {*Починаємо розрахунок для парного* n.}

v:=(n-7) div 2;

while v>0 do

begin

x:=v;

k:=0;

while x>0 do{*Значення x введено для підрахунку сумарної кількості варіантів при незмінній ширині найвищої сходинки*.}

begin

sum:=sum+x;

k:=k+1;

if k mod 2=1 {*Це – перехід до варіанту зі зміною ширини другої згори сходинки*.}

then x:=x-2

else x:=x-1;

end;

v:=v-2;{*Перехід при збільшенні ширини найвищої сходинки*.}

end

end

else begin {*Початок підрахунку кількості варіантів для непарної кількості брусів* n}

v:=(n-9)div 2;

while v>0 do

begin

x:=v;

k:=0;

while x>0 do

begin

sum:= sum+x;

k:=k+1;

if k mod 2=1

then x:=x-1

else x:=x-2;

end;

v:=v-2;

end

end;

writeln(sum); ;*{Виведення результату.}*

Readln *{Пауза перед завершенням роботи.}*

End.

**Програма мовою C++ (CodeBlocks)**

#include <bits/stdc++.h>//Підключення необхідних бібліотек та просторів імен.

using namespace std;

int main()

{

int n;

long long x,v,k,sum;

cin>>n;//*Введення значення n.*

sum=0;

if(n%2==0)//*Якщо n парне*

{//*Тут і далі – такі ж дії, як у попередній програмі. Тому коментарі мінімальні.*

v=(n-7)/2;

while(v>0)

{

x=v;

k=0;

while(x>0)

{

sum+=x;

k++;

if(k%2==0)

x-=2;

else

x--;

}

v-=2;

}

}

else

{

v=(n-9)/2;

while(v>0)

{

x=v;

k=0;

while(x>0)

{

sum+=x;

k++;

if(k%2==0)

x--;

else

x-=2;

}

v-=2;

}

}

cout <<sum << endl;//*Виведення результату.*

system("pause");//*Пауза перед завершенням роботи.*

return 0;

}

**Розв’язання учасника олімпіади (CodeBlocks)**

Оскільки головна ознака олімпіади – дефіцит часу на прийняття рішень, тут запропоновано евристичний алгоритм.

#include <bits/stdc++.h>

using namespace std;

int main() {

ios::sync\_with\_stdio(false);//*У перших рядках підключено необхідні бібліотеки та описано змінні та вимкнення синхронізацій задля прискорення введення-виведення*

cin.tie(0);//*Вимкнення синхронізації з потоком виведення.*

cout.tie(0);//*Вимкнення синхронізації з потоком вводу.*

int n;

cin >> n;//*Ввели n*.

long long c2=0;

int c=0,n1,n2,n3,n4,nk=-1,nnn,r,k,nn;

for (n1 = n-6; n1 >= max(n/4,1); n1--) {//*Цикл пошуку через змінну n1 – кількість //кубиків в останній (найбільшій) сходинці.*

nn = n-n1; **//***Кількість сходинок без останньої*

for (n2 = min(n1-1,nn); n2 >= max(nn/3,1); n2--) {//*Вкладений цикл по* *n2 –кількість //кубиків в попередній за останню сходинку**–* *від /меншого з nn та n-n1 до найбільшого з //nn та n1-1 – за зменшенням.*

nnn = nn-n2; **//***Кількість сходинок без останньої та передостанньої.*

if (nnn > (n2<<1)-3 || !(n1 && n2)) continue;//*Якщо ця умова виконується,* ***то //****можливих варіантів вже не буде, отже**переходимо до початку ітерації.*

r = nnn-n2;

k = ((nnn-1) >> 1)-(r>0?r:0); //*Кількість можливих варіантів на певній ітерації****.***

if(k>0) {//*Перевірка на існування знайдених варіантів.*

c2+=(long long) k;//*Приведення до довгого цілого та додавання отриманих* ***//****варіантів.*

}

}

}

cout << c2 << endl;//*Виводимо результат.*

system("PAUSE");

return 0;

}

Розв’язок знайдений наступним чином. Нескладно написати три вкладених цикли: пербір із перевіркою умов, необхідних для правильного розв’язку. Та тоді обчислювальна складність кубічна, що не дозволяє вкластися в обмеження за часом. Тому шукано формулу для відмови від останнього циклу й забезпечення квадратичної складності алгоритму. Всі тести успішно.

За обчислювальною складністю алгоритми автора і учня квадратичні. Трохи полегшує справу те, що починаємо не від n, а хоча б від n-7, націло поділеного на 2 і довжини кроків – то 1, то 2. Отже, обчислювальна складність не вища від n2.

## **Й.А.Ентін, В.В.Савельєв[[2]](#footnote-2)**

## **Алгоритми розв’язування задачі «Стартова сума (startsum)»**

## **Умова задачі**

Три різні цифри – x, y, z – є взаємно простими натуральними числами. Визначити число min – таке, що всі натуральнічисла, не менші відmin,можна представити у вигляді k\*x+j\*y+i\*z. k, j, i – натуральні числа.

**Технічні умови.**

Вхідні величини **(**x, y, z) ввести зі стандартного пристрою введення, 1<x<y<z. Результат (min) вивести на стандартний пристрій виведення.

**Приклади**

Вхідні дані min

2 3 5 12

5 7 8 32

**Авторське розв’язання**

Ця задача має довгу передісторію. Зокрема, для двох взаємно простих чисел x та y розв’язання відоме: це xy+1. Для трьох таких чисел не вдалося знайти інформації. Тому використано перебір. Оскільки в умові задачі йдеться тільки про цифри (тривалість олімпіади невелика), всі можливі приклади неважко прорахувати вручну. Такі підрахунки показали, що для всіх трійок взаємно простих цифр min<100. Числові досліди дозволили встановити, що коли знайдено найменше q, для якого в інтервалі значень [q;q+x] всі числа є лінійними комбінаціями x, y, z, min дорівнює цьому q.

Кроки алгоритму вибрано такими.

1.Пишемо вкладені цикли по i, j, k. Обчислюємо у внутрішньому циклі k\*x+j\*y+i\*z, збільшуємо змінну m на 1 та записуємо знайдене значення до a[m]. Головна проблема цього пункту: одержувані в циклі числа не дають впорядкованого масиву, тому невідомо, до яких значень i, j, k доведеться вести обчислення. Є ризик, ввівши наче велику кількість елементів, не заповнити інтервал, в якому всі числа є лінійними комбінаціями наших взаємно простих цифр. Отже, слід експериментувати з максимальними значеннями параметрів циклів.

2. Оскільки треба аналізувати впорядковані значення елементів одержаного масиву, сортуємо його за зростанням (оскільки він не дуже великий, можна застосувати алгоритм простих обмінів). Тут маємо продовження проблеми пункту 1: невідомо, скільки елементів доведеться враховувати в сортуванні, щоб не втратити точний результат: легко одержати у відповіді таке значення, що перевищує реальне min.

3. Тепер пересуваємося по відсортованому масиву від великих значень до малих, перевіряючи на кожному кроці, чи буде різниця між сусідніми елементами масиву не більшою від одиниці. Якщо знайдено таку пару, для якої вказана різниця перевищує одиницю, min дорівнює більшому числу цієї пари. Треба починати не від останніх елементів масиву: оскільки вони генеруються не у порядку зростання і найбільші з них може розділяти різниця, більша від одиниці. Треба гарантувати, що елементи масиву, від яких починаємо пошук min, або дорівнюють сусіднім, або перевищують їх на одиницю.

Маємо комбінацію перебору з евристикою. Остання дозволяє скоротити кількість обчислень (зокрема, для цього чим менше число з трійки, тим глибше вкладено цикл для його параметра). Далі – програма авторського розв’язання на CodeBlocks. Оскільки вона за використаними алгоритмами та засобами мало відрізняється від коду на Free Pascal, програму на Free Pascal не наведено.

#include <bits/stdc++.h>

using namespace std;

int main()

{

int const nmax=2000;

int n,b,c,d,i,j,k,v,x,z;

int a[nmax];

cin>>x>>y>>z;//x<y<z

v=0;

for(i=1;i<13;i++)

for(j=1;j<13;j++)

for(k=1;k<13;k++)//*Максимальні значення параметрів циклів визначені дослідом*.

{

v++;

a[v]=z\*i+y\*j+x\*k;//*Формуємо масив сум лінійних комбінацій x, y, z*.

}

for(i=2;i<441;i++)//*Сортуємо одержаний масив за зростанням.*

for(j=440;j>=i;j--)

{

if (a[j-1]>a[j])

{

x=a[j-1];

a[j-1]=a[j];

a[j]=x;

}

}

n=25;//*Значення n підібрано в дослідах.*

while(n>1 &&(a[n]-a[n-1]<=1))//*Пошук min.*

n--;

cout<<a[n]<<endl;//*Виводимо min та індекс n відповідного елемента масиву.*

cout<<n<<endl;//

system("pause");

return 0;

}

**Розв’язання учасника олімпіади.**

Обмеження в умові невеликі, тому використано перебір із використанням структури даних set (множина) та її властивостей. Зручності використання множини наступні. По-перше, будь-яке значення може міститися в ній, на відміну від масиву, тільки один раз. По-друге, множина автоматично розташовує всі свої елементи в порядку зростання. По-третє, можливості швидкого доступу до усіх значень і просте додавання значень до множини (обчислювальна складність - O(log(n)), де n – кількість елементів у структурі даних. По-четверте, можна обійтися без масиву, що прискорює роботу програми і зменшує необхідний обсяг пам’яті.,

Опис програми.

// Код:

#include <bits/stdc++.h>//*Підключення бібліотек, введення даних.*

#define qq cout << "!!" << endl;

#define ww cout << "??" << endl;

#define rr return 0;

#define nl cout << endl;

using namespace std;

int a, b, c, m = 0;//*Опис змінних.*

set<int> A;

int main() {

cin >> a >> b >> c;//*Далі – перебір потрібних чисел*

for(int i = 1; i <= 30; i++)

{

for(int j = 1; j <= 30; j++)

{

for(int h = 1; h <= 30; h++)

{

A.insert(a \* i + b \* j + c \* h);//*Додавання числа до множини.*

}

}

}/\**Проходимо по всіх значеннях множини доки знайдемо довгий ланцюжок із*

*натуральних чисел, різниця між якими в ланцюжку не більша від одиниці. Ланцюжок вважаємо довгим, якщо в ньому не менше від 100 чисел./*

int t = 0, count = 0;

for(auto i : A)

{

if(i - t == 1)/\**Якщо різниця між сусідніми елементами дорівнює одиниці,*

*збільшуємо лічильник на одиницю.* \*/

{

count++;

}

else

{

count = 0;/\* *Якщо різниця між сусідніми елементами більша від одиниці,*

*обнулюємо лічильник і фіксуємо значення, яке є розв’язанням задачі.\*/*

m = i;

}

t = i;

if(count > 100)/\**Якщо числа, різниця між якими дорівнює одиниці,*

*зустрілися більше від ста разів, виводимо відповідь.\*/*

{

cout << m << endl;

system("pause"); rr;

}

}

}

Обчислювальна складність використаного алгоритму – O(n^3 log(n)), де n = 30. Отже, заміна масиву структурою set прискорює роботу програми, хоча і при роботі з set доводиться використовувати певні евристики.

**Заключна частина.**

Сподіваюся, що представлена у статті інформація доводить плідність проведення обласних олімпіад МАН. Маючи зв’язки з деякими випускниками ліцею, можу стверджувати, що успішні учасники олімпіад швидше стають успішними професіоналами. На олімпіадах вони навчилися швидко розбиратися в незнайомій інформації, швидко знаходити правильні рішення, швидко та раціонально думати. З олімпіадних задач нерідко виростають теми дослідницьких робіт учнів. Останні приклади (2019 р.) – Андрій Шевцов, що виборов диплом II ступеня на Всеукраъському конкурсы МАН і призери обласного конкурсу МАН Сергій Єрмаков (диплом II ступеня) і Борис Зелікман (диплом III ступеня). Ці учні – майбутні автори статей до щорічного збірника наукових робіт ліцеїстів «МІФ» («Математика», «Інформатика», «»Фізика»), що виходить із 2010 року.

**Література**

1.Окулов С. М. Программирование в алгоритмах. - М.: БИНОМ Лаборатория знаний, 2007 – 383 с.

2.Інформатика («Шкільний світ») № 21 (509), 2009 – 48 с.

1. Байнак Дмитро Андрійович – випускник Дніпровського ліцею інформаційних технологій 2017 року, неодноразовий призер обласних олімпіад з інформатики, призер Всеукраїнської олімпіади з інформатики 2016 року, студент ДНУ.Н [↑](#footnote-ref-1)
2. Владислав Володимирович Савельєв – випускник Ліцею інформаційних технологій (2018), студент університету м. Ополе (Польща) призер обласних олімпіад із програмування, має диплом III ступеня Всеукраїнської олімпіади з програмування-2016. [↑](#footnote-ref-2)