ОЛІМПІАДА З ІНФОРМАТИКИ У МІСТІ КИЄВІ У 2014–2015 НАВЧАЛЬНОМУ РОЦІ

Мисак Данило Петрович,
керівник гуртка СШ №52 м. Києва.
Рибак Олександр Владиславович,
аспірант Інституту математики НАН України.
Рудик Олександр Борисович,
доцент Київського університету імені Бориса Грінченка.

Стаття містить умови завдань II (районного) і завдань III (міського) етапу олімпіади з основ інформатики й обчислювальної техніки у місті Києві у 2014/2015 навчальному році та авторські розв'язання цих завдань. Публікацію адресовано учням класів з поглибленим вивченням математики, учасникам олімпіад з інформатики, студентам математичних спеціальностей, учителям і викладачам вищих навчальних закладів. Цього навчального року, як і попереднього, упорядником завдань II етапу був Данило Мисак.

I. ЗАВДАННЯ II ЕТАПУ

Максимальна оцінка за кожну з чотирьох задач — 100 балів. Для всіх задач обмеження на час — 1 секунда/тест; обмеження на пам'ять — 256 МБ.

1. Стрілки

Назва програми: **hands.pas/hands.cpp**

В Орисі та Марисі є по одному улюбленому числу в межах від 1 до 12 включно, причому ці числа не обов'язково різні. З'ясуйте, скільки протягом доби є таких моментів часу, що хоча б одна зі стрілок годинника (годинна або хвилинна) точно вказує на улюблене число принаймні однієї з дівчат.

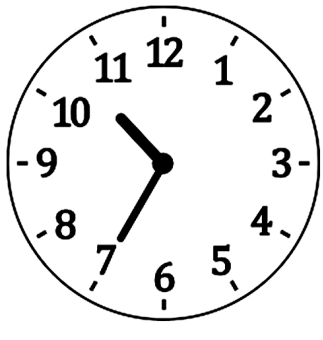

Рис. 1

Вхідні дані. У вхідному файлі вказано два натуральних числа: улюблене число Орисі та улюблене число Марисі.

Вихідні дані. У вихідний файл виведіть кількість відповідних моментів часу протягом однієї доби.

Приклад

hands.in	hands.out
2 7	52

2. Сірники

Назва програми: **matches.pas/matches.cpp**

Орися та Марися люблять гратися з сірниками. Кожна з дівчат має по n сірників. Орися хоче викласти із своїх сірників якомога більше число, а Марися — якомога менше число, причому дівчата хочуть використати всі свої сірники. Допоможіть Орисі та Марисі. Те, як із сірників викладаються цифри, ви можете подивитися на рисунку 2.

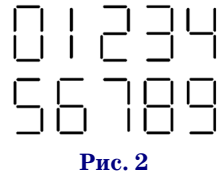

Рис. 2

Дівчата знають, що ставити на початок чисел нулі не можна.

Вхідні дані. У вхідному файлі записано натуральне число n. Воно є не меншим за 2 і не перевищує 2000.

Вихідні дані. У вихідний файл через пробіл виведіть два числа: найбільше і найменше натуральні числа, які можуть викласти дівчата саме з n сірників.

Приклад

matches.in	matches.out
5	71 2

3. Цукерки

Назва програми: **candy.pas/candy.cpp**

Якось Орисі наснилися n гномів, у кожного з яких була певна додатна кількість цукерок, що ділиться на $n-1$. На Новий рік кожен гном розділив свої цукерки на $n-1$ рівну частину та подарував усім іншим гномам по одній такій частині. Коли Орися розповіла про цей сон Марисі, тій стало цікаво, скільки ж цукерок було спочатку в кожного гнома. На жаль, Орися запам'ятала тільки те, скільки цукерок опинилося в кожного гнома після святкування Нового року. Допоможіть дівчатам за цією інформацією відновити початкові кількості цукерок.

Вхідні дані. У першому рядку вхідного файлу вказано натуральне число n, не менше за 2 і не більше за 200 000. У другому рядку записано n натуральних чисел, менших за мільйон, — кінцеві кількості цукерок у гномів.

Вихідні дані. У вихідний файл виведіть початкові кількості цукерок у кожного з гномів у тому самому порядку, у якому гноми задані у вхідному файлі. Якщо можливих відповідей декілька, виведіть будь-яку з них. Якщо жодного варіанта відповіді, що задовольняє умову задачі, не існує, виведіть лише одне число 0.

Приклади

candy.in	candy.out
4 6 5 4 6	3 6 9 3
3 4 4 5	0

Пояснення до прикладів

При початковому розташуванні цукерок 3, 6, 9, 3 перший гном отримає від другого 6/(4–1)=2 цукерки, від третього — 9/(4–1)=3 цукерки, а від четвертого — 3/(4–1)=1 цукерку (разом він матиме 2+3+1=6 цукерок, бо свої три він віддасть); другий гном отримає 3/3+9/3+3/3=5 цукерок; третій матиме 3/3+6/3+3/3=4

цукерки; четвертий — 3/3+6/3+9/3=6 цукерок. Можна переконатися, що в другому прикладі жодного можливого початкового розташування не існує.

4. Поїзди

Назва програми: **trains.pas/trains.cpp**

У місті, де живуть Орися та Марися, є n ліній метрополітену і m станцій (деякі з них можуть бути станціями пересадки і належати відразу кільком лініям). Усі станції занумеровано натуральними числами від 1 до m. Орися живе біля станції з номером 1, а Марися — біля станції з номером m. Знаючи, скільки хвилин забирає проїзд між кожними двома сусідніми станціями на кожній лінії, визначте, за який найменший час Орися зможе доїхати до Марисі. Відомо, що між станціями, де живуть дівчата, існує сполучення (можливо, з пересадками). Пересадка між лініями, а також вхід у метро (на будь-яку лінію) та вихід (з будь-якої лінії) здійснюють миттєво. Поїзди ходять щохвилини, тож Орися не чекатиме поїздів на станціях. Часом зупинки поїзда на станції також можна знехтувати. На всіх лініях поїзди рухаються в обидва боки.

Вхідні дані. У першому рядку вхідного файлу записано два натуральних числа n та m, які не перевищують 500. Кожен з наступних n рядків задає лінію метро: перше число в рядку — кількість станцій на лінії (не менша за 2). Далі вказано такі натуральні числа: номер першої станції на лінії; кількість хвилин, які забирає поїздка між першою та другою станціями лінії; номер другої станції лінії; кількість хвилин, які забирає поїздка між другою і третьою станціями лінії; номер третьої станції лінії і т. д. Жодна станція не може повторюватися на одній і тій самій лінії двічі, тобто поїзди не ходять по колу і лінії метро не перетинають самі себе. Крім того, жодні дві станції не можуть бути сусідніми відразу на двох різних лініях. Час, необхідний на переїзд між будь-якими двома сусідніми станціями, не перевищує 8 хвилин. У 50% тестів цей час дорівнює 1 хвилині для всіх пар сусідніх станцій.

Вихідні дані. У вихідний файл виведіть єдине число — найменшу кількість хвилин, за яку Орися зможе доїхати до Марисі.

Приклад

trains.in	trains.out
3 9 4 5 2 7 2 8 2 4 7 6 1 9 1 8 3 2 2 7 1 1 1 3 2 1 8 9	4

Пояснення до прикладу

Маємо три лінії метро, що проходять через дев'ять станцій. Схему ліній та час переїзду між кожними двома сусідніми станціями показано на рисунку 3.

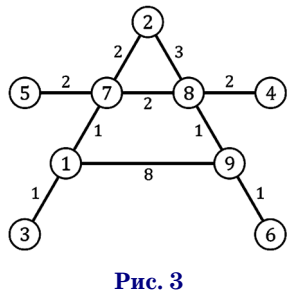

Рис. 3

Щоб дістатися з першої до дев'ятої станції якнайшвидше, зробимо так: сядемо на другу лінію (з сімома станціями), доїдемо до сьомої станції, пересядемо на першу лінію (з чотирма станціями), доїдемо до восьмої станції та знову пересядемо на другу лінію, вийдемо на дев'ятій станції. Це забере 1+2+1=4 хвилини. Альтернативні маршрути без пересадок тривають 1+2+3+1=7 хвилин по другій лінії та 8 хвилин по третій.

II. ІДЕЇ РОЗВ'ЯЗАННЯ ЗАВДАНЬ II ЕТАПУ

1. Стрілки

Якщо улюблені числа дівчат різні, то є рівно 2×2=4 моменти протягом доби, коли на одне з них вказує годинна стрілка та 2×24=48 моментів, коли на одне з них вказує хвилинна стрілка; загалом 4+48=52 моменти. При цьому деякі з них ми могли порахувати двічі — ті, під час яких і годинна, і хвилинна стрілка показують на улюблені числа дівчат. Але якщо годинна стрілка показує на ціле число, то хвилинна має показувати на 12. Якщо одне з чисел справді дорівнює 12, то двічі ми порахували чотири моменти: два, коли обидві стрілки показують на 12, і два, коли хвилинна показує на 12, а годинна — на улюблене число іншої дівчини. Відповідь у такому випадку дорівнює 52−4=48. Якщо ж серед чисел немає числа 12, то відповідь — 52.

Якщо улюблені числа дівчат однакові, то є рівно 2 моменти, коли на них указує годинна стрілка, та 24 моменти, коли на них вказує хвилинна стрілка. Разом 24+2=26 моментів. Одночасно стрілки можуть вказувати на числа лише тоді, коли ті дорівнюють 12. Тоді моментів, які ми порахували двічі, — два, а відповіддю є 26−2=24.

2. Сірники

Випишемо, скільки сірників містить кожна цифра:

0	1	2	3	4	5	6	7	8	9
6	2	5	5	4	5	6	3	7	6

Найменшу кількість сірників — два — містить цифра 1, а найбільшу — сім — цифра 8. Як відомо, число є тим більшим, що більшою є в ньому кількість цифр, а за однакової кількості цифр число є більшим, якщо має більші старші розряди. Спершу розглянемо те, яке *максимальне* число можна утворити з n сірників.

- Якщо n парне, тобто $n=2k$, то число може містити щонайбільше k цифр, причому k цифр воно міститиме тоді й лише тоді, коли всі вони — одиниці. Отже, Орися складе число 11...1, де кількість одиниць дорівнює $n/2$.
- Якщо n непарне, тобто $n=2k+1$, то число також може містити щонайбільше k цифр, причому k цифр воно міститиме тоді й лише тоді, коли всі цифри, крім однієї, одиниці, а цифра, відмінна від одиниці, містить три сірники. Такою є лише цифра 7; при цьому сімку, очевидно, є сенс поставити на перше місце. Отже, Орися складе число 711...1, де кількість одиниць дорівнює $(n-3)/2$.

Тепер розглянемо варіанти для *мінімального* числа, яке можна утворити рівно з n сірників.

- Якщо n ділиться на 7, тобто $n=7k$, то число може містити щонайменше k цифр, причому k цифр воно міститиме тоді й лише тоді, коли всі вони — вісімки. Отже, Марися складе число 88...8, де кількість вісімок дорівнює $n/7$.

- Якщо $n=7k+1$, то число може містити щонайменше $k+1$ цифру, при цьому першою цифрою може бути одиниця, а другою — нуль. Тоді решта $(n-8)/7$ цифр — вісімки.
- Якщо $n=7k+2$ ($k\geq 0$), то число може містити щонайменше $k+1$ цифру, при цьому першою цифрою може бути одиниця. Тоді решта $(n-2)/7$ цифр — вісімки.
- Якщо $n=7k+3$ ($k\geq 0$), то число може містити щонайменше $k+1$ цифру, при цьому одиниця першою бути не може (бо інакше на решту k цифр залишилося $7k+1>7k$ сірників). Тоді у випадку $k=0$ перша і єдина цифра дорівнює 7, а інакше першою цифрою можна зробити 2. Далі, якщо $k=1$, наступна цифра остання і повинна містити 5 сірників. Найменшою такою цифрою знову є 2. Отже, відповідь — 22. Якщо ж $k>1$, то дві наступні цифри можна зробити нулями, а решту $(n-17)/7$ цифр доведеться зробити вісімками.
- Якщо $n=7k+4$ ($k\geq 0$), то число може містити щонайменше $k+1$ цифру, при цьому одиниця першою бути знову не може. Тоді у випадку $k=0$ перша і єдина цифра дорівнює 4, а інакше першою цифрою можна зробити 2. Наступна цифра може бути нулем, а решта $(n-11)/7$ цифр — вісімки.
- Якщо $n=7k+5$ ($k\geq 0$), то число може містити щонайменше $k+1$ цифру, при цьому одиниця першою бути не може. Але першою цифрою можна зробити 2; решта $(n-5)/7$ цифр — вісімки.
- Якщо $n=7k+6$ ($k\geq 0$), то число може містити щонайменше $k+1$ цифру, при цьому на першому місці повинна стояти ненульова цифра, що містить принаймні 6 сірників, найменша з таких — цифра 6. На решті $(n-6)/7$ місць стоятимуть вісімки.

Наостанок зауважимо, що відповіді можуть виявитися досить великими і не вміщатися у межі стандартних типів, тому їх слід виводити не як числа, а просто як послідовність цифр.

3. Цукерки

Позначимо початкові кількості цукерок у гномів через a_k, а кінцеві через b_k, $1\leq k\leq n$. Загальна кількість цукерок у гномів не змінилась, позначимо її через S:

$$S=b_1+b_2+\ldots+b_n=a_1+a_2+\ldots+a_n.$$

Крім того, згідно з умовою задачі

$$b_k=\frac{a_1}{n-1}+\frac{a_2}{n-1}+\ldots+\frac{a_{k-1}}{n-1}+\frac{a_{k+1}}{n-1}+\ldots+\frac{a_n}{n-1}=$$
$$=\frac{a_1+a_2+\ldots+a_n}{n-1}-\frac{a_k}{n-1}=\frac{S}{n-1}-\frac{a_k}{n-1},$$

звідки $a_k=S-(n-1)b_k$. Якщо всі такі значення діляться на $n-1$ (а це буде тоді й лише тоді, коли на $n-1$ ділиться число S) і є додатними, то вони становлять відповідь: у цьому нескладно переконатися, підставивши їх у вираз

$$\frac{a_1+a_2+\ldots+a_n}{n-1}-\frac{a_k}{n-1}$$

і пересвідчившись, що його значення справді дорівнює b_k. Інакше потрібно вивести нуль.

Додамо, що перераховувати суму чисел кожного разу не слід, адже тоді складність виконання алгоритму стане квадратичною від кількості гномів і для великих значень n програма буде працювати дуже повільно. Достатньо порахувати суму чисел один раз. Вона може виявитися досить великою, тому у програмі потрібно оперувати 64-бітовими змінними.

4. Поїзди

Заданий у вхідному файлі план ліній метрополітену слід подати у програмі як звичайний неорієнтований граф зі зваженими ребрами (вага ребра — кількість хвилин, які поїзд їде від однієї станції до іншої).

Для пошуку найкоротшого маршруту між вершиною 1 та вершиною k можна використати алгоритм Дейкстри, причому на повний бал достатньо найпростішої його реалізації, що працює за квадратичний від кількості вершин час.

Інший підхід — штучно розбити кожне ребро ваги k графа додатковими проміжними вершинами на k одиничних ребер. Далі застосувати пошук у ширину. Кількість ребер графа при цьому зростає не більш ніж у 8 разів, а нових вершин додається стільки ж, скільки й нових ребер. Отже, час виконання пошуку в ширину залишається фактично квадратичним від початкової кількості вершин.

Оскільки у 50 % тестів час на проїзд між будь-якими двома сусідніми станціями складає 1 хвилину, простий пошук у ширину на графі з ігноруванням ваг ребер заробить 50 балів. Якщо при цьому виводити не довжину маршруту, а суму ваг ребер на ньому, програма може пройти додатково ще один чи два тести.

3. АВТОРСЬКІ РОЗВ'ЯЗАННЯ ЗАВДАНЬ II ЕТАПУ

1. Стрілки

```
/* GCC */
#include<stdio.h>

intmain()
{
freopen("hands.in", "r", stdin);
freopen("hands.out", "w", stdout);

inta, b, ans;

// Зчитування даних:
scanf("%d %d", &a, &b);

// Обчислення відповіді:
if(a == 12 || b == 12)
ans = 24;
else
ans = 26;
if(a != b)
ans *= 2;
// Виведення відповіді:
printf("%d\n", ans);
return0;
}
```

2. Сірники

```
/* GCC */
#include<stdio.h>

intmain()
```

```
{
freopen("matches.in", "r", stdin);
freopen("matches.out", "w", stdout);

intn;

// Зчитування кількості сірників:
scanf("%d", &n);

// Виведення максимального числа, яке
   // можна скласти з n сірників:
printf(n % 2 == 0 ? "1" : "7");
for(inti = 0; i < n/2–1; i++)
printf("1");

printf(" ");

// Виведення мінімального числа, яке
   // можна скласти з n сірників:
inteights = 0; // Кількість вісімок у
               // числі (буде оновлю-
               // ватись далі в коді)
switch(n % 7)
  {  // Залежно від остачі при діленні
     // n на 7:

case0:
eights = n/7;
break;
case1:
printf("10");
eights = (n–8)/7;
break;
case2:
printf("1");
eights = (n–2)/7;
break;
case3:
if(n/7 == 0)
printf("7");
elseif(n/7 == 1)
printf("22");
else
{
printf("200");
eights = (n–17)/7;
       }
break;
case4:
if(n/7 == 0)
printf("4");
else
{

printf("20");
eights = (n–11)/7;
       }
break;
case5:
printf("2");
eights = (n–5)/7;
break;
case6:
printf("6");
eights = (n–6)/7;
break;
  }
// Виводимо вісімки:
for(inti = 0; i <eights; i++)
printf("8");
printf("\n");
return0;
}
```

3. Цукерки

```
{FreePascal }

constmaxN = 200000; // Максимальна можлива
                    // кількість гномів

vara, b: array[1..maxN] ofint64; // По-
                    // чаткові та кінцеві кіль-
                    // кості цукерок у гномів
S: int64; // Загальна кількість цукерок
i, n: longint;
ans: boolean; // Чи існує відповідь

begin
assign(input, 'candy.in');
reset(input);
assign(output, 'candy.out');
rewrite(output);

  S := 0;

// Зчитування даних та підрахунок
   // загальної кількості цукерок:
read(n);
fori := 1 ton dobegin
read(b[i]);
     S := S + b[i];
end;

ans := (S mod (n–1) = 0);
// Підрахунок відповіді:
if(ans) then
fori := 1 ton dobegin
a[i] := S–(n–1) * b[i];
ifa[i] <= 0 thenbegin// Якщо
ans := false; // вийшло не-
break;   // додатне число,
end;     // відповіді не існує
end;

// Виведення відповіді:
if(ans) thenbegin
fori := 1 ton–1 do
write(a[i], ' ');
writeln(a[n]);
endelse
writeln(0);
end.
```

4. Поїзди

```
/* GCC */
#definemaxM500   // Найбільша можлива
                 // кількість станцій
#include<stdio.h>
#include<vector>
usingnamespacestd;

vector<pair<int, int>>graph[maxM];
// graph[i][j].first — номер вершини,
// суміжної з вершиною i (нумерація з нуля)
// graph[i][j].second — вага відповідного
// ребра

intdistances[maxM];
// Поточні відстані від початкової вершини
// (-1, якщо дана вершина поки недосяжна)

boolvisited[maxM];
// Індикатор того, чи додано на даний
// момент відповідні вершини до тієї
// частини графа, де відстані пораховані
// остаточно

intmain()
{
freopen("trains.in", "r", stdin);
freopen("trains.out", "w", stdout);

intn, m;

// Зчитування даних та побудова графа:
scanf("%d %d", &n, &m);
for(inti = 0; i < n; i++)
  {
intstations, current, previous,
time;
scanf("%d %d", &stations,
&current); // Кіль-
           // кість станцій на лінії
           // та номер першої станції
for(intj = 0; j <stations−1;
        j++)
    {
previous = current;
scanf("%d %d", &time,
&current); // Час
             // на переїзд та номер
             // наступної станції
       // Додаємо ребро ваги time між
       // вершинами previous та
       // current, зважаючи, що
       // нумерація вершин у нашій
       // програмі повинна починатися
       // не з одиниці, а з нуля:
graph[previous−1].push_back(
make_pair(current−1, time));
graph[current−1].push_back(
make_pair(previous−1, time));
    }
  }
// Шукаємо найкоротший шлях від вершини
  // 0 до всіх інших вершин графа:
for(inti = 0; i < m; i++) // Ініціа-
{       // лізація масивів відстаней
          // і доданості
distances[i] = −1;
visited[i] = false;
  }
intaddedVertex = 0; // Починаємо
                    // з вершини 0
distances[addedVertex] = 0;
while(addedVertex != −1) // Поки є
{    // нова вершина, яку слід додати
visited[addedVertex] = true; // По-
          // значаємо вершину як додану
for(inti = 0;
        i <graph[addedVertex].size();
        i++) // Для кожного ребра,
{       // що з неї виходить
intvertex = // Номер суміжної
                    // вершини
graph[addedVertex][i].first;
intweight = // Вага відповід-
                    // ного ребра
graph[addedVertex][i].second;
intdistance =
distances[addedVertex]
         + weight; // Час, за який
       // можна потрапити у vertex, якщо
       // йти через addedVertex
if(distances[vertex] == −1
|| distances[vertex]
>distance) // Якщо зна-
            // йдений час менший за
            // наявний, оновлюємо його
distances[vertex] =
distance;
     }
// Яку вершину слід додати
     // наступною:
addedVertex = −1;
for(inti = 0; i < m; i++) // Для
                    // кожної вершини
if(! visited[i]
&&distances[i] != −1 // Якщо
    // її ще не додано, але вона досяжна
&& (addedVertex == −1
|| distances[i]
<distances[addedVertex]))
// І якщо відстань до неї менша
    // за поточний мінімум
addedVertex = i;
  }

// Виводимо відповідь: найкоротшу від-
  // стань від вершини 0 до вершини m-1
printf("%d\n", distances[m−1]);

return0;
}
```

(Далі буде)