Задача A. Двійкові маніпуляції

Назва вхідного файлу:	`sort.in`
Назва вихідного файлу:	`sort.out`
Обмеження використання часу:	0.1 секунда
Обмеження використання пам'яті:	256 мегабайтів

У Козака Вуса суцільні двійки з інформатики... Ні, не тому що він погано володіє комп'ютером. Просто віднедавна усі завдання слід виконувати на спеціальному автоматі, який дуже полюбляє степені двійки.

Степінь двійки — число вигляду 2^x, де x — ціле невід'ємне число. Наприклад, числа $1, 2, 4, 8, 16$ — степені двійки, а ось $3, 7, 10, 15, 19$ — ні.

Автомат оперує масивом цілих чисел a довжини n, причому n — степінь двійки. Усе, що вміє автомат, — переставити число з t-ої позиції на початок масиву, зсунувши при цьому всі числа, що стоять від початку до цієї позиції. Наприклад, якщо $a = [1, 2, 3, 4, 5, 6, 7, 8]$ і ми переставляємо 4-ий елемент на початок, отримаємо $a = [4, 1, 2, 3, 5, 6, 7, 8]$. Зверніть увагу, що автомат уміє виконувати цю операцію лише для t, що є **степенем двійки**.

Вус зі всіх сил намагається виконати останнє домашнє завдання — для заданого масиву написати послідовність $t_1, t_2, \ldots, t_k$, яка б перетворила масив на відсортований за неспаданням. Щось йому вдається, але він просить Вас зробити те саме, щоб звірити відповіді. Допоможіть йому, знайшовши таку послідовність!

Формат вхідних даних

Перший рядок містить два цілі числа n та g ($2 \leqslant n \leqslant 128$, $0 \leqslant g \leqslant 6$) — довжина масиву та номер блоку. Гарантується, що n — степінь двійки.

Наступний рядок містить рівно n цілих чисел $a_1, a_2, \ldots, a_n$ ($1 \leqslant a_i \leqslant n$) — елементи масиву.

Формат вихідних даних

У першому рядку виведіть одне ціле число k ($0 \leqslant k \leqslant 16384$) — кількість команд.

У другому рядку виведіть k цілих чисел $t_1, t_2, \ldots, t_k$ ($1 \leqslant t_i \leqslant n$) — послідовність команд, що сортує масив. Таких послідовностей може бути багато, тому виведіть будь-яку (не обов'язково найменшої довжини).

Приклади

sort.in	sort.out
4 0 4 3 2 1	4 2 4 4 2
4 0 1 3 1 2	3 4 2 4

Примітка

У першому прикладі масив був $[4, 3, 2, 1]$. Після перестановки другого елемента отримаємо $[3, 4, 2, 1]$. Далі двічі переставляємо останній елемент, отримуючи спочатку $[1, 3, 4, 2]$, а потім $[2, 1, 3, 4]$. Останнім ходом переставляємо другий елемент на перше місце, отримуючи $[1, 2, 3, 4]$ — відсортований за неспаданням масив.

У другому прикладі масив був $[1, 3, 1, 2]$. Після перестановки четвертого елемента отримаємо $[2, 1, 3, 1]$. Далі переставляємо другий елемент, отримуючи $[1, 2, 3, 1]$. Укінці переставляємо останній елемент, отримуючи $[1, 1, 3, 4]$ — упорядкований масив.

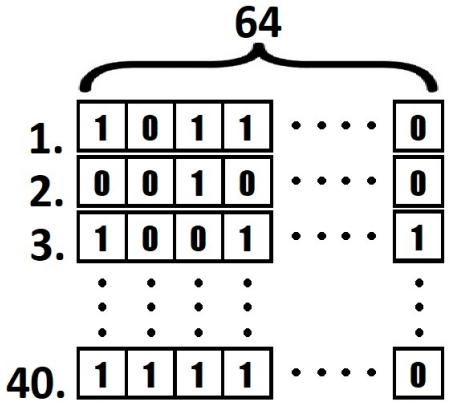
Оцінювання

1. (11 балів) $n = 2$; усі числа різні;
2. (15 балів) $n = 4$; усі числа різні;
3. (20 балів) $n = 8$; усі числа різні;
4. (22 бали) рівно $\frac{n}{2}$ чисел масиву — одиниці, усі інші — двійки;
5. (23 бали) усі числа різні;
6. (9 балів) без додаткових обмежень.

Задача B. Козак Вус і програмування

Обмеження використання часу:	1 секунда
Обмеження використання пам'яті:	256 мегабайтів

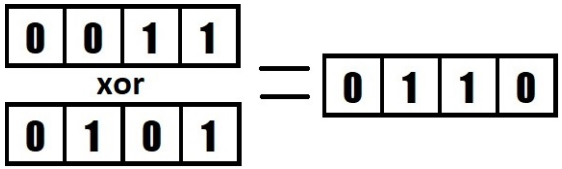
Усім відомо, що у Козака є його улюблений робот Атлас, з яким він багато грався у дитинстві. Сьогодні Вус зрозумів, що ще ніколи не намагався запрограмувати робота на якісь інші дії, і тому почав читати інструкцію. З'ясувалося, що у пам'яті Атласа зберігаються 40 пронумерованих з одиниці регістрів, кожний із яких містить 64 біти. Це означає, що кожен регістр складається з 64-ох нулів та одиниць:



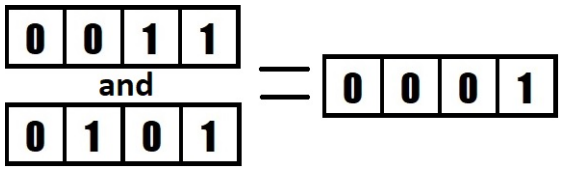
Вус також зрозумів, що для програмування робота можна використовувати наступні 7 команд, які виконують операції над регістрами. Для зручності будемо регістром i називати регістр під номером i, а також a_i — число, яке записане в i-му регістрі

1. `setValue(i, j)` — замінити регістр i на регістр j. Тобто кожен біт першого регістра замінити на відповідний біт у другому регістрі. Іншими словами, виконати операцію присвоєння $a_i = a_j$.
2. `setXor(i, j, k)` — замінити регістр i на побітовий `xor` регістрів j та k. Тобто кожен біт регістра i замінити на `xor` відповідних бітів у регістрів j та k. Про операцію `xor` буде зазначено нижче.
3. `setAnd(i, j, k)` — замінити регістр i на побітовий `and` регістрів j та k. Тобто кожен біт регістра i замінити на `and` відповідних бітів у регістрів j та k. Про операцію `and` буде зазначено нижче.
4. `setOr(i, j, k)` — замінити регістр i на побітовий `or` регістрів j та k. Тобто кожен біт регістра i замінити на `or` відповідних бітів у регістрів j та k. Про операцію `or` буде зазначено нижче.
5. `shiftLeft(i, x)` — **зсунути** всі біти регістру i на x позицій вліво. Про операцію `зсуву` буде зазначено нижче.
6. `shiftRight(i, x)` — **зсунути** всі біти регістру i на x позицій вправо. Про операцію `зсуву` буде зазначено нижче.
7. `setNot(i, j)` — замінити регістр i на **інвертований** регістр j. Тобто кожен біт регістра i замінити на 1, якщо на відповідній позиції регістра j записане 0, і замінити на 0, якщо на відповідній позиції регістра j записане 1.

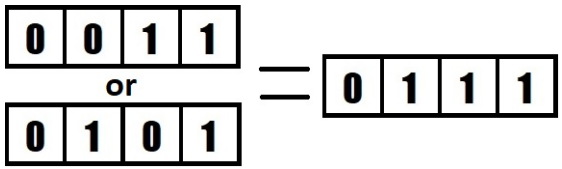
Результатом операції `xor` двох бітів є 0, якщо біти однакові, та 1, якщо різні.



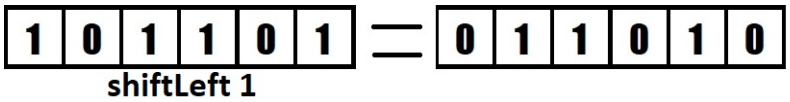
Результатом операції `and` двох бітів є 1, якщо обидва біти є рівними 1, та 0, якщо хоча б один біт рівний 0.



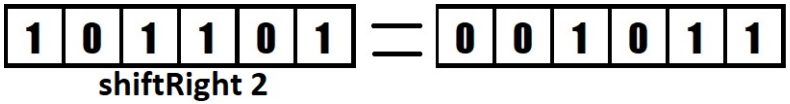
Результатом операції `or` двох бітів є 0, якщо обидва біти є рівними 0, та 1, якщо хоча б один біт рівний 1.



Результатом операції `зсув` регістру на x позицій уліво є регістр, у якому кожний біт рівний біту, що був записаний у початковому регістрі на x позицій правіше, та рівний 0, якщо не існувало у початковому регістрі позиції на x правіше.



Результатом операції `зсув` регістру на x позицій управо є регістр, у якому кожний біт рівний біту, що був записаний у початковому регістрі на x позицій лівіше, та рівний 0, якщо не існувало у початковому регістрі позиції на x лівіше.



Прочитавши інструкцію, Козак Вус одразу ж придумав 6 досить цікавих задач. Ще трохи подумавши, Козак зрозумів, що для того, щоб запрограмувати робота на конкретну задачу, потрібно лише ввести роботу набір команд, який би вирішував цю задачу. Припустимо, що t — кількість команд, яку ви виконали в певній задачі.

А ось і самі задачі:

`Підзадача 1:` (до 6 балів) у регістрі 1 записане число x. Усі інші регістри повністю складаються з 0-ів. Нехай число y дорівнює 1, якщо число x парне, і дорівнює 0, якщо x непарне. Задача полягає в тому, щоб після виконання роботом операцій у регістрі 2 було записане число y. Ви отримаєте $\lfloor 6 - \sqrt[3]{t - \min(3, t)} \rfloor$ балів.

`Підзадача 2:` (до 10 балів) у регістрі 1 записане x, а в регістрі 2 записане y. Усі інші регістри повністю складаються з 0-ів. Нехай $z = x + y$. Задача полягає в тому, щоб після виконання роботом операцій у регістрі 3 було записане число z. Потрібно зазначити, що якщо двійковий запис числа z потребує більше ніж 64 розряди, то в регістрі 3 мають бути записані лише перші 64 розряди. Ви отримаєте $\lfloor 10 - \sqrt[3]{t - \min(260, t)} \rfloor$ балів.

`Підзадача 3:` (до 13 балів) у регістрі 1 записане x. Нехай число y — це кількість одиниць у двійковому записі числа x. Тоді після виконання роботом операцій у регістрі 2 має бути записане y. Ви отримаєте $\lfloor 13 - \log(t - \min(1505, t) + 1) \rfloor$ балів.

`Підзадача 4:` (до 23 балів) у регістрах 1 та 2 записані два **різні** числа x та y відповідно. Нехай якщо $x > y$, то $a = 0$, $b = 1$, а якщо $y > x$, то $b = 0$, $a = 1$. Задача полягає в тому, щоб після виконання роботом операцій у регістрах 3 та 4 були записані числа a та b відповідно. Ви отримаєте $\lfloor 23 - \sqrt[2.5]{t - \min(92, t)} \rfloor$ балів.

`Підзадача 5:` (до 18 балів) у регістрі 1 записане число x. Нехай число y — це кількість одиниць у двійковому записі числа x. Тоді після виконання роботом операцій у регістрі 1 має бути записане число $2^y - 1$. Ви отримаєте $\lfloor 18 - \sqrt[4]{t - \min(6560, t)} \rfloor$ балів.

`Підзадача 6:` (до 30 балів) нехай a — це масив, який містить 9 невід'ємних цілих **попарно різних** чисел. Тоді для кожного i від 1-ого до 9-и у i-ому регістрі записане a_i. Нехай b — це відсортований за зростанням масив a. Тоді після виконання роботом операцій, для кожного i від 1-ого до 9-и у i-ому регістрі має бути записане b_i. Потрібно зазначити, що нумерація у масивах a та b починається з одиниці. Ви отримаєте $\lfloor 30 - \sqrt[3]{t - \min(4644, t)} \rfloor$ балів.

Якщо про початкове значення регістру нічого не сказано, тоді на початку він повністю складається з нулів. Усі числа від 0 до $2^{64} - 1$.

Козак Вус якимось чином дізнався кількість команд, яка необхідна для вирішення кожної задачі, проте він ще не знає, які саме команди потрібно використати, і тому звернувся по допомогу до вас. Він просить вас знайти таку послідовність команд, яка розв'яже задачу для будь-яких можливих вхідних даних.

Зверніть увагу, що ви можете виконати не більше 10^5 команд. Якщо ви виконаєте більше, то ви отримаєте 0 балів та вердикт `Неправильна відповідь`. Якщо ви виконаєте неправильний запит, то ви отримаєте 0 балів та вердикт `Помилка виконання`. Результатом вашого рішення в кожній підзадачі будуть ті регістри, у яких мають бути записані рішення. В усіх інших регістрах можуть бути будь-які числа.

Протокол взаємодії

Для кожної підзадачі вам потрібно окремо реалізувати одну функцію:

```
void solve()
```

- ця функція не приймає ніяких параметрів та нічого не повертає.

Ви можете використовувати наступні функції:

```
void setValue(integer i, integer j)
```

- ця функція надає регістру i ($1 \leqslant i \leqslant 40$) значення регістра j ($1 \leqslant j \leqslant 40$).

```
void setXor(integer i, integer j, integer k)
```

- ця функція надає регістру i ($1 \leqslant i \leqslant 40$) значення операції `xor` регістрів j ($1 \leqslant j \leqslant 40$) та k ($1 \leqslant k \leqslant 40$).

```
void setAnd(integer i, integer j, integer k)
```

- ця функція надає регістру i ($1 \leqslant i \leqslant 40$) значення операції `and` регістрів j ($1 \leqslant j \leqslant 40$) та k ($1 \leqslant k \leqslant 40$).

```
void setOr(integer i, integer j, integer k)
```

- ця функція надає регістру i ($1 \leqslant i \leqslant 40$) значення операції `or` регістрів j ($1 \leqslant j \leqslant 40$) та k ($1 \leqslant k \leqslant 40$).

```
void shiftLeft(integer i, integer x)
```

- ця функція реалізовує `зсув` регістру i ($1 \leqslant i \leqslant 40$) на x ($0 \leqslant x \leqslant 64$) позицій уліво.

```
void shiftRight(integer i, integer x)
```

- ця функція реалізовує `зсув` регістру i ($1 \leqslant i \leqslant 40$) на x ($0 \leqslant x \leqslant 64$) позицій управо.

```
void setNot(integer i, integer j)
```

- ця функція надає регістру i ($1 \leqslant i \leqslant 40$) значення `інвертованого` регістру j ($1 \leqslant j \leqslant 40$).

Приклад

Припустимо, що в регістрі 1 записане число 6, а в регістрі 2 число 3, тоді якщо виконаємо такі команди:

```
setValue(3, 1)
setXor(4, 1, 2)
setAnd(5, 1, 2)
setOr(6, 2, 1)
shiftLeft(2, 3)
shiftRight(1, 2)
setNot(7, 2)
```

то перші сім чисел масиву будуть виглядати так: [1, 24, 6, 5, 2, 7, 18446744073709551591].

Задача C. Авіашляхи Потоколяндії

Назва вхідного файлу:	`air.in`
Назва вихідного файлу:	`air.out`
Обмеження використання часу:	2 секунди
Обмеження використання пам'яті:	256 мегабайтів

Нещодавно Козака Вуса було обрано на посаду голови Міністерства інфраструктури Потоколяндії, у якій n міст, пронумерованих цілими числами від 1 до n.

Зараз у Потоколяндії всі люди використовують автомобілі і немає жодного авіаперельоту. Саме тому Міністерство вирішило відкрити перші авіаперельоти в Потоколяндії. Відповідно до законодавства країни між кожною парою міст може бути не більше одного авіаперельоту.

Відомо, що у Козака є план відкриття авіаперельотів на наступні m років. У нього є m списків міст. Кожного року він хоче вибрати один список зі ще не вибраних та відкрити авіасполучення між кожною парою міст у цьому списку, що ще не є сполученими авіаперельотом. Зверніть увагу, що якщо в списку є два міста, між якими вже є авіапереліт, то відкривати новий **забороняється**.

Крім того, усім жителям Потоколяндії відомі так звані `три найважливіші числа`: a, b та c.

Для кожного списку i є певне число r_i — його важливість. Відомо, що якщо відкрити всі авіасполучення між містами в i-му списку, то це принесе $f(e) \cdot r_i$ грошових одиниць прибутку державі, де e — кількість нових авіаперельотів, які були відкриті цього року, а $f(e)$ — деяка функція, що визначається так: $f(e) = (a \cdot e^2 + b \cdot e + c) \bmod n$ (де $x \bmod y$ — залишок від ділення x на y).

Козак Вус задумався, який саме список потрібно використовувати кожного року, щоб сумарно через m років принести якомога більший прибуток державі.

Допоможіть Козаку, знайшовши максимальну кількість грошових одиниць, яку він може принести державі, якщо він може кожного року вибирати список, який він не використовував раніше, та відкрити нові авіаперельоти між кожною парою міст у цьому списку, що ще не сполучені авіашляхом.

Формат вхідних даних

Перший рядок містить три цілі числа n, m, g ($1 \leqslant n \leqslant 10^6$, $1 \leqslant m \leqslant 20$, $0 \leqslant g \leqslant 8$) — кількість міст, списків у Потоколяндії та номер блоку, до якого належить тест, відповідно.

Другий рядок містить три цілі числа a, b, c ($0 \leqslant a, b, c < n$) — `три найважливіші числа` Потоколяндії.

Третій рядок містить m цілих чисел $r_1, r_2, \ldots, r_m$ ($0 \leqslant r_i \leqslant 10^6$) — важливість i-го списку.

Кожний із наступних m рядків містить ціле число s_i ($1 \leqslant s_i \leqslant n$) та s_i цілих чисел $t_{i1}, t_{i2}, \ldots, t_{is_i}$ ($1 \leqslant t_{ij} \leqslant n$) — кількість міст в i-му списку та міста в цьому списку. Гарантується, що всі числа в списку різні.

Гарантується, що сума значень s не перевищує $3 \cdot 10^6$.

Формат вихідних даних

Виведіть одне ціле число — максимальну кількість грошових одиниць, яку може принести Козак державі.

Приклади

air.in	air.out
5 3 0 0 2 1 1 2 1 2 1 3 3 1 4 5 4 1 2 3 4	11
6 4 0 1 2 3 3 2 3 4 3 4 5 6 3 1 4 5 2 1 3 3 3 4 5	35

Примітка

У першому прикладі можна використовувати списки в такому порядку: $[1, 2, 3]$.

У другому прикладі можна використовувати списки в такому порядку: $[2, 1, 3, 4]$.

Оцінювання

1. (5 балів) $n \leqslant 10^3$; $m \leqslant 20$; усі значення s_i рівні 2; не існує двох списків, яким належить одна і та сама пара міст;
2. (7 балів) $n \leqslant 10^3$; $m \leqslant 20$; усі значення s_i рівні 2; $c = 0$;
3. (16 балів) $n \leqslant 50$; $m \leqslant 7$;
4. (14 балів) $n \leqslant 50$; $m \leqslant 12$;
5. (8 балів) $n \leqslant 10^5$; $m \leqslant 3$;
6. (17 балів) $n \leqslant 5 \cdot 10^4$; $m \leqslant 10$;
7. (7 балів) $n \leqslant 2 \cdot 10^5$; $m \leqslant 16$;
8. (26 балів) без додаткових обмежень.

Задача D. Козак Вус і найкраща країна

Обмеження використання часу:	2 секунди
Обмеження використання пам'яті:	1 гігабайт

Козак Вус нарешті знайшов країну своєї мрії. Вона складається з n міст, між якими не проходить жодна дорога. Звичайно, Вус захотів виправити цю ситуацію, і тому придумав m різних двосторонніх доріг, які можна було би прокласти. За його задумом кожна дорога з'єднує два різні

міста. Але виникла проблема: кожне місто i може виділити на побудову доріг c_i копійок, а для кожної дороги j на її побудову пішло б w_j копійок. Тому Козак вирішив, що йому вистачить побудувати декілька доріг так, аби всі міста стали сусідами. Міста називаються сусідами, якщо можливо дорогами країни дістатися з одного міста в інше.

При плануванні своїх робіт Козак Вус зрозумів одну річ: коли він будуватиме чергову дорогу, міста будуть об'єднуватися в групи сусідів. Щоб побудувати j-ту дорогу, потрібно, щоб міста (або їх сусіди), між якими будується дорога, сумарно мали принаймні w_j копійок (адже спершу потрібно заплатити за дорогу й лише потім будувати). Після побудови дороги бюджети міст об'єднуються, а вартість дороги вираховується з нової спільної казни.

Для кожного з n міст ви знаєте число c_i — кількість копійок, які може витратити місто i. Для кожної з m доріг ви знаєте числа v_i, u_i та w_i, які означають, що дорога i з'єднує міста v_i та u_i, а на її побудову потрібно w_i копійок. Скажіть, чи можливо вибрати певну кількість доріг та впорядкувати їх так, щоб усі міста стали сусідами. А якщо це можливо, то знайдіть послідовність доріг, у якій їх потрібно будувати.

Протокол взаємодії

Для того щоб показати дороги, які потрібно побудувати, використовуйте таку функцію:

```
void add(integer i)
```

- Ця функція будує дорогу під номером i ($1 \leqslant i \leqslant m$).

Вам потрібно реалізувати одну функцію:

```
boolean solve(integer n, integer m, integer g, array of integers c, array of integers v,
              array of integers u, array of integers w)
```

- n — кількість міст у країні;
- m — кількість доріг, які можна прокласти;
- g — номер блока;
- c_i ($|c| = n$) — початкова кількість копійок у i-тому місті;
- v_i та u_i ($|v| = |u| = m$) — номери вершин, що з'єднує i-та дорога;
- w_i ($|w| = m$) — кількість копійок, необхідних на побудування i-тої дороги;
- ця функція повинна повертати `true`, якщо можливо правильно вибрати послідовність доріг, і `false`, якщо ні.

Якщо ваша функція повертає `true`, то до того, як вона його поверне, вона має зробити виклики функції `add` для всіх побудованих доріг у тому порядку, у якому вони побудуються.

Формат вхідних даних

Перший рядок містить три цілі числа n, m та g ($1 \leqslant n \leqslant 10^6$, $0 \leqslant m \leqslant 10^6$, $0 \leqslant g \leqslant 7$) — кількість міст, кількість доріг та номер блока відповідно.

Наступний рядок містить n цілих чисел $c_1, c_2, \ldots, c_n$ ($1 \leqslant c_i \leqslant 10^6$) — початкова кількість копійок в i-му місті.

Кожний із наступних m рядків містить по три цілі числа v_i, u_i та w_i ($1 \leqslant v_i, u_i \leqslant n$, $1 \leqslant w_i \leqslant 10^6$) — номер міст, між якими i-та дорога, та ціна її побудови відповідно.

Формат вихідних даних

Якщо функція повертає `false`, то в першому рядку буде виведено одне число -1.

Інакше в першому рядку буде виведено число q — кількість побудованих доріг. У наступних q рядках буде виведено по одному число x — номер побудованої дороги.

Приклади

Вхідні дані	Вихідні дані
4 5 0 2 5 2 4 1 2 7 3 4 4 1 4 5 4 2 3 3 2 4	3 4 2 3
3 3 0 6 2 5 2 3 9 2 1 5 1 3 10	-1

Примітка

У першому прикладі країна складається з 4-ох міст, а план Козака Вуса складається з 5-и доріг. Спочатку будується дорога під номером 4: міста 2 та 4 об'єднаються у групу, а з їхнього загального бюджету сплачується 3 копійки. На рахунку цієї групи залишається 6 копійок. Після побудови дороги під номером 2 бюджет групи буде складати 4 копійки, адже приєдналося місто 3 з бюджетом у 2 копійки, а також витратили 4 копійки на побудову дороги. Після побудови 3-ої дороги всі міста стануть сусідами, а їх загального бюджету вистачить, щоб заплатити 5 копійок за дорогу.

У другому прикладі неможливо вибрати дороги та порядок їх побудови так, щоб усі міста стали сусідами й щоб за усі дороги було заплачено.

Оцінювання

1. (3 балів) $n \leqslant 10$; $m = n - 1$; $c_i = 1$; $w_i = 1$; якщо побудувати всі m доріг, усі міста стануть сусідами;
2. (8 балів) $n, m \leqslant 10$;
3. (12 балів) $n, m \leqslant 10^5$; $c_i = 1$;
4. (14 балів) $n, m \leqslant 10^5$; усі w_i однакові;
5. (16 балів) $n, m \leqslant 10^3$;
6. (19 балів) $n, m \leqslant 10^5$;
7. (12 балів) $n, m \leqslant 5 \cdot 10^5$;
8. (16 балів) без додаткових обмежень.