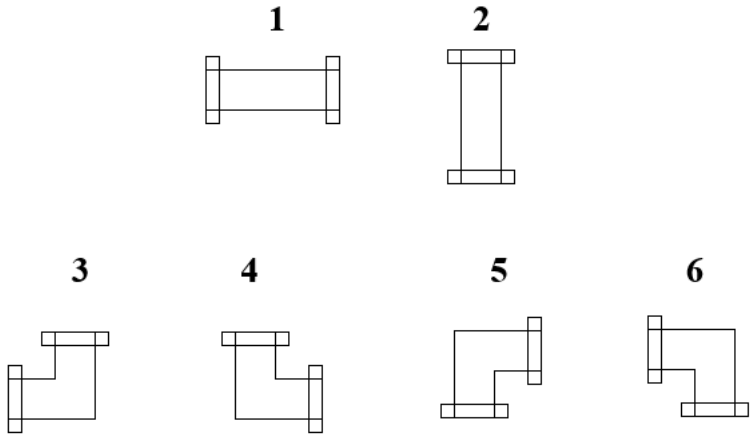
# Задача A. Труби

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

Козак Вус захотів стати президентом Потоколяндії, а для цього, як водиться, потрібно зробити якусь добру справу для добрих громадян, щоб вони стали ще добрішими, а козак Вус — популярнішим серед народу. Отож він вирішив відремонтувати водопровідну систему країни. Вона знаходиться під землею, і для простоти вона розділена на однакові квадратні фрагменти так, що має вигляд поля $n \times m$. Фрагменти нумеруються зверху вниз від 1 до $n$ та зліва направо від 1 до $m$. Кожен фрагмент може бути або порожнім (позначається символом 0), або в ньому може знаходитись труба одного з 6 типів:




Водопровід називається **замкненим**, якщо для кожної труби кожен її кінець з'єднаний з кінцем якоїсь іншої труби.

Нещодавно вас призначили заступником Козака Вуса з питань водопроводу, тож тепер ви можете повернути будь-яку трубу на кут, кратний 90°. Ваше завдання — за допомогою поворотів зробити цей водопровід замкненим або повідомити, що це неможливо.

## Формат вхідних даних

Перший рядок містить три цілі числа $n$, $m$ та $g$ ($1 \leqslant n, m \leqslant 400$, $0 \leqslant g \leqslant 8$) — довжина та ширина поля, а також номер групи відповідно.

Кожний з наступних $n$ рядків містить по $m$ цілих чисел $a_{i,1}, a_{i,2}, \ldots, a_{i,m}$ ($0 \leqslant a_{i,j} \leqslant 6$) — число, що описує тип труби, розташованої в клітинці з координатами $(i, j)$.

## Формат вихідних даних

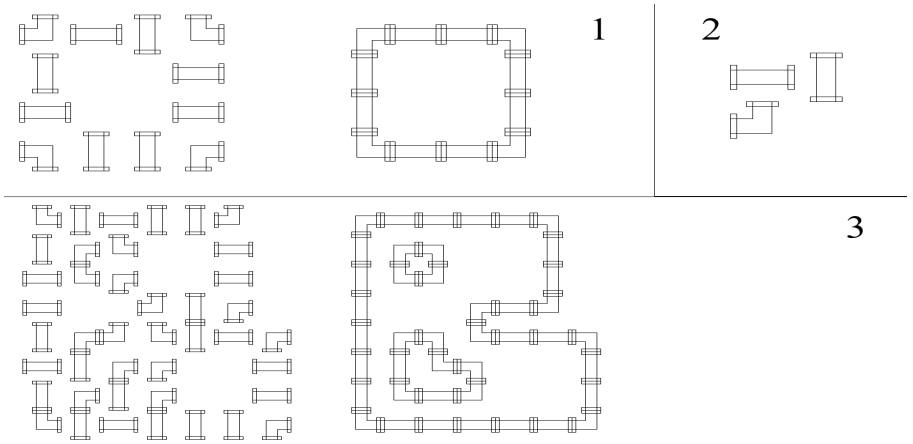
Виведіть `NO`, якщо труби неможливо повернути так, щоб утворити замкнену систему, у протилежному випадку виведіть `YES`, а далі виведіть $n$ рядків по $m$ чисел у кожному — опис замкненої системи, утвореної поворотами, у форматі, описаному вище.

## Приклади

| tubes.in | tubes.out |
|---|---|
| 5 5 0<br>0 0 0 0 0<br>0 3 1 2 4<br>0 2 0 0 1<br>0 1 0 0 1<br>0 6 2 2 5 | YES<br>0 0 0 0 0<br>0 5 1 1 6<br>0 2 0 0 2<br>0 2 0 0 2<br>0 4 1 1 3 |
| 3 2 0<br>0 0<br>1 2<br>3 0 | NO |
| 8 8 0<br>0 4 2 1 2 2 3 0<br>0 2 5 4 0 0 1 0<br>0 1 4 5 0 0 1 0<br>0 1 0 0 3 2 5 0<br>0 2 5 3 4 2 1 5<br>0 1 2 5 5 0 0 1<br>0 2 5 2 5 0 0 1<br>0 4 2 1 2 2 2 5 | YES<br>0 5 1 1 1 1 6 0<br>0 2 5 6 0 0 2 0<br>0 2 4 3 0 0 2 0<br>0 2 0 0 5 1 3 0<br>0 2 5 6 4 1 1 6<br>0 2 2 4 6 0 0 2<br>0 2 4 1 3 0 0 2<br>0 4 1 1 1 1 1 3 |

## Примітка

Ілюстрація до тестових прикладів: початковий та замкнений (якщо він існує) стан:



### Оцінювання

1. (7 балів) $1 \leqslant n, m \leqslant 3$;
2. (9 балів) $1 \leqslant n, m \leqslant 400$; до 4 кутових труб;
3. (12 балів) $1 \leqslant n, m \leqslant 400$; до 8 кутових труб;
4. (13 балів) $1 \leqslant n \leqslant 400$; $m = 2$;
5. (13 балів) $1 \leqslant n, m \leqslant 400$; якщо розв'язок існує, то в хоча б одному з них кожен замкнений контур складається з 4 кутових труб і довільної кількості прямих труб;
6. (14 балів) $1 \leqslant n \leqslant 400$; $m = 4$;
7. (15 балів) $1 \leqslant n, m \leqslant 50$;
8. (17 балів) без додаткових обмежень.

## Задача B. Козак Вус та цукерки

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

Козак Вус дуже любить стрибати, а також любить цукерки. Одного разу його занесло на числову пряму, у певних цілочисельних точках якої лежали цукерки (у кожній точці не більше одної цукерки). Вус нечувано зрадів та вирішив зібрати якомога більше цукерок. Для цього він може вибрати будь-яке ціле число $x > 1$ і початкову позицію $s$ ($s$ — ціле число), після чого він стрибками довжини $x$ побуває у всіх точках вигляду $s + kx$, де $k$ — невід'ємне ціле число. Коли Вус потрапляє в точку, у якій є цукерка, то він підбирає її з землі (хоча так робити не можна!).

Допоможіть Вусу зібрати якомога більшу кількість цукерок.

### Протокол взаємодії

Вам потрібно реалізувати одну функцію:

```
integer solve(integer n, integer g, array of integers m)
```

- $n$ — кількість цукерок;
- $g$ — номер блока;
- $m$ — масив із позицій цукерок;
- ця функція має повертати одне ціле число — максимальну кількість цукерок, яку може зібрати Вус.

### Формат вхідних даних

Перший рядок містить два цілі числа $n$ та $g$ ($1 \leqslant n \leqslant 10^5, 0 \leqslant g \leqslant 6$) — кількість цукерок та номер блоку, відповідно.

Другий рядок містить $n$ цілих чисел $a_1, a_2, \ldots, a_n$ ($1 \leqslant a_i \leqslant 10^9$) — позиції, у яких знаходяться цукерки. Гарантується, що всі $a_i$ попарно різні.

### Формат вихідних даних

У єдиному рядку буде виведена максимальна кількість цукерок, що зможе зібрати Вус.

### Приклади

| Вхідні дані | Вихідні дані |
|---|---|
| 5 0<br>1 2 3 4 7 | 3 |
| 7 0<br>1 2 10 4 7 3 13 | 5 |

### Примітка

У першому прикладі Козак може вибрати $x = 2$ та підібрати цукерки на позиціях $1, 3, 7$.

У другому прикладі Козак може вибрати $x = 3$ та підібрати цукерки на позиціях $1, 4, 7, 10, 13$.

### Оцінювання

1. (4 бали) $n \leqslant 2$; $a_i \leqslant 10$;
2. (5 балів) $n \leqslant 3$; $a_i \leqslant 10^2$;
3. (12 балів) $n \leqslant 10$; $a_i \leqslant 10^2$;
4. (20 балів) $n \leqslant 10^3$; $a_i \leqslant 10^4$;
5. (25 балів) $n \leqslant 10^4$; $a_i \leqslant 10^6$;
6. (34 бали) без додаткових обмежень.

## Задача C. Кольорові книги

| | |
|---|---|
| Назва вхідного файлу: | `permutation.in` |
| Назва вихідного файлу: | `permutation.out` |
| Обмеження використання часу: | 1 секунда |
| Обмеження використання пам'яті: | 256 мегабайтів |

Сьогодні Козак Вус знайшов поличку книг. Усього на поличці було $n$ книг, розташованих у ряд. На кожній книзі було написане одне ціле число $a_i$. Усі ці числа різні з проміжку $[1, n]$. Іншими словами, $a$ — перестановка. Також кожна книга має свій колір, колір $i$-ої книги $c_i$.

Козак хоче навести лад на поличці, для цього потрібно, щоб книги йшли в порядку зростання чисел, записаних на них, тобто книга 1 іде першою, наступна 2 і так далі. За одну операцію він може поміняти місцями будь-які дві книги **різного** кольору. Допоможіть Вусу впоратись з цією задачею за мінімальну кількість операцій. Гарантується, що це можливо зробити.

### Формат вхідних даних

Перший рядок містить одне ціле число $n$ ($1 \leqslant n \leqslant 10^5$) — кількість книг.

Другий рядок містить $n$ цілих чисел $a_1, a_2, \ldots, a_n$ ($1 \leqslant a_i \leqslant n$) — числа, записані на книгах. Гарантується, що всі числа різні.

Третій рядок містить $n$ цілих чисел $c_1, c_2, \ldots, c_n$ ($1 \leqslant c_i \leqslant n$) — кольори книг.

Гарантується, що рішення завжди існує.

### Формат вихідних даних

У першому рядку виведіть одне число $k$ — кількість операцій.

У наступних $k$ рядках виведіть по два цілі числа — позиції книг, які потрібно поміняти місцями.

### Приклади

| permutation.in | permutation.out |
|---|---|
| 7<br>2 5 3 7 1 6 4<br>3 2 2 3 3 2 3 | 5<br>2 4<br>7 4<br>2 7<br>1 2<br>1 5 |
| 2<br>2 1<br>1 2 | 1<br>1 2 |

### Оцінювання

1. (до 50 балів) $n \leqslant 1\,000$; припустимо, що $m$ — мінімальна кількість операцій, які потрібно зробити, щоб відсортувати книги в певному тесті, а $k$ — кількість операцій, які ви використали. Тоді результат за цей тест буде рівний:
   - 50, якщо $k = m$
   - $10 + \lfloor 30 - 30 \cdot \frac{k-m}{3n-m} \rfloor$, якщо $m < k \leqslant 3n$
   - 0, якщо $k > 3n$.

   Ваша кількість балів за цей блок буде рівною мінімуму з результатів у всіх тестах з блоку.
2. (10 балів) $1\,000 < n \leqslant 10^5$, усі кольори різні.
3. (40 балів) без додаткових обмежень.

## Задача D. Пироголяндія

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

Як відомо, Пироголяндія — країна, жителі якої дуже полюбляють пироги. Саме тому в ній побудовано $n$ пекарень, кожна з яких має унікальний цілий номер від 1 до $n$. У кожної пекарні є свій склад, що зберігає певну кількість пирогів, зокрема пекарня з номером $i$ зберігає $a_i$ пирогів на своєму складі.

Також між деякими пекарнями було прокладено дороги, по яких можна перевозити пироги від однієї пекарні до іншої. Усього було прокладено $n-1$ доріг, кожна з яких має свій унікальний цілий номер від 1 до $n-1$. $i$-а дорога була прокладена між пекарнями $u_i$ та $v_i$, тобто можна перевозити будь-яку кількість пирогів зі складу пекарні $u_i$ до складу пекарні $v_i$ та навпаки.

Шляхом між пекарнями $u$ та $v$ будемо називати послідовність **унікальних** номерів пекарень $p_1, p_2, ..., p_k$, таку, що існує дорога між кожною парою пекарень $p_i$ та $p_{i+1}$, для будь-якого $i$ від 1 до $k-1$, а також $p_1 = u$ та $p_k = v$. Відомо, що між кожною парою пекарень існує рівно один шлях, що не відвідує жодну пекарню двічі.

Іноді у Пироголяндії трапляються землетруси, через що деякі дороги можуть ставати перекритими, а тому по них не можна перевозити пироги. Також іноді служба ремонту доріг може ремонтувати деякі з доріг, після чого вони стають знову доступними для перевезень. Отже, кожна дорога має мати один із двох станів — вона має бути або неперекритою, або перекритою.

Компонентою пекарні $u$ будемо називати множину всіх пекарень, до яких є шлях від пекарні з номером $u$, що не проходить через жодну перекриту дорогу. Сама пекарня $u$ також входить до своєї компоненти.

Одним із найвідоміших жителів Пироголяндії є Козак Вус, який отримав свою популярність на тому, що з'їв більше всіх пирогів на щорічному святі пироголюбів.

Іноді у Пироголяндії відбуваються різні події, що пов'язані з пекарнями, дорогами та нашим другом Козаком Вусом. Вам потрібно вміти оброблювати $m$ подій, що будуть подані у вигляді запитів різних типів, та вміти відповідати на деякі з них.

Усього є 7 типів запитів:

1. Змінити стан дороги з номером $p$ на протилежний. Тобто якщо до цього запиту дорога з номером $p$ була перекритою, то після нього вона стає неперекритою, і навпаки, якщо дорога була неперекритою то вона стає перекритою.
2. До складу кожної пекарні, що належить компоненті пекарні із номером $p$, завезли $w$ пирогів. Тобто якщо пекарня з номером $u$ належить компоненті пекарні $p$, то треба збільшити значення $a_u$ на $w$.
3. Усі пекарні з компоненти пекарні з номером $p$ перевозять усі свої пироги до складу пекарні з номером $p$. Тобто після цього запиту на складі пекарні із номером $p$ будуть зберігатись усі пироги з усіх пекарень, що знаходяться в компоненті пекарні $p$, а в усіх інших пекарнях цієї компоненти склади будуть пусті.
4. Козак Вус хоче дізнатись, скільки пирогів зберігається на складі пекарні з номером $p$, тобто значення $a_p$.
5. Козак Вус хоче дізнатися сумарну кількість пирогів, що зберігаються на складах пекарень із компоненти пекарні з номером $p$.
6. Козак Вус з'їдає всі пироги зі складу кожної пекарні з компоненти пекарні з номером $p$. Тобто після цього запиту для кожної пекарні $u$, що належить компоненті пекарні $p$, значення $a_u$ повинно бути рівним 0.
7. Козак Вус хоче дізнатись, яку мінімальну кількість доріг треба відремонтувати, щоб він міг з'їсти всі пироги, що зберігаються на складах пекарень Пироголяндії, якщо він може почати свій шлях із будь-якої пекарні та пересуватись лише по неперекритих дорогах.

Зауважте, що запити типів 4, 5 та 7 ніяк не впливають на пекарні та дороги, вони лиш повинні знаходити відповідь.

### Протокол взаємодії

Вам потрібно реалізувати вісім функцій:

```
void init(integer n, array of integers u, array of integers v, array of integers b,
          array of integers a, integer g)
```

- $n$ — кількість пекарень у Пироголяндії;
- $u_i$ та $v_i$ ($|u| = |v| = n-1$) — номери пекарень, між якими прокладено $i$-ту дорогу;
- $b_i$ ($|b| = n-1$) рівне 1, якщо $i$-та дорога перекрита, та 0 інакше;
- $a_i$ ($|a| = n$) — кількість пирогів в $i$-ій пекарні;
- $g$ — номер блока;
- ця функція викликається першою лише один раз. Вона потрібна для того, щоб повідомити вам кількість пекарень у Пироголяндії, пари пекарень, між якими є дороги, стан кожної дороги, початкову кількість пирогів на складі кожної пекарні та номер блоку. Лише після виклику цієї функції будуть викликатись інші сім.

```
void query_1(integer p)
```

- $p$ — номер дороги, стан якої треба змінити;
- ця функція викликається, коли потрібно задати запит першого типу.

```
void query_2(integer p, integer w)
```

- $p$ — номер пекарні;
- $w$ — кількість пирогів, які завезли до кожної пекарні компоненти пекарні з номером $p$;
- ця функція викликається, коли потрібно задати запит другого типу.

```
void query_3(integer p)
```

- $p$ — номер пекарні;
- ця функція викликається, коли потрібно задати запит третього типу.

```
integer query_4(integer p)
```

- $p$ — номер пекарні;
- ця функція викликається, коли потрібно задати запит четвертого типу;

- функція має повернути ціле число — відповідь на запит.

`integer query_5(integer p)`

- $p$ — номер пекарні;
- ця функція викликається, коли потрібно задати запит п'ятого типу;
- функція має повернути ціле число — відповідь на запит.

`void query_6(integer p)`

- $p$ — номер пекарні;
- ця функція викликається, коли потрібно задати запит шостого типу.

`integer query_7()`

- ця функція викликається, коли потрібно задати запит сьомого типу;
- функція має повернути ціле число — відповідь на запит.

## Формат вхідних даних

Перший рядок містить три цілі числа $n$, $m$ та $g$ ($1 \leqslant n, m \leqslant 250\,000$, $0 \leqslant g \leqslant 11$) — кількість пекарень у Пироголяндії, кількість подій та номер блока відповідно.

$i$-й з наступних $n-1$ рядків містить по два цілі числа $u_i$ та $v_i$ ($1 \leqslant u_i, v_i \leqslant n$) — пара пекарень між якими прокладено дорогу із номером $i$. Гарантується, що між кожною парою пекарень є рівно один шлях.

Наступний рядок містить рядок з $n-1$ символів 0 або 1. $i$-й з цих символів рівний 1, якщо дорога з номером $i$ перекрита, та 0, якщо вона не перекрита.

Наступний рядок містить послідовність з $n$ цілих чисел $a_1, a_2, \ldots, a_n$ ($0 \leqslant a_i \leqslant 10^5$) — кількість пирогів на складі кожної пекарні.

Кожен із наступних $m$ рядків містить опис відповідного запиту. Перше число в рядку задає тип запиту. У випадку, якщо запит 2-го типу, то рядок містить ще два параметри $p$ ($1 \leqslant p \leqslant n$) та $w$ ($0 \leqslant w \leqslant 10^5$). Якщо тип запиту рівний 1, то рядок містить один додатковий параметр $p$ ($1 \leqslant p \leqslant n-1$). Якщо тип запиту рівний від 3 до 6, то рядок містить лише один додатковий параметр $p$ ($1 \leqslant p \leqslant n$). В іншому випадку, якщо запит 7-го типу, рядок не містить додаткових параметрів.

## Формат вихідних даних

На кожний запит типу 4, 5 та 7 виведіть відповідь в окремому рядку.

## Оцінювання

1. (2 балів) $n \leqslant 3\,000$; $m \leqslant 3\,000$; немає запитів типу 1 та 7; усі дороги спочатку перекриті;
2. (3 балів) $n \leqslant 3\,000$; $m \leqslant 3\,000$; немає запитів типу 1 та 7; усі дороги спочатку неперекриті;
3. (5 балів) $n \leqslant 3\,000$; $m \leqslant 3\,000$; немає запитів типу 1 та 7;
4. (6 балів) $n \leqslant 3\,000$; $m \leqslant 3\,000$; немає запитів типу 7;
5. (8 балів) $n \leqslant 3\,000$; $m \leqslant 3\,000$;
6. (10 балів) немає запитів типу 1 та 7;
7. (16 балів) немає запитів типу 7; у запитах типу 1 дороги завжди змінюються лише з перекритих на неперекриті;
8. (15 балів) немає запитів типу 1;
9. (9 балів) немає запитів типу 7;
10. (17 балів) кожна пекарня поєднана дорогою не більше ніж з двома іншими пекарнями;
11. (9 балів) без додаткових обмежень;

## Приклад

| Вхідні дані | Вихідні дані |
|---|---|
| 5 11 0<br>1 2<br>1 3<br>3 4<br>3 5<br>0100<br>1 0 6 1 3<br>5 3<br>3 4<br>2 5 4<br>4 3<br>7<br>6 4<br>1 2<br>5 4<br>2 2 1<br>1 3<br>5 2 | 10<br>4<br>1<br>1<br>5 |

## Примітка

На малюнках пекарні зображені у вигляді кола, всередині якого задано два цілі числа — номер пекарні та кількість пирогів на складі цієї пекарні. Неперекриті дороги між пекарнями зображено суцільною лінією, а перекриті пунктирною.

На малюнку 0 зображено Пироголяндію до всіх змін. Дорога з номером 2 між пекарнями з номерами 1 та 3 перекрита. У компоненті пекарні з номером 1, так само як для пекарні з номером 2, знаходяться пекарні з номерами 1 та 2. У компоненті пекарень із номерами 3, 4 та 5 знаходяться пекарні з номерами 3, 4 та 5, отже, кількість пирогів у пекарнях із компоненти пекарні 3 рівна $6 + 1 + 3 = 10$.

Після другого запиту (Мал. 1) пекарні з номерами 3 та 5 перевозять свої пироги до складу пекарні з номером 4.

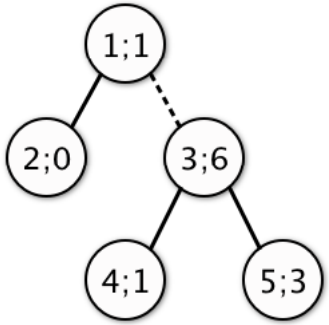
Після третього запиту (Мал. 2) до складу пекарень з номерами 3, 4 та 5 завезли по 4 пироги, отже, кількість пирогів на складі пекарні з номером 3 тепер рівна 4. На складі кожної пекарні, окрім пекарні з номером 2, є хоча б 1 пиріг. Якщо Козак Вус почне свій шлях із пекарні з номером 1 або 2, то йому доведеться відремонтувати дорогу з номером 2, щоб він міг дістатись пекарні з номером 3. Якщо він почне свій шлях з пекарні з номером 3 або 4, або 5, то йому доведеться відремонтувати ту ж саму дорогу з номером 2, щоб дістатись до пекарні з номером 1. Тому відповідь на запит з номером 5 рівна 1.

Після шостого запиту (Мал. 3) Козак Вус з'їдає всі пироги зі складу пекарень з номерами 3, 4 та 5.

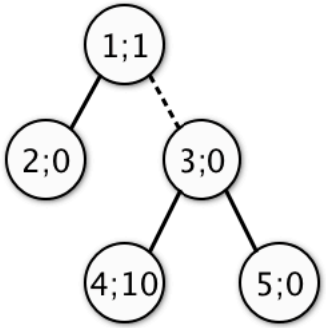
Після сьомого запиту (Мал. 4) служба ремонту доріг відремонтує дорогу з номером 2, тому тепер вона стає неперекритою. Тепер у компоненті пекарні з номером 1, так само як і для всіх інших пекарень, знаходяться всі пекарні Пироголяндії. Тому відповідь на восьмий запит рівна $0 + 1 + 0 + 0 + 0 = 1$.

Після дев'ятого запиту (Мал. 5) до складу кожної пекарні завезли по 1 пирогу.

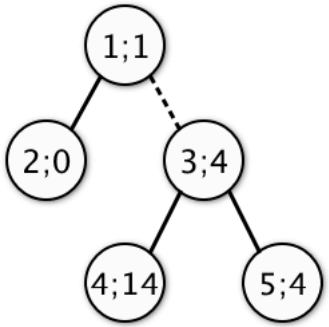
Після десятого запиту (Мал. 6) через землетрус руйнується дорога з номером 3, тому вона стає перекритою. Тепер у компоненті пекарні з номером 1 знаходяться всі пекарні, окрім пекарні з номером 4, а в компоненті пекарні з номером 4 знаходиться лише вона сама. Тому відповідь на останній запит рівна $2 + 1 + 1 + 1 = 5$.



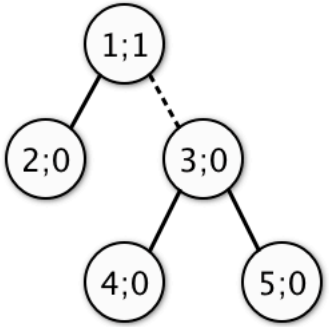

Мал. 0



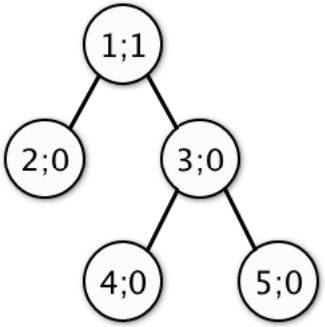

Мал. 1



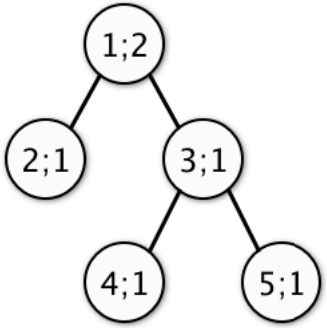

Мал. 2



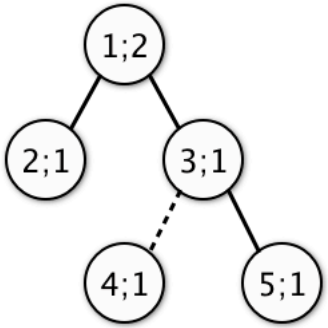

Мал. 3




Мал. 4




Мал. 5




Мал. 6