**Типи даних**

**список**

**стек**

**черга**

**дерево**

<https://www.e-olymp.com/uk/problems/2479>

**Баланс дужок**

Є рядок, що містить дужки ( ) та [ ]. Дужковий вираз вважається правильним, якщо:

• він є порожнім

• якщо A та B правильні, то AB правильно

• якщо A правильно, то (A) та [A] правильні

Напишіть програму, яка за вхідним рядком, що містить дужковий вираз, визначить його правильність. Довжина рядка не більша за 128символів.

Вхідні дані

Перший рядок містить кількість тестів n. Кожний з наступних n рядків містить вираз, який складається із дужок ( ) та [ ].

Вихідні дані

Для кожного тесту вивести в окремому рядку "Yes", якщо вираз є правильним і "No" інакше.

Ліміт часу 1 секунда

Ліміт використання пам'яті 122.49 MiB

Вхідні дані #1

3

([])

(([()])))

([()[]()])()

Вихідні дані #1

Yes

No

Yes

**Бінарне дерево**

**Повний обхід (рекурсивний)**

**Бінарний пошук**

<https://www.e-olymp.com/uk/problems/4117>

**Бінарний пошук**

Следующий фрагмент кода выполняет бинарный поиск целого числа в массиве, который отсортирован в неубывающем порядке:

|  |  |
| --- | --- |
| Pascal | C |
| const  MAXN = 10000;  Var A: array[0..MAXN-1] of integer;  N: integer;  procedure BinarySearch(x: integer);  var  p, q, i, L: integer;  begin  p := 0; { Левая граница поиска }  q := N-1; { Правая граница поиска }  L := 0; { Счетчик сравнений }  while p <= q do begin  i := (p + q) div 2;  inc(L);  if A[i] = x then begin  writeln('Found item i = ', i,  ' in L = ', L, ' comparisons');  Exit end;  if x < A[i] then  q := i - 1  else  p := i + 1  end  end; | #define MAXN 10000  int A[MAXN];  int N;  void BinarySearch(int x)  {  int p, q, i, L;  p = 0; /\* Левая граница поиска \*/  q = N-1; /\* Правая граница поиска \*/  L = 0; /\* Счетчик сравнений \*/  while (p <= q) {  i = (p + q) / 2;  ++L;  if (A[i] == x) {  printf("Found item i = %d"  " in L = %d comparisons", i, L);  return;  }  if (x < A[i])  q = i - 1;  else  p = i + 1;  }  } |

Перед вызовом BinarySearch, N присвоено некоторое число от 1 до 10000 включительно, а массив A содержит целочисленную неубывающую последовательность.

Известно, что процедура остановится с сообщением "Found item i = XXX in L = XXX comparisons" для некоторых найденных значений i и L.

Напишите программу, которая находит все возможные значения N, ведущих к такому сообщению. Однако количество возможных N может быть велико. Поэтому следует сгруппировать в интервалы последовательные N-ки и вывести только первое и последнее значение каждого интервала.

Входные данные

В одной строке содержится для целых числа i и L(0≤ i< 10000, 1≤ L≤ 14).

Выходные данные

В первой строке вывести количество интервалов K всех возможных значений N. В следующих K строках вывести эти интервалы в возрастающем порядке. Каждая строка содержит два числа Ai и Bi (Ai ≤ Bi), задающие начальную и конечную границу интервала.

Если искомых значений N не существует, то вывести одно число 0.

Ліміт часу 2 секунди

Ліміт використання пам'яті 16 MiB

Вхідні дані

10 3

Вихідні дані

4

12 12

17 18

29 30

87 94

https://www.e-olymp.com/uk/problems/1516

Створення двійкового дерева пошуку

БДП (бінарне дерево пошуку) є ефективною структурою для пошуку. В БДП усі елементи лівого піддерева менші, а усі елементи правого піддерева більші за значення кореня. Розглянемо приклад БДП:

Зазвичай БДП будується в результаті послідовної вставки елементів. У такому випадку послідовність вставки елементів впливає на структуру результуючого дерева. Наприклад:

У цій задачі Вам необхідно знайти такий порядок вставки чисел від 1 до n, щоб отримане БДП мало висоту не більшу за h. Висота БДПвизначається наступним чином:

1. Висота БДП, яке не містить жодної вершини, дорівнює 0.

2. Інакше висота БДП дорівнює 1 плюс максимум висот лівого та правого піддерева.

Умові задачі можуть задовольняти декілька послідовностей вставок. У такому випадку слід вивести послідовність, в якій спочатку йдуть менші числа. Наприклад, для n = 4, h = 3 слід вивести послідовність 1 3 2 4, а не 2 1 4 3 чи 3 2 1 4.

Вхідні дані

Кожний тест містить два натуральних числа n (1 ≤ n ≤ 10000) та h (1 ≤ h ≤ 30). Останній тест містить n = 0, h = 0 і не обробляється. На вхід подається не більш ніж 30 тестів.

Вихідні дані

Результат кожного теста слід вивести в окремому рядку. Кожний рядок починається з "Case #: ", де "#" - номер тесту. Далі у цьому ж рядку слід вивести послідовність із n цілих чисел – порядок вершин, в якому вони будуть вставлятися в БДП висоти не більшої за h. У кінці рядка не повинно бути зайвих проміжків. Якщо потрібне дерево побудувати неможливо, то вивести "Impossible." (без подвійних лапок).

Ліміт часу 1 секунда

Ліміт використання пам'яті 64 MiB

Вхідні дані

4 3

4 1

6 3

0 0

Вихідні дані

Case 1: 1 3 2 4

Case 2: Impossible.

Case 3: 3 1 2 5 4 6

**Графи**

**Пошук в глибину**

<https://www.e-olymp.com/uk/problems/122>

**Маршрути в горах**

Гірський туристичний комплекс складається з n турбаз, з’єднаних між собою k гірськими переходами (інші маршрути в горах небезпечні). Кожен перехід між двома базами займає 1 день. Туристична група знаходиться на базі a і збирається потрапити на базу b не більш ніж за d днів. Скільки існує різних таких маршрутів (без циклів) між a і b?

Вхідні дані

В першому рядку через проміжок записані числа n, k, a, b, d (n ≤ 50, d ≤ 10). Кожен з наступних k рядків містить пару чисел, яка описує можливий гірський перехід. Усі числові значення натуральні.

Вихідні дані

Вивести одне число – кількість маршрутів.

Ліміт часу 1 секунда

Ліміт використання пам'яті 64 MiB

Вхідні дані #1

5 8 2 5 3

1 2

1 3

1 5

2 1

2 4

3 4

3 5

4 1

**Пошук в ширину**

[**https://www.e-olymp.com/uk/problems/982**](https://www.e-olymp.com/uk/problems/982)

**Зв`язність**

Перевірити, чи є заданий неорієнтований граф зв'язним, тобто що з довільної вершини можна по ребрам цього графа потрапити у довільну іншу.

Вхідні дані

У першому рядку задано кількість вершин n та кількість ребер m у графі відповідно (1 ≤ n ≤ 100, 1 ≤ m ≤ 10000). Наступні m рядків містять по два числа ui і vi (1 ≤ ui, vi ≤ n); кожен такий рядок означає, що у графі існує ребро між вершинами ui і vi.

Вихідні дані

Виведіть "YES", якщо граф є зв'язним, і "NO" у протилежному випадку.

Ліміт часу 1 секунда

Ліміт використання пам'яті 122.17 MiB

Вхідні дані #1

3 2

1 2

3 2

Вихідні дані #1

YES

Вхідні дані #2

3 1

1 3

Вихідні дані #2

NO

**Жадібні**

[**https://www.e-olymp.com/uk/problems/978**](https://www.e-olymp.com/uk/problems/978)

**Отримай дерево**

Задано зв'язний неорієнтовний граф без петель і кратних ребер. Дозволяється видаляти з нього ребра. Потрібно отримати дерево.

Вхідні дані

Перший рядок містить кількість вершин n (1 ≤ n ≤ 100) та кількість ребер m графу відповідно. Наступні m пар чисел задають ребра графу. Гарантується, що граф зв'язний.

Вихідні дані

Виведіть n - 1 пару чисел - ребра, які увійдуть у дерево. Ребра можна виводити у довільному порядку.

Ліміт часу 1 секунда

Ліміт використання пам'яті 128 MiB

Вхідні дані #1

4 4

1 2

2 3

3 4

4 1

Вихідні дані #1

1 2

2 3

3 4

**Динамічне**

[**https://www.e-olymp.com/uk/problems/976**](https://www.e-olymp.com/uk/problems/976)

**Флойд - існування**

Задано орієнтовний зважений граф. За його матрицею суміжності потрібно для кожної пари вершин визначити, чи існує найкоротший шлях між ними чи ні.

Найкоротший шлях може не існувати з двох причин:

• Немає жодного шляху.

• Є шлях як завгодно маленької ваги.

Вхідні дані

У першому рядку задається кількість вершин графа n (1 ≤ n ≤ 100). У наступних n рядках записано по n чисел, які задають матрицю суміжності графа (j-те число у i-ому рядку відповідає вазі ребра з вершини i у вершину j). В ній число 0 означає відсутність ребра, а довільне інше число - наявність ребра відповідної ваги. Всі числа по модулю не перевищують 100.

Вихідні дані

Виведіть n рядків по n чисел: j-те число у i-ому рядку повинно бути рівним 0, якщо шлях з i у j не існує, 1 - якщо існує найкоротший шлях, і 2- якщо існує шлях як завгодно маленької ваги.

Ліміт часу 1 секунда

Ліміт використання пам'яті 64 MiB

Вхідні дані

5

0 1 2 0 0

1 0 3 0 0

2 3 0 0 0

0 0 0 0 -1

0 0 0 -1 0

Вихідні дані

1 1 1 0 0

1 1 1 0 0

1 1 1 0 0

0 0 0 2 2

0 0 0 2 2