Упражнения: Алгоритми за търсене

Problem 1. Линейно търсене

Реализирайте алгоритъм, който намира индекса на елемент в неподреден масив от цели числа с помощта на линейно търсене. Прочетете поредица от числа на първия ред и едно число на втория ред от конзолата. Намерете индекса на числото в дадения масив. Върнете -1, ако елементът не присъства в масива.

Примери

Вход	Изход
1 2 3 4 5 1	0
1 2 3 4 5 6	-1

Problem 2. Реализиране на двоично търсене

Реализирайте алгоритъм, който намира индекса на елемент в подреден масив от цели числа за логаритмично време

Примери

Вход	Изход	Коментари
1 2 3 4 5 1	0	Индексът на 1 е 0
-1 0 1 2 4 1	2	Индексът на 1 е 2

Подсказки

Първо, ако не сте запознати с принципа му на работа, прочетете за двоичното търсене в Wikipedia.

Тук може да намерите инструмент, който показва нагледно как се осъществява търсенето.

Накратко, ако имаме **сортирана колекция** от сравними елементи, вместо да правим последователно търсене (което отнема линейно нарастващо с броя на елементите време), можем да елиминираме половината от елементите на всяка стъпка и да свършим за логаритмично време.

Двоичното търсене е алгоритъм от типа **разделяй-и-владей**; започваме в средата на колекцията - ако не намерим елемента там, имаме три възможности:

- Елементът, който търсим е по-малък – тогава търсим наляво от текущия елемент, защото знаем, че всички надясно от него са по-големи;
- Елементът, който търсим е по-голям – тогава търсим надясно от текущия елемент;
- Елементът, който търсим не е наличен - в този случай, по традиция, връщаме -1.

Започнете с дефинирането на клас с метод

```
public class BinarySearch
{
    public static int IndexOf(int[] arr, int key)
    {
    }
}
```

Вътре в метода, декларирайте две променливи, определящи границите на търсене и един цикъл while

```
int lo = 0;
int hi = arr.Length - 1;
while (lo <= hi)
{
    // TODO: Find index of key
}

return -1;
```

Вътре в цикъла while трябва да намерим средата на областта, в която търсим числото

```
int mid = lo + (hi - lo) / 2;
```

Ако key в вляво от средата, местим дясната граница. Ако key е надясно от средата, местим лявата граница

```
if (key < arr[mid])
{
    hi = mid - 1;
}
else if (key > arr[mid])
{
    lo = mid + 1;
}
else
{
    return mid;
}
```

Problem 3. Фибоначи търсене

Реализирайте алгоритъм, който намира индекса на елемент в подреден масив от цели числа с помощта на Фибоначи търсене. Прочетете поредица от числа на първия ред и едно число на втория ред от конзолата. Намерете индекса на числото в дадения масив. Върнете -1, ако елементът не присъства в масива.

Примери

Вход	Изход
1 2 3 4 5 1	0
1 2 3 4 5 6	-1

Problem 4. Състезание за търсене

Напишете програма, която **сравнява бързодействието** на различните алгоритми за търсене в три кръга и определя кой е победителят за всеки кръг, както и финалния победител (този с най-малко време общо за трите кръга). Използвайте два или повече от следните алгоритми за търсене:

- Линейно търсене
- Двоично търсене
- Фибоначи търсене
- Търсене чрез интерполация

Във всеки кръг, всеки от алгоритмите за търсене трябва да бъде тестван върху един от следните три масива:

1. Масив от **поредни числа от 1 до N**, подредени в нарастващ ред (т.е. 1, 2, 3, … N)
2. Масив от **поредни числа от N до 1**, подредени в намаляващ ред (т.е. N, N-1, N-2, … 3, 2, 1)
3. Масив от **N случайни числа**, всяко в диапазона **от 1 до N** включително.

Тестването ще става чрез **търсенето на R на брой случайни числа** (всяко в диапазона от 1 до N включително) в съответния масив с помощта на всеки от алгоритмите. Масивът, в който се търсят числата, също е един и същ за всеки от алгоритмите. Засича се **общото време** за откриването на **всичките R** на брой числа. Ако даден алгоритъм се нуждае от сортиран масив за намиране на числата, то това време се добавя **веднъж** към времето за търсене на всички числата за съответния алгоритми. Масивът се сортира само веднъж и се ползва за всички алгоритми. Изберете алгоритъм за сортиране, който смятате, че ще е най-удачен за съответния масив от числа.

Вход

- Входните данни трябва да се прочетат от конзолата.
- На първия и единствен ред ще ви бъдат подадени целите числа N и R, разделени с интервал.
  - N е броят на числата в масива, който ще тествате и максимално допустима стойност за всяко число. Минималната е 1.
  - R е броят на случайните числа, които ще търсите в масива. Всяко трябва да е в диапазона от 1 до N.
- Входните данни винаги ще са валидни и в описания формат. Не е необходимо да бъдат изрично проверявани.

Изход

- Изходните данни трябва да бъдат отпечатани на конзолата.
- На четири реда трябва да бъдат изведени от 2 до 4 числа, разделени с интервал, съобразно броят на реализираните алгоритми. Всяко число представлява времето на съответния алгоритъм за намирането на R случайни числа в:
  - масива от поредни числа от 1 до N
  - масива от поредни числа от N до 1
  - масива от N случайни числа
  - общото време за изпълнението на 3-те горни кръга за съответния алгоритъм.

Подсказки

- Тествайте задачата с различно големи масиви, различни алгоритми за сортиране и различен брой търсени числа.
- Коментирайте в клас получените резултати.

Problem 5. * Игли

В тази задача трябва да намерите вярното място на числата в дадена поредица. От конзолата ще прочетете поредица от ненамаляващи цели числа, между които на случаен принцип има „дупки” (представени чрез нули).

Ще ви бъдат дадени иглите – числа, които трябва да бъдат вмъкнати в поредицата, така че тя да остане ненамаляваща (без да се вземат под внимание „дупките”). За всяка игла намерете най-левия възможен индекс, където може да бъде вмъкната.

Вход

- Входните данни трябва да се прочетат от конзолата.
- На първия ред ще ви бъдат дадени числата C и N, разделени с интервал.
- На втория ред ще ви бъдат дадени C неотрицателни цели числа, формиращи ненамаляваща редица (като не вземаме в предвид нулите).
- На третия ред ще ви бъдат дадени N положителни цели числа, иглите.
- Въведените данни винаги ще са валидни и в описания формат. Не е необходимо да бъдат изрично проверявани.

Изход

- Отговорът трябва да се отпечата на конзолата. Трябва да бъде на един ред.
- На единствения изведен ред отпечатайте N числа, разделени с интервал. Всяко число представлява най-левият възможен индекс, където може да бъде вмъкната съответната игла.

Ограничения

- Всички входни числа ще бъдат 32-битови цели числа със знак.
- N ще бъдат в обхвата [1 … 1000].
- C ще бъдат в обхвата [1 … 50000].
- Допустимо време за работа на вашата програма: 0.1 секунди. Допустима памет: 16 MB.

Примери

Вход
23 9 3 5 11 0 0 0 12 12 0 0 0 12 12 70 71 0 90 123 140 150 166 190 0 5 13 90 1 70 75 7 188 12
Изход
1 13 15 0 13 15 2 21 3
Коментари
5 отива при индекс 1 – между 3 и 5 13 отива при индекс 13 – 12 и 70 90 отива при индекс 15 – между 71 и 0 1 отива при индекс 0 – преди 3 Etc.
Вход
11 4 2 0 0 0 0 0 0 0 0 0 3 4 3 2 1
Изход
11 1 0 0

Министерство на образованието и науката (МОН)

- Настоящият курс (презентации, примери, задачи, упражнения и др.) е разработен за нуждите на Национална програма "**Обучение за ИТ кариера**" на МОН за подготовка по професия "Приложен програмист".

- Курсът е базиран на учебно съдържание и методика, предоставени от **фондация "Софтуерен университет"** и се разпространява под **свободен лиценз CC-BY-NC-SA** (Creative Commons Attribution-Non-Commercial-Share-Alike 4.0 International).