Алгоритми за търсене

Лекция 3.1 по СДА, Софтуерно Инженерство
Зимен семестър 2019-2020г
Милен Чечев

Задачи за търсене

Задача 1: Да се намери дали числото X се среща в масив arr

Задача 2 : Да се намери колко пъти число X се среща в масив.

Задача 3 : Да се намери на кои позиции, се среща числото X в масив.

Линейно търсене (Linear search)

```
boolean linear_search(int[]arr, int x){
    for(int i = 0 ; i < arr.length; i++){
        if(arr[i]==x){
            return true;
        }
    }
    return false;
}
```

Двоично търсене(Binary search)

При несортиран масив няма как да търсим със сложност по-малка от O(n), в случай, че ще извършваме много търсения върху един масив то за да ускорим търсенето можем първоначално да го сортираме (за O(nlog(n))) и после да търсим със по-ниска сложност O(log(n))

Двоично търсене(Binary search)

```
Boolean binary_search(int[] sorted, int x, int start, int end){

        if(start>end) return false;
        middle = (end+start)/2]
        if(arr[(end+start)/2] == x) return true;
        if(arr[(end+start)/2] > x) return binary_search(sorted,x,start,middle-1);
        if(arr[(end+start)/2] < x) return binary_search(sorted,x,middle+1,end);
}
```

Тристранно търсене (Ternary Search)

```
boolean ternarySearch(arr, x, left, right){

        if(right < left) return false;

        mid1 = (2*left + right)/3;
        mid2 = (left + 2*right)/3;
        if(arr[mid1] == x || arr[mid2]==x) return true;

        if(arr[mid1]) > x) return ternary_search(arr, x, left,mid1-1);
        if(arr[mid2]) > x) return ternary_search(arr, x, mid1+1,mid2-1);

        return ternary_search(arr,x, mid2+1,right)
}
```

Задача за самостоятелно решаване

Имаме 2 пластмасови топки и 100 етажна сграда. Искаме да разберем каква е устойчивостта на материала на топките като знаем, че материала при изпускане или се счупва или не понася никакви поражения. С колко най-малко опита може със сигурност да се каже до кой етаж е издръжливостта на такава пластмасова топка?

Търсене със скоци(Jump Search)

```
boolean jumpSearch(int[] arr, int x) {

        int step = (int)Math.floor(Math.sqrt(arr.length));

        int prev = 0; int next = prev + step;
        while (arr[Math.min(next, n)-1] < x) {
                prev = next;
                next += step;
                if (prev >= n)
                return false;
        }
 // continue on next slide
```

Jump Search (продължение)

```
        while (arr[prev] < x) {
                prev++;
                if (prev == Math.min(step, n))
                return false;
        }

        if (arr[prev] == x)
                return true;

        return false;
}
```

Минимум, Максимум, Средна точка

Задача: Намерете най-големият(най-малкият) елемент на масив

Задача: Намерете к-тия най-голям елемент на масив

Задача: Намерете средният елемент на масив.

К-тия най-голям елемент на масив

Подходи:

O(nlog(n)) - сортираме масива и взимаме съответният елемент

Как да се справим по-бързо?

1. Не е необходимо да сортираме масива.
2. Трябва само да знаем кои са елементите по-големи и по-малки от к-тия елемент
3. Може да ползваме подход подобен на quicksort

К-тия най-голям елемент на масив (реализация)

```
randomized_select(arr, left, right, k){
	if(left==right) return arr[left];
	q = randomized_partition(arr,left,right)
	i = q-left+1

	if(i==k) return arr[q]

	if(i < k) return randomized_select(arr,left, q-1,k)

	return randomized_select(arr,q+1,right,k)
}
```