Алгоритми за търсене

Лекция 3 по СДА, Софтуерно Инженерство
Зимен семестър 2021-2022г
Милен Чечев

План за днес

- Организационни въпроси
- Видове търсене
  - Линейно търсене
  - Двоично търсене
  - Тристранно търсене
  - Търсене със скоци
  - Минимум/максимум и средна точка
  - К-тия най-голям елемент
- Почивка: 17:40-17:55
- Контролно 1 - Начало 18:00. Време за работа 60 мин.

Организационни въпроси

Домашна работа 2:

- 95 студента са решили всички задачи
- 120 са решили две или повече задачи

Поздравления! Продължавайте с добрата работа.

Въпрос: Колко време отделихте за домашна работа 2.

A) 1 час B) 2-4 часа C) 5-10 часа D) повече от 10 часа

Организационни въпроси

Преписването на контролни и домашни е забранено и всички сте се съгласили в началото на курса, че ще решавате задачите си самостоятелно!

На домашно 1 са идентифицирани над 10 човека, чиито решения са идентични. Всички те са поставени под допълнително внимание/проверки и след проверките на още няколко работи ще се приложи наказание(анулиране на точките).

Алгоритми за търсене

Лекция 3.1 по СДА, Софтуерно Инженерство
Зимен семестър 2020-2021г
Милен Чечев

Задачи за търсене

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

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

Дали има по-ефективен алгоритъм?

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 общото време за търсене първоначално следва да сортираме масива (за O(nlog(n))) и после да търсим със по-ниска сложност O(log(n))

Пример

Не сортиран масив: (търсим числото 2)

1 6 2 5 3 7 4 9 8

Сортиран масив:

1 2 3 4 5 6 7 8 9

Двоично търсене(Binary search)

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

Тристранно търсене (Ternary Search)

Сортиран масив:

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)
}
```

Търсене със скоци(Jump Search)

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 25 25

Търсене със скоци(Jump Search)

```
int jumpSearch(int arr[], int x, int n){
    int step = sqrt(n);

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

Jump Search (продължение)

```
        while (arr[prev] < x){
                prev++;
                 if (prev == min(step, n))
                     return -1;
        }

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

        return -1;
}
// Каква е сложността на това решение:
 A) O(logN)   B) O(N)  C) O(N^2)  D)Друга?
```

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)
}
```

Задачка за междучасието.

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?

Контролно 1

- Започва точно в 18:00
- Време за работа 60 мин
- 2 задачи по 50 точки всяка

Следващият път: тема: Свързан Списък