САА – Упражнение 5

Алгоритми за търсене на елементи в масив

1. *Търсене на елемент в несортиран масив*

- Алгоритъм с последователно обхождане на всички елементи в масива. Елементите на масива се обхождат последователно, докато се достигне края на масива или се открие търсения елемент. В най-тежкия случай този алгоритъм ще използва 2n сравнения, където n е броят на елементите в масива. На всяка стъпка ще се правят 2 проверки:
    1. за достигане на край на масива;
    2. за откриване на търсения елемент.

- Алгоритъм с добавяне на нов елемент. В края на масива се добавя нов елемент със стойност, равна на търсената. По този начин се избягва проверка 1 от предишния алгоритъм и така броя на сравненията намалява до n+1.

2. *Търсене на елемент в сортиран масив*

- Алгоритъм с последователно обхождане. Елементите на сортирания масив се обхождат последователно, докато се достигне края на масива или елемент, който е по-голям от търсения. В най-тежкия случай този алгоритъм ще използва 2n сравнения.

- Двоично търсене. Нека разглеждания масив има начален индекс l и краен индекс r. От масива се избира елемента със среден индекс – т. е. $(l+r)/2$ и се сравнява с търсения елемент. Ако съвпадат – търсеният елемент е намерен. В случай, че не съвпадат са възможни два варианта:
    1. Търсеният елемент е по-малък от елемента със среден индекс – тогава търсенето продължава в лявата част с двойно по-малък размер: начало l и край $(l+r)/2$ - 1.
    2. Търсеният елемент е по-голям от елемента със среден индекс – тогава търсенето продължава в дясната част с двойно по-малък размер: начало $(l+r)/2$ + 1 и край r.

  Сложността на алгоритъма за двоично търсене е $O(\log_2 n)$. Това е най-бързият алгоритъм за търсене в сортиран масив.

  Програмна реализация на функцията за двоично търсене:

```
int BinSearch(int l, int r, t_element x, t_element A[])
{
        int m;
        while(l<=r)
        {
```

```
                m=(l+r)/2;
                if (x<A[m])
                        r=m-1;
                else
                        if (x>A[m])
                                l=m+1;
                        else
                                return m;
        }
        return -1;
}
```

<u>Задачи:</u>

1. Реализирайте програмно като отделни функции всеки от алгоритмите за търсене (нерекурсивни).
2. Подредете алгоритмите по възходящ ред според тяхната сложност.
3. Докажете сложността на алгоритъма за двоично търсене.