

Lab10: Divide et Impera

Formatori:

Tutor: [Stiegelbauer Paul](#)  

Tutor: [Iovanovici Alexandru](#)  

+1

Data de începere a cursului:

 19.02.2024

 [Utilizatori înscriși](#)

 [Calendar](#)

 [Note](#)

 [Cursurile mele](#) ▶ [S2-L-AC-CTIRO1-1C-TP](#) ▶ Saptamana 10 Del ▶ [Lab10: Divide et Impera](#)

Lab10: Divide et Impera

1. Algoritmul de cautare binara a unui element intr-un tablou ordonat.

a) Implementati o functie recursiva cu antetul `int cautaBin(int x, int v[], unsigned stg, unsigned dr)`; care cauta elementul `x` in tabloul `v` ordonat (nu conteaza daca crescator sau descrescator) si returneaza pozitia pe care sa gaseste una dintre aparitiile lui `x` sau -1 daca `x` nu apare in tablou.

Indicatie: algoritmul a fost prezentat in pseudocod in cadrul sesiunii de curs din saptamana 7;

b) Implementati functia de mai sus, in maniera nerecursiva (iterativa)

Indicatie: puteti folosi descrierea in pseudocod de aici <https://www.pbinfo.ro/articole/3633/cautarea-binara>

c) masurati timpul de executie (sau alternativ numarul de apeluri ale functiei recursive) pe masura ce dimensiunea tabloului creste; estimati complexitatea

2. Algoritmul de sortare prin interclasare

Implementati o functie cu antetul `void mergeSort(int v[], unsigned stg, unsigned dr)`; care implementeaza algoritmul de sortare prin interclasare, prezentat in cadrul cursului din saptamana 7. Alternativ va puteti folosi de modelul de implementare disponibil la <https://www.pbinfo.ro/articole/7667/sortarea-prin-interclasare>.

Determinati experimental dependenta timpului de executie de dimensiunea tabloului si incercati sa deduceti un ordin de complexitate. Comparati cu valoarea de referinta din literatura, $O(n \cdot \log(n))$ [cf. Knuth]

3. Algoritmul se sortare quick-sort (QS)

Implementati o functie cu antetul `void qs(int v[], unsigned stg, unsigned dr)`; care sorteaza crescator un tablou `v` cu elemente intregi folosind algoritmul de sortare numit quick-sort, descris conform pseudocodului de mai jos.

a) analizati (pe hartie) functionarea algoritmului pentru un set de date de test (ex. {10, 80, 30, 90, 40, 50, 70});

b) masurati timpul de executie al algortmului in implementarea de mai jos si comparati cu timpul de executie al implementarii oferite de functia `qsort(...)` din `stdlib.h`

```
//Algoritmul QS
```

```
quickSort(arr[], stg, dr)
{
    if (stg < dr)
    {
        //pi este indexul (pozitia) pivotului/partitiei
        pi = partition(arr, stg, dr);
        quickSort(arr, stg, pi - 1); // pana la pozitia indexului
        quickSort(arr, pi + 1, dr); // dupa pozitia indexului
    }
}
```

```
//functia de determinare a pozitiei pivotului (preluat din CLRS)
```

```
/* This function takes last element as pivot, places
the pivot element at its correct position in sorted
array, and places all smaller (smaller than pivot)
to left of pivot and all greater elements to right
of pivot */
partition (arr[], stg, dr)
{
    // pivot (Element to be placed at right position)
    pivot = arr[dr];

    i = (stg - 1) // Index of smaller element and indicates the
                // right position of pivot found so far


    for (j = stg; j <= dr - 1; j++)
    {
        // If current element is smaller than the pivot
        if (arr[j] < pivot)
        {
            i++; // increment index of smaller element
            swap arr[i] and arr[j]
        }
    }
    swap arr[i + 1] and arr[dr])
    return (i + 1)
}
```

4. Se citesc doua numere naturale, **n** si **m**, urmate de un sir de **n** numere naturale, ordonate crescator si urmate de **m** perechi de elemente naturale, **x** si **y**. Sa se determine cate dintre intervalele **[x, y]** contin cel putin un element din sir.

◀ Merge sort si altele

Sari la...

Divide et Impera ►

Sunteți conectat în calitate de 
S2-L-AC-CTIRO1-1C-TP

Meniul meu

Profil

Preferinte

Calendar

 ZOOM

Română (ro)

English (en)

Română (ro)

Rezumatul păstrării datelor

Politici utilizare site