11 Laborator: Divide et Impera

11.1 Obiective

Scopul acestui laborator este de a prezenta aplicativ o clasa de algoritmi numita *divide et impera (imparte si cucereste)*.

11.2 Noțiuni teoretice

Metoda *divide et impera* consta in impartirea repetata a unei probleme in doua sau mai multe sub-probleme de acelasi tip si apoi combinarea sub-problemelor rezolvate, in final obtinandu-se solutia problemei initiale.

Fie vectorul $A = (a_1, a_2, ..., a_n)$ ale carui elemente se doreste a fi procesate. Metoda *divide et impera* este aplicabila daca pentru orice numere naturale p si q avem $1 \leq p < q \leq n$ si $\exists m \in [p+1, q-1]$ astfel incat prelucrarea secventei $a_p, a_{p+1}, ..., a_q$ se poate face prelucrand secventele $a_p, a_{p+1}, ..., a_m$ si $a_{m+1}, a_{m+2} ..., a_q$, și apoi prin combinarea rezultatelor se obține prelucrarea dorita.

Metoda *divide et impera* poate fi descrisa in pseudocod astfel:

```
DivideAndConquer(a, p, q)
    a = sequence
    p, q = indices to be processed
    {
        if (|q - p|) ≤ ε)
                return Process(a, p, q)
        m = Divide(a, p, q)
        s_{p,m} = DivideAndConquer(a, p, m)
        s_{m+1,q} = DivideAndConquer(a, m + 1, q)
        s = Combine(s_{p,m}, s_{m+1,q})
        return s
    }
```

11.2.1 Cautarea binara

Considerand un sir ordonat de elemente $A = (a_1, a_2, ...a_n)$, se doreste a stii daca un element k se afla in sirul A. In acest caz problema poate fi impartita in doua sub probleme mai mici datorita relatiei de ordine dintre elemente. Procedura poate fi sintetizata prin urmatorii pasi si exemplificata in cadrul Fig. 11.1.

1. $p = 1$ si $q = n$
2. daca $p > q \implies k$ **nu este prezent**
3. se identifica mijlocul intervalului (p, q), $m = \left\lfloor \frac{p+q}{2} \right\rfloor$
4. daca $a_m = k \implies k$ **este prezent**
5. daca $a_m > k \implies$ **aplica pasul 2 pentru sub-sirul** $A_{p,m}$
6. daca $a_m < k \implies$ **aplica pasul 2 pentru sub-sirul** $A_{m+1,q}$

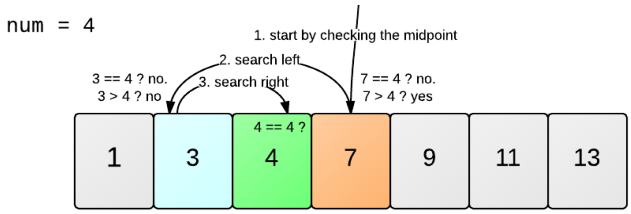

Figure 11.1: Exemplu cautare binara

11.2.2 Sortarea prin interclasare

Fie un sir $A = (a_1, a_2, ...a_n)$, indicii p, q cu $1 \leq p < q \leq n$ si $A_{p,q}$ sub-sirul $(a_p, ..., a_q)$. Sortarea prin interclasare presupune urmatorii pasi:

1. $p = 1$ si $q = n$
2. daca $p + 1 = q \implies A_{p,q}$ **ordonat**
3. se identifica mijlocul intervalului (p, q), $m = \left\lfloor \frac{p+q}{2} \right\rfloor$
4. **se aplica pasul 2 pentru sub-sirul** $A_{p,m} \implies A_{pm}$ **ordonat**
5. **se aplica pasul 2 pentru sub-sirul** $A_{m+1,q} \implies A_{m+1,q}$ **ordonat**
6. se interclaseaza $A_{p,m}$ si $A_{m+1,q} \implies A_{p,q}$ **ordonat**

In cadrul figurii 11.2 este ilustrat un exemplu de trasare pentru algoritmul de sortare prin interclasare.

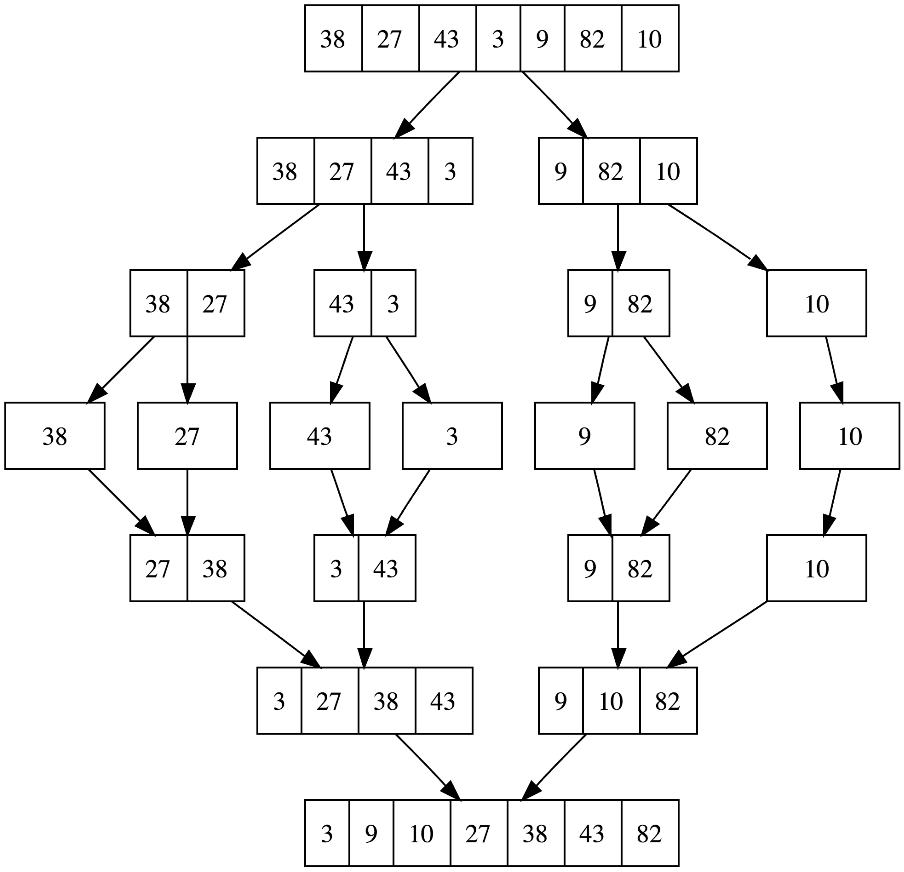

Figure 11.2: Exemplu sortare prin interclasare

Pentru interclasarea a doua siruri o metoda simpla presupune alocarea unui nou sir $A_{p,q}^{sortat}$ de fiecare data cand are loc operatia. Pentru a optimiza utilizarea memoriei se poate utiliza un singur sir de dimensiune n, reutilizat de fiecare data cand are loc interclasarea, acesta putand fi returnat la ultima interclasare ca si rezultat final al ordonarii.

11.2.3 Înmulțirea Karatsuba

Inmultirea Karatsuba este o procedura prin care doua numere cu n cifre pot fi inmultite cu o complexitate de $\Theta(n^{log_2 3})$, in loc de $\Theta(n^2)$ care este complexitatea metodei *clasice* pentru inmultirea a doua numere.

Fie x si y doua numere in baza 10 de n cifre. Pentru $\forall m, 1 \leq m \leq n$, cele doua numere pot fi reprezentate ca:

$$x = x_1 10^m + x_0$$
$$y = y_1 10^m + y_0$$

In cazul acesta produsul poate fi reprezentat in felul urmator:

$$xy = (x_1 10^m + x_0)(y_1 10^m + y_0)$$
$$xy = z_2 10^{2m} + z_1 10^m + z_0$$

Unde:

$$z_0 = x_0 y_0$$
$$z_1 = x_1 y_0 + x_0 y_1$$
$$z_2 = x_1 y_1$$

Iar pentru eficienta cele patru multiplicari pot fi reduse la trei prin rescrierea termenului z_1:

$$z_1 = (x_1 + x_0)(y_1 + y_0) - z_2 - z_0$$

Din descompunerea anteriora se poate dezvolta urmatoarea procedura de tipul *divide et impera* pentru produsul a doua numere:

```
Karatsuba(x,y)
    x,y = two numbers
    {
        if (x < 10 or y < 10)
                return x * y

        m = max(size(x), size(y)) / 2
        x1,x0 = split(x,m)
        y1,y0 = split(y,m)

        z0 = Karatsuba(x0,y0)
        z2 = Karatsuba(x1,y1)
        z1 = Karatsuba(x1 + x0, y1 + y0) - z2 - z1
        return z2 10^(2m) + z1 10^m + z0
    }
```

11.3 Mersul lucrării

11.3.1 Probleme obligatorii

1. Implementati si testati algoritmul de cautare binara. Modificati solutia astfel incat sa se returneze pozitia unde elementul cautat a fost identificat in sir sau pozitia unde acesta ar trebui inserat (si nu *negăsit* pentru cazul in care nu este in sir).

2. Implementati si testati algoritmul de sortare prin interclasare.

3. Implementati si testati algoritmul de inmultire Karatsuba.

11.3.2 Probleme opționale

1. Implementati si testati metoda de sortare *quicksort* (pseudocod in cursul 10).

2. Se da un sir de siruri de caracter, folosind tehnica *didive et impera* implementati si testati un algoritm care identifica cel mai lung prefix comun. Exemplu:

   Intrare : “gigel”, “gin”, “gir”, “giratoriu”

   Iesire : "gi"

   Intrare : "apple", "ape", "april"

   Iesire : "ap"

3. Sa scrie o functie care construieste un arbore binar de cautare perfect echilibrat, dandu-se cheile intr-un vector sortat.

11.3.3 Probleme extra-credit

1. Se da un sir de coordate: $(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)$. Sa se identifice cele mai apropiate doua puncte, considerand distanta Euclidiana.