Metode de rezolvare a recurențelor

October 31, 2017

1 Sortarea prin interclasare (MergeSort)

Algorithm 1 Pseudocod mergesort

```
function MERGE(V1[1..n], V2[1..m])
    Let R[1..n + m]
    i = 0
    j = 0
    while i < n and j < m do
        if V1[i] < V2[j] then R[i + j] = V1[i + +]
        elseR[i + j] = V2[j + +]
        end if
    end while
    while i < n do
        R[i + j] = V1[i + +]
    end while
    while j < m do
        R[i + j] = V2[j + +]
    end while
end function

function MERGESORT(V[1..n], start, end)
    if start ≥ end then return
    end if
    middle = (start + end)/2
    MergeSort(V, start, middle)
    MergeSort(V, middle, end)
    return Merge(V[start..middle], V[middle..end])
end function
```

Sortarea prin interclasare este un algoritm de tip Divide et Impera și funcționează după următoarea schemă:
1. Imparte lista inițială (de dimensiune n) in n subliste de dimensiune 1 (pentru care sortarea este trivială).

2. Aplică funcția merge() in mod repetat pentru a obține noi subliste sortate, de dimensiune dublă. In final se obține lista inițială sortată.

Recurența pentru MergeSort se poate rezolva ușor dacă se observă că problema inițală $T(n)$ se imparte in 2 subprobleme $T(n/2)$, la a căror complexitate se adună cea pentru merge(). Se observă cu ușurință că funcția merge() are complexitatea $\Theta(n)$ (deoarece parcurge elementele celor doi vectori o singură dată). Rezultă că MergeSort are recurența de forma:

$$T(n) = 2T(n/2) + \Theta(n)$$

Folosind metoda iterației sau cea a arborilor de recurență, rezultă $T(n) \in \Theta(nlog(n))$.

2 Metoda substituției

2.1 Prezentare

Metoda substituției se folosește atunci cand este posibilă intuirea complexității unei probleme care are atașată o relație de recurență. După ce se găsește soluția, aceasta trebuie verificată folosind inducția matematică. Schema pentru metoda substituției este:

- alegem soluția:
  - știm complexitatea pentru o recurență asemănătoare
  - stabilim limite inferioare și superioare din ce in ce mai apropiate de soluție
  - se folosesc celelalte metode studiate
- verificăm apartenența lui $T(n)$ la o anumită clasă de complexitate prin inducție:
  - caz de bază
  - pas de inducție de tipul $P(n) \Rightarrow P(n+1)$ sau $P(n/k) \Rightarrow P(n)$

2.2 Aplicarea metodei pentru MergeSort

Ne propunem să oferim o demonstrație formală a faptului că algoritmul MergeSort are complexitatea $\Theta(nlog(n))$.
Fie $T(n) = 2T(n/2) + \Theta(n)$ și ceea ce trebuie demonstrat $P(n) : T(n) \in \Theta(nlog(n))$

$$P(n) : \exists c_1, c_2 \in R_+^*, n_0 \in N, c_1 nlog_2 n \leq T(n) \leq c_2 nlog_2 n, \forall n \geq n_0$$

1. Caz de bază
$n = 0$ sau $n = 1$ nu sunt cazuri valide (primul ar duce la $log_2 0$, iar din al doilea ar rezulta că $T(1) = 0$), de aceea pornim de la $n = 2$.

$$P(2) : T(2) \in \Theta(2log_2 2) \Rightarrow 2c_1 \leq T(2) \leq 2c_2 \Rightarrow 2c_1 \leq 2T(1) + 2c \leq 2c_2$$

$$\Rightarrow c_1 \leq T(1) + c \leq c_2, T(1) = 1 \Rightarrow c_1 \leq c + 1 \leq c_2$$

Putem alege $c_1 = 1/2$, $c = 1$, $c_2 = 3$, iar inegalitatea este satisfăcută. Astfel, $P(2)$ este adevărată.

2. Pasul de inducție: $P(n/2) \Rightarrow P(n)$
Presupunem că $P(n/2) : T(n/2) \in \Theta(n/2log_2 n/2)$ este adevărată $\forall n \geq 2$

$$\Rightarrow \exists c_1, c_2 \in R_+^*, n_0 \in N, c_1 \frac{n}{2} log_2 \frac{n}{2} \leq T(\frac{n}{2}) \leq c_2 \frac{n}{2} log_2 \frac{n}{2}, \forall n \geq n_0$$

Inmulțim inegalitatea cu 2 si adunăm $c * n$

$$\Rightarrow 2c_1 \frac{n}{2} log_2 \frac{n}{2} + cn \leq 2T(\frac{n}{2}) + cn \leq 2c_2 \frac{n}{2} log_2 \frac{n}{2} + cn$$

$$\Rightarrow c_1 n log_2 \frac{n}{2} + cn \leq T(n) \leq c_2 n log_2 \frac{n}{2} + cn$$

$$\Rightarrow c_1 n(log_2 n - 1) + cn \leq T(n) \leq c_2 n(log_2 n - 1) + cn$$

$$\Rightarrow c_1 n log_2 n - c_1 n + cn \leq T(n) \leq c_2 n log_2 n - c_2 n + cn$$

$$\Rightarrow c_1 n log_2 n \leq c_1 n log_2 n + n(c - c_1) \leq T(n) \leq c_2 n log_2 n + n(c - c_2) \leq c_2 n log_2 n$$

Pentru $c - c_1 \geq 0$ și $c - c_2 \leq 0$, inegalitatea de mai sus este adevărată. Cum putem alege c_1 arbitrar oricat de mic și c_2 arbitrar oricat de mare $\Rightarrow P(n)$ adevărată. Conform ipotezei de inducție, rezultă că $P(n)$ este adevărată $\forall n \geq 2$.

3 Teorema Master

Teorema Master este o metodă generală de rezolvare a anumitor tipuri de recurențe. Putem afirma că teorema Master ne oferă o "rețetă" de rezolvare a recurențelor, insă dezavantajul constă in faptul că ea nu este aplicabilă in orice situație.

3.1 Enunțul teoremei

Fie o recurență de forma $T(n) = aT(n/b) + f(n)$, unde $a \geq 1, b \geq 1$ constante și $f(n)$ o funcție asimptotic crescătoare pozitivă. Putem afirma că T(n) aparține unei anumite clase de complexitate, in următoarele cazuri:

1. Dacă $f(n) \in O(n^{log_b a - \epsilon})$, pentru o constantă $\epsilon > 0$, atunci $T(n) \in \Theta(n^{log_b a})$.

2. Dacă $f(n) \in \Theta(n^{log_b a} log_2^k n)$, atunci $T(n) \in \Theta(n^{log_b a} log_2^{k+1} n)$.

3. Dacă $f(n) \in \Omega(n^{log_b a + \epsilon})$, pentru o constantă $\epsilon > 0$ și $af(n/b) \leq cf(n)$ pentru o constantă $c < 1$ și pentru un n suficient de mare, atunci $T(n) \in \Theta(f(n))$.

Pentru a ne da seama in ce caz ne aflăm, este recomandat să comparăm $f(n)$ cu $n^{log_b a}$:

- dacă sunt egale $\Rightarrow$ caz 2
- dacă $f(n)$ este polinomial mai mare decat $n^{log_b a}$ $\Rightarrow$ caz 3
- dacă $f(n)$ este polinomial mai mic decat $n^{log_b a}$ $\Rightarrow$ caz 1

3.2 Math cheatsheet

- $\sum_{k=1}^{n} k = \frac{n(n+1)}{2} \approx \frac{n^2}{2}$
- $\sum_{k=1}^{n} k^2 = \frac{n(n+1)(2n+1)}{6} \approx \frac{n^3}{3}$
- $\sum_{k=1}^{n} k^3 = \frac{n^2(n+1)^2}{4} \approx \frac{n^4}{4}$
- $\sum_{k=1}^{n} k^\alpha \approx \frac{n^{\alpha+1}}{\alpha+1}$
- $x_{n+1} = qx_n \Rightarrow \sum_{k=1}^{n} x_k = x_0 \frac{q^{n+1}-1}{q-1}$

4 Exerciții

1. Arătați că $T(n) = 2T(n/2) + \Theta(n) \in \Theta(n + nlog_2n)$.

2. Rezolvați următoarele recurențe:

- **(a)** $T(n) = 3T(2n/3) + n^3log_2n$
- **(b)** $T(n) = T(n/4) + 1$
- **(c)** $T(n) = 4T(n/2) + n\sqrt{n}$
- **(d)** $T(n) = 16T(n/4) + n!$
- **(e)** $T(n) = 8T(n/3) + 2^n$
- **(f)** $T(n) = T(n/4) + lgn$
- **(g)** $T(n) = 4T(n/2) + \frac{n}{lgn}$
- **(h)** $T(n) = 3T(n/3) + nlgn$
- **(i)** $T(n) = T(n/2 - log_2n) + 1$

3. Demonstrați că $T(n) = 5T(n/4) + n \in \Theta(5n^{log_4 5} - 4n)$

4. Studiați posibilitatea aplicării teoremei Master in următoarele cazuri:

- **(a)** $T(n) = 2T(n/2) + nlog_2n$
- **(b)** $T(n) = 3T(n/3) + \frac{n}{log_2n}$

5. Rezolvați următoarele recurențe:

- **(a)** $T(n) = 2T(n/2) + 1$
- **(b)** $T(n) = 3T(n/3) + n/2$
- **(c)** $T(n) = 3T(n/3) + nlog_2^2n$
- **(d)** $T(n) = 3T(n/4) + nlog_2n$
- **(e)** $T(n) = 5T(n/4) + n$