Αλγόριθμοι και Πολυπλοκότητα

7ο εξάμηνο

Σ.Η.Μ.Μ.Υ. & Σ.Ε.Μ.Φ.Ε.

http://www.corelab.ece.ntua.gr/courses/

3η εβδομάδα: *Divide and Conquer, Αλγόριθμοι Ταξινόμησης*

Διδάσκοντες:

Στάθης Ζάχος – Άρης Παγουρτζής

Divide and Conquer

```
function divconq (a:array[p..q] of item): info;
  var m: index;
begin
  if small(p..q) then divconq := g(a[p..q])
  else
  begin
    m := PartitionPoint(p,q);
    divconq := Combine(divconq(a[p..m]), divconq(a[m+1..q]))
  end
end
```

Αναδρομικές Συναρτήσεις Πολυπλοκότητας

$$T(n) = \begin{cases} g(n_0) & \text{για } n \leq n_0 \\ 2T(n/2) + f(n) & \text{αλλιώς} \end{cases}$$

όπου $f(n)$ είναι η πολυπλοκότητα των PartitionPoint και Combine.

Επίλυση Αναδρομής

$$
\begin{aligned}
T(n) &= 2T(\frac{n}{2}) + f(n) = \\
&= 2(2T(\frac{n}{4}) + f(\frac{n}{2})) + f(n) = \\
&= 4T(\frac{n}{4}) + 2f(\frac{n}{2}) + f(n) = \\
&= 4(2T(\frac{n}{8}) + f(\frac{n}{4})) + 2f(\frac{n}{2}) + f(n) = \\
&= 8T(\frac{n}{8}) + 4f(\frac{n}{4}) + 2f(\frac{n}{2}) + f(n) = \\
&= \ldots = \\
&= 2^i T(\frac{n}{2^i}) + 2^{i-1} f(\frac{n}{2^{i-1}}) + \cdots + 2^0 f(\frac{n}{2^0})
\end{aligned}
$$

Αν $n = 2^k$, τότε όταν $i = k$ έχουμε:

$$
T(n) = ng(1) + \frac{n}{2}f(2) + \frac{n}{4}f(4) + \cdots + \frac{n}{2^j}f(2^j) + \cdots + 2f(\frac{n}{2}) + f(n)
$$

Παραδείγματα

$$T(n) = \begin{cases} a & \text{για } n = 1 \\ 2T(n/2) + cn & \text{για } n > 1 \end{cases} \quad \Rightarrow$$

$$T(n) = n \cdot a + \sum (\frac{n}{2^j}) \cdot c \cdot 2^j = n \cdot a + cn \log_2 n = O(n \log n)$$

$$T(n) = \begin{cases} a & , \text{για } n = 1 \\ 2T(n/2) + c & , \text{για } n > 1 \end{cases} \Rightarrow T(n) = n \cdot a + c \cdot (n-1) = O(n)$$

Master Theorem (Απλό)

Αν ισχύει $T(1) = d$ και $T(n) = aT(\frac{n}{b}) + cn$ τότε:

$$T(n) = \begin{cases} O(n) & , \text{για } a < b \\ O(n \log n) & , \text{για } a = b \\ O(n^{\log_b a}) & , \text{για } a > b \end{cases}$$

Δυαδική Αναζήτηση

```
function BinSearch (a:array[p..q] of item; search: item): info;
  var first, last, mid: index; found: boolean;
begin
  first:=p; last:=q; found:=(search=a[first]) or (search=a[last]);
  while not found and (first<last) do
  begin
    mid := (first+last) div 2;
    if search < a[mid] then begin last:=mid−1; first:=first+1 end
                       else begin first:=mid; last:=last−1 end
    found:=(search = a[first]) or (search = a[last])
  end
  if found then if search=a[first] then binsearch:=first else binsearch:=last
           else binsearch := 'not found'
end
```

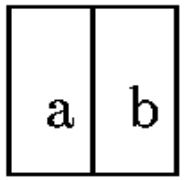
Πολυπλοκότητα Δυαδ. Αναζήτησης

$$T(n) = \begin{cases} a & , \text{για } n = 1 \\ T(\frac{n}{2}) + c & , \text{για } n > 1 \end{cases}$$

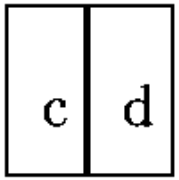
$$T(n) = T(\frac{n}{2}) + c = T(\frac{n}{4}) + 2c = T(\frac{n}{8}) + 3c = \ldots = T(\frac{n}{2^k}) + kc$$

$$T(n) = O(\log n)$$

Πολλ/σμός Ακεραίων

X : 

$= a \cdot 2^{\frac{n}{2}} + b$

Y : 

$= c \cdot 2^{\frac{n}{2}} + d$

$$X\ Y = ac \cdot 2^n + (ad + bc) \cdot 2^{\frac{n}{2}} + bd$$

Πολυπλοκότητα Πολλ/σμού Ακεραίων

$$T(n) = \begin{cases} a & , \text{για } n = 1 \\ 4T(\frac{n}{2}) + cn & , \text{για } n > 1 \end{cases}$$

$$T(n) = (c+1)n^2 - cn = O(n^2)$$

Βελτιωμένος Πολλ/σμός Ακεραίων

$$(ad + bc) \;=\; [(a - b)(d - c) + ac + bd]$$

$$X\ Y = ac \cdot 2^n + [(a - b)(d - c) + ac + bd]\ 2^{\frac{n}{2}} + bd$$

Πολυπλοκότητα Βελτίωσης

$$T(n) = \begin{cases} a & , \text{για } n = 1 \\ 3T(\frac{n}{2}) + cn & , \text{για } n > 1 \end{cases}$$

$$T(n) = O(n^{\log_2 3}) = O(n^{1.59})$$

Πολλ/σμός Πινάκων

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}$$

$$T(n) = O(n^3)$$

Με Πολλ/σμό Υποπινάκων

$$\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix} = \begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix}$$

$$\begin{aligned} C_{11} &= A_{11}B_{11} + A_{12}B_{21} \\ C_{12} &= A_{11}B_{12} + A_{12}B_{22} \\ C_{21} &= A_{21}B_{11} + A_{22}B_{21} \\ C_{22} &= A_{21}B_{12} + A_{22}B_{22} \end{aligned}$$

$$T(n) = 8T(\frac{n}{2}) + 4(\frac{n}{2})^2 = O(n^3)$$

Αλγόριθμος Strassen (1969)

7 πολλαπλασιασμοί υποπινάκων αρκούν!

$$
\begin{aligned}
P &= (A_{11} + A_{22})(B_{11} + B_{22}) \\
Q &= (A_{21} + A_{22})B_{11} \\
R &= A_{11}(B_{12} - B_{22}) \\
S &= A_{22}(-B_{11} + B_{12}) \\
T &= (A_{11} + A_{12})B_{22} \\
U &= (A_{21} - A_{11})(B_{11} + B_{12}) \\
V &= (A_{12} - A_{22})(B_{21} + B_{22})
\end{aligned}
$$

$$
\begin{aligned}
C_{11} &= P + S - T + V \\
C_{12} &= R + T \\
C_{21} &= Q + S \\
C_{22} &= P + R - Q + U
\end{aligned}
$$

$$T(n) = 7T(\frac{n}{2}) + 18(\frac{n}{2})^2 \leq 43n^{\log 7} \qquad T(n) = O(n^{\log 7}) \approx O(n^{2.81})$$

Αντιστροφή Πινάκων

Πολυπλοκότητα ίδια με τον πολλ/σμό (Schur)

1. $$\begin{pmatrix} A & B \\ C & D \end{pmatrix} = \begin{pmatrix} I & 0 \\ CA^{-1} & I \end{pmatrix} \cdot \begin{pmatrix} A & 0 \\ 0 & X \end{pmatrix} \cdot \begin{pmatrix} I & A^{-1}B \\ 0 & I \end{pmatrix}$$

2. $$\begin{pmatrix} A & B \\ C & D \end{pmatrix}^{-1} = \begin{pmatrix} I & -AC^{-1} \\ 0 & I \end{pmatrix} \cdot \begin{pmatrix} A^{-1} & 0 \\ 0 & X^{-1} \end{pmatrix} \cdot \begin{pmatrix} I & 0 \\ -B^{-1}A & I \end{pmatrix}$$

$$X = D - CA^{-1}B.$$

$$\begin{pmatrix} I & A & 0 \\ 0 & I & B \\ 0 & 0 & I \end{pmatrix}^{-1} = \begin{pmatrix} I & -A & AB \\ 0 & I & -B \\ 0 & 0 & I \end{pmatrix}$$

Ταξινόμηση

Sorting Problem

Είσοδος: Ένας αταξινόμητος πίνακας A με n διακριτά κλειδιά:

$$A[1], \ldots, A[n]$$

Έξοδος: Ένας ταξινομημένος πίνακας A:

$$A[1] < A[2] < \ldots < A[n]$$

Αλγόριθμοι Ταξινόμησης

- Μέθοδος της Φυσαλλίδας (Bubble-Sort): $O(n^2)$ χειρότερη και μέση περίπτωση.
- Μέθοδος Εισαγωγής (Insertion-Sort): $O(n^2)$ χειρότερη και μέση περίπτωση.
- Μέθοδος Συγχώνευσης (Merge-Sort): $O(n \log n)$ χειρότερη και μέση περίπτωση.
- ΓρήγορηΤαξινόμηση (Quick-Sort): $O(n \log n)$ μέση περίπτωση. $O(n^2)$ χειρότερη περίπτωση.
- Μέθοδος Σωρού (Heap-Sort): $O(n \log n)$ χειρότερη και μέση περίπτωση.

Ταξινόμηση Φυσαλλίδας (BubbleSort)

```
procedure Bubble_Sort(a[1..n]);
begin
    for i := 1 to n−1 do
        for j := n downto i + 1 do
            if a[j] < a[j − 1] then swap(a[j], a[j − 1])
end
```

Αριθμός Συγκρίσεων:

$$(n-1)+(n-2)+\ldots+1=\frac{n(n-1)}{2}=O(n^2).$$

Ταξινόμηση Εισαγωγής (InsertionSort)

```
procedure Insertion_Sort(a[1..n]);
begin
    for j := 2 to n do
    begin
        key := a[j]; i := j − 1;
        while (i > 0) and (key < a[i]) do (* ολίσθηση *)
        begin a[i + 1] := a[i]; i = i − 1
        end;
        a[i + 1] := key (*τοποθέτηση*)
    end
end
```

Αριθμός συγκρίσεων:

ίδιος με Bubblesort (χειρότερη περίπτωση)

Ταξινόμηση Mergesort (I)

```
procedure Merge(a[p..mid] and b[mid+1..q] into c[p..q]);
(* Συγχωνεύει τα ταξινομημένα a και b δίνοντας το c *)
begin
  i:=p; j:=mid+1; k:=i;
  while (i<=mid) and (j<=q) do
   begin
     if a[i]<b[j] then
       begin c[k]:=a[i]; i:=i+1 end
     else
       begin c[k]:=b[j]; j:=j+1 end;
     k:=k+1
   end
  if i>mid then for l:=j to q do
    begin c[k]:=b[l]; k:=k+1 end
  else for l:=i to mid do
    begin c[k]:=a[l]; k:=k+1 end
end
```

Ταξινόμηση Mergesort (II)

```
procedure MergeSort (a[p..q]: list);
begin
  if p=q then return a[p]
  else
  begin
    mid:=(p+q) div 2;
    mergesort(a[p..mid]); mergesort(a[mid+1..q]);
    merge( (a[p..mid] and a[mid+1..q]) into a[p..q] )
  end
end
```

Πολυπλοκότητα Mergesort

$$T(n) = \begin{cases} 0 & , \text{για } n = 1 \\ 2T(\frac{n}{2}) + n - 1 & , \text{για } n > 1 \end{cases}$$

$$T(n) = n \log n - n + 1 = O(n \log n)$$

Ταξινόμηση QuickSort

```
procedure Partition (p, q:integer; var k:integer);
begin
  i:=p; j:=q; x:=a[k];
  repeat while a[i]<x do i:=i+1; while x<=a[j] do j:=j−1;
         if i<j then swap(a[i],a[j]);
  until i>=j;
  k:=i
end

procedure Quicksort (p, q);
begin
  if p<q then begin choose k from [p..q]; Partition (p, q, k);
                    Quicksort(p, k−1); Quicksort(k,q);
              end
end
```

Παράδειγμα QuickSort

1	2	3	4	5	6	7	8	9	10
$\overrightarrow{\text{Τ}}$	Α	Ξ	Ι	Ν	Ο	Μ	Η	Σ	$\overleftarrow{\text{Η}}$
Η	Α	$\overrightarrow{\text{Ξ}}$	Ι	Ν	Ο	Μ	$\overleftarrow{\text{Η}}$	Σ	Τ
Η	Α	Η	Ι	$\overrightarrow{\text{Ν}}$	Ο	$\overleftarrow{\text{Μ}}$	Ξ	Σ	Τ
Η	Α	Η	Ι	$\overleftarrow{\text{Μ}}$	$\overrightarrow{\text{Ο}}$	Ν	Ξ	Σ	Τ
Η	Α	Η	Ι	Μ	Ο	Ν	Ξ	Σ	Τ

Παράδειγμα QuickSort (συν.)