ΔΙΑΙΡΕΙ ΚΑΙ ΒΑΣΙΛΕΥΕ

Ένας Αλγόριθμος Διαίρει και Βασίλευε συνίσταται στις εξής σχεδιαστικές αποφάσεις:

1. <u>**ΒΗΜΑ ΔΙΑΙΡΕΣΗΣ:**</u> Διάσπαση του αρχικού προβλήματος σε μικρότερα επιμέρους υποπροβλήματα.
2. <u>**ΒΗΜΑ ΕΠΙΛΥΣΗΣ ΕΠΙΜΕΡΟΥΣ ΣΤΙΓΜΙΟΤΥΠΩΝ:**</u> Επίλυση των επιμέρους υποπροβλημάτων (με αναδρομικές κλήσεις του ίδιου αλγόριθμου)
3. <u>**ΒΗΜΑ ΣΥΝΘΕΣΗΣ ΛΥΣΕΩΝ:**</u> Υπολογισμός της λύσης του αρχικού προβλήματος, από τις επιμέρους λύσεις των υποπροβλημάτων.

Αλγόριθμοι Διαίρει και Βασίλευε:

1. **MergeSort**, για το πρόβλημα ταξινόμησης μιας ακολουθίας n ακεραίων, Αναδρομική Σχέση: T(n)=2T(n/2)+n. Πολυπλοκότητα: O(nlogn)
2. **QuickSort** για το πρόβλημα της ταξινόμησης μιας ακολουθίας n ακεραίων. Αναδρομική Σχέση: T(n)=T(k)+T(n-k)+n, πολυπλοκότητα $O(n^2)$ στην χείριστη περίπτωση.
3. **BinarySearch**, για το πρόβλημα αναζήτησης στοιχείου σε μία ακολουθία n ακεραίων, Αναδρομική Σχέση: T(n)=T(n/2)+n. Πολυπλοκότητα: O(logn)
4. **QuickSelect** για την επιλογή του στοιχείου που είναι στην θέση k στην ταξινομημένη ακολουθία. Αναδρομική Σχέση: T(n)=T(7n/10)+n. Πολυπλοκότητα: O(n).
5. **Strassen**, για τον πολλαπλασιασμό δύο nxn πινάκων. Αναδρομική Σχέση: $T(n)=7T(n/2)+\Theta(n^2)$. Πολυπλοκότητα: $\Theta(n^{2.81})$

ΔΥΝΑΜΙΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ

ΣΧΕΔΙΑΣΗ ΑΛΓΟΡΙΘΜΩΝ www.psounis.gr

Σχεδιάζουμε αλγόριθμο δυναμικού προγραμματισμού σε **προβλήματα** που έχουν τα εξής **χαρακτηριστικά**:

- <u>**Ιδιότητα των Βέλτιστων Επιμέρους Δομών:**</u> Οτί για να λύσουμε το πρόβλημα αρκεί να υπολογίσουμε την βέλτιστη λύση σε κάποια υποπροβλήματα, συνήθως με αναδρομή.
- <u>**Μικρός Αριθμός Υποπροβλημάτων:**</u> Το πλήθος των υποπροβλημάτων που πρέπει να λύσουμε είναι μικρό (δηλαδή πολυωνυμικό ως προς το μέγεθος του προβλήματος)
- <u>**Επικαλυπτόμενα Επιμέρους Προβλήματα:**</u> Ότι λύνουμε πολλές φορές τα ίδια υποπροβλήματα με αποτέλεσμα να χάνουμε χρόνο

<u>**Βήματα Σχεδίασης Αλγόριθμου Δυναμικού Προγ/μού**</u>

1. Περιγράφουμε έναν **αναδρομικό αλγόριθμο** που λύνει το πρόβλημα
2. Δίνουμε την **αναδρομική σχέση** που υπολογίζει την βέλτιστη λύση (επίλυση από πάνω προς τα κάτω)
3. Διαπιστώνουμε ότι ισχύουν οι **τρεις συνθήκες** για την κατασκευή του αλγορίθμου δυναμικού προγραμματισμού.
4. Με βάση την αναδρομική σχέση, κατασκευάζουμε την διαδικασία επίλυσης από τα μικρά προβλήματα σε όλο και μεγαλύτερα (**επίλυση από κάτω προς τα πάνω**)
5. Δίνουμε τον **επαναληπτικό αλγόριθμο** που κάνει την επίλυσή του προβλήματος
6. Υπολογίζουμε την **πολυπλοκότητα** του επαναληπτικού αλγορίθμου

Αλγόριθμοι Δυναμικού Προγραμματισμού:

1. <u>**Υπολογισμός Αριθμού Fibonacci**. Πολυπλοκότητα: O(n)</u>
2. <u>**Αλυσιδωτός Πολλαπλασιασμός Πινάκων**. Πολυπλοκότητα $O(n^3)$</u>
3. <u>**Μέγιστη Κοινή Υπακολουθία**. Πολυπλοκότητα: Θ(nm).</u>
4. <u>**Συντομότερο Μονοπάτι σε Άκυκλο Κατευθυνόμενο Γράφημα (DAG)**. Πολυπλοκότητα: $O(n^2)$.</u>

ΑΠΛΗΣΤΟΙ ΑΛΓΟΡΙΘΜΟΙ

ΣΧΕΔΙΑΣΗ ΑΛΓΟΡΙΘΜΩΝ www.psounis.gr

Σχεδιάζουμε αλγόριθμο δυναμικού προγραμματισμού σε **προβλήματα** που έχουν τα εξής **χαρακτηριστικά**:

- <u>Ιδιότητα της Άπληστης Επιλογής:</u> Μια ακολουθία άπληστων επιλογών οδηγεί στην βέλτιστη λύση.
- <u>Ιδιότητα των Βέλτιστων Επιμέρους Δομών:</u> Οτί για να λύσουμε το πρόβλημα αρκεί να υπολογίσουμε την βέλτιστη λύση σε κάποια υποπροβλήματα, συνήθως με αναδρομή.

<u>**Συνήθης διαδικασία για την κατασκευή ενός άπληστου αλγορίθμου**</u>

1. Ταξινομούμε τα δεδομένα από τα οποία επιλέγουμε την λύση
2. Επιλέγουμε το επόμενο στοιχείο με βάση την ταξινόμηση για να το εισάγουμε στη λύση μας.
   1. Αν η λύση που προκύπτει δεν παραβιάζει τους περιορισμούς του προβλήματος, διατηρούμε το στοιχείο στη λύση
   2. Αν η λύση που προκύπτει παραβιάζει τους περιορισμούς του προβλήματος, τότε απορρίπτουμε το στοιχείο.

   Εωσότου κατασκευαστεί η λύση

Ένας Άπληστος Αλγόριθμος:

<u>**Μπορεί να είναι βέλτιστος:**</u> Η απόδειξη γίνεται με δύο εναλλακτικούς (και συμπληρωματικούς) τρόπους:

1. Με μαθηματική επαγωγή. Ότι κάθε επιλογή του άπληστου αλγορίθμου είναι βέλτιστη.
2. Με απόδειξη των δύο ιδιοτήτων (βέλτιστες επιμέρους δομές και άπληστη επιλογή)

<u>**Μπορεί να μην είναι βέλτιστος:**</u> Η απόδειξη γίνεται με κατάλληλο αντιπαράδειγμα:

1. Δείχνουμε ότι ο αλγόριθμος επιστρέφει μία λύση που έχει ένα κόστος, που είναι χειρότερο από
2. Την βέλτιστη λύση.

Παραδείγματα Άπληστων Αλγορίθμων:

1. <u>**Αγλόριθμος Dijkstra** για υπολογισμό συντομότερων μονοπατιών σε γράφημα:</u> Πολυπλοκότητα: $O(n^2)$ και με ειδική δομή δεδομένων: O(m+nlogn)
2. <u>**Αλγόριθμος Prim** για υπολογισμό συνδετικού δένδρου ελαχίστου κόστους:</u> Πολυπλοκότητα: $O(n^2)$ και με ειδική δομή δεδομένων: O(m+nlogn)
3. <u>**Αλγόριθμος Kruskal** για υπολογισμό συνδετικού δένδρου ελαχίστου κόστους:</u> Πολυπλοκότητα O(m logn)
4. <u>**Επιστροφή Ρέστων.** Πολυπλοκότητα: O(X), όπου X το ποσό επιστροφής</u>

QUICKSORT (ΓΡΗΓΟΡΗ ΤΑΞΙΝΟΜΗΣΗ)

ΕΙΣΟΔΟΣ: Πίνακας Στοιχείων

ΕΞΟΔΟΣ: Ταξινομημένος (σε αύξουσα σειρά) πίνακας

<u>**ΑΛΓΟΡΙΘΜΙΚΗ ΙΔΕΑ:**</u> Αναδρομικά:

- Βρες Οδηγό Στοιχείο
- Βάλε τα μικρότερα αριστερά και τα μικρότερα δεξιά
- Αναδρομικά ταξινόμησε τους δύο πίνακες.

<u>**ΨΕΥΔΟΚΩΔΙΚΑΣ**</u>

```
procedure QuickSort(A,start,finish)
   if start<finish then
      pos=Partition(A,start,finish)
      QuickSort(A,start,pos-1)
      QuickSort(A,pos+1,finish)
   end if
end procedure
```

```
procedure Partition(A,start,finish)
    odigo=A[start]
    i=start;   j=finish
    for (k=start+1 to finish)
       if (A[k]>odigo)
          B[j]=A[k]; j=j-1
       else
          B[i]=A[k]; i=i+1
    end for
    B[i]=odigo; A=B
    return pos;
end procedure
```

<u>**ΠΑΡΑΔΕΙΓΜΑ ΕΚΤΕΛΕΣΗΣ:**</u>

<u>Αναδρομικές Κλήσεις:</u>

<u>**ΠΟΛΥΠΛΟΚΟΤΗΤΑ:**</u>
Εξαρτάται από το οδηγό στοιχείο (με βάση αυτό αλλάζει το πλήθος των δεδομένων των αναδρομικών κλήσεων

Χειρότερη Περίπτωση: $T(n) = T(n-1) + \Theta(n)$

Μέθοδος .... Επανάληψης

$T(n) = O(n^2)$

QUICKSELECT (ΓΡΗΓΟΡΗ ΕΠΙΛΟΓΗ)

ΕΙΣΟΔΟΣ: Δίνεται ένας <u>αταξινόμητος</u> πίνακας με n στοιχεία. Ζητείται να βρεθεί το k-μικρότερο στοιχείο
ΕΞΟΔΟΣ: Η θέση του k-μικρότερου στοιχείου

<u>ΑΛΓΟΡΙΘΜΙΚΗ ΙΔΕΑ:</u> Αναδρομικά:
- Βρες Οδηγό Στοιχείο
- Βάλε τα μικρότερα αριστερά και τα μικρότερα δεξιά
- Αναδρομικά επέλεξε τον έναν υποπίνακα.

<u>ΨΕΥΔΟΚΩΔΙΚΑΣ</u>

```
procedure QuickSelect(A,start,finish,k)
  if start>finish then
    return 0
  else
    Επιλογή στοιχείου m με την διαδικασία των
                                        5-άδων.
    swap(A[m],A[start])
    pos=Partition(A,start,finish)
    if k=pos then
      return A[pos]
    else if k<pos then
      return QuickSelect(A,start,pos-1,k)
    else if k>pos then
      return QuickSelect(A,pos+1,finish,k-pos)
    end if
  end if
end procedure
```

```
procedure Partition(A,start,finish)
    … βλέπε QuickSort …
end procedure
```

<u>ΠΟΛΥΠΛΟΚΟΤΗΤΑ:</u>
T(n)=O(n) στην χειρότερη περίπτωση

<u>ΠΑΡΑΔΕΙΓΜΑ ΕΚΤΕΛΕΣΗΣ</u><u>(Αναζητώ το 12ο μικρότερο)</u>
<u>Αναδρομικές Κλήσεις</u>:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
18	7	4	11	9	20	6	1	22	19	14	5	2	3	10	13
7	4	11	9	6	1	14	5	2	3	10	13	18	19	22	20

1	2	3	4	5	6	7	8	9	10	11	12
7	4	11	9	6	1	14	5	2	3	10	13
4	6	1	5	2	3	7	13	10	14	9	11

8	9	10	11	12
13	10	14	9	11
10	9	11	13	14

12
14
14

<u>Διαδικασία 5-άδων:</u> (Επιλογή του μεσαίου των μεσαίων …)

- Π.χ.:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
18	7	4	11	9	20	6	1	22	19	14	5	2	3	10	13

- 5-άδες

1	2	3	4	5
18	7	4	11	9

6	7	8	9	10
20	6	1	22	19

11	12	13	14	15
14	5	2	3	10

16
13

- Νεος Πίνακας

1	2	3	4
9	19	5	13

αναδρομική εκτέλεση

- Επιστρέφεται το 9 (επιλογή οδηγού στοιχείου)

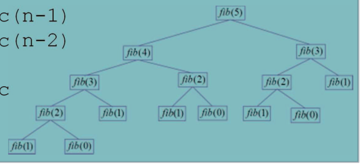
ΑΚΟΛΟΥΘΙΑ FIBONACCI

ΕΙΣΟΔΟΣ: Φυσικός n
ΕΞΟΔΟΣ: Ο n-οστός Fibonacci

$$f_n = \begin{cases} 1, & n=1 \ ή \ n=2 \\ f_{n-1}+f_{n-2}, & n>2 \end{cases}$$

<u>ΨΕΥΔΟΚΩΔΙΚΑΣ (ΜΕ ΑΝΑΔΡΟΜΗ)</u>

```
procedure FibRec(n)
   if n=1 or n=2 then
      return 1
   else
      a=FibRec(n-1)
      b=FibRec(n-2)
      c=a+b
      return c
   end if
end procedure
```



<u>ΠΟΛΥΠΛΟΚΟΤΗΤΑ ΑΝΑΔΡΟΜΙΚΟΥ ΑΛΓΟΡΙΘΜΟΥ:</u>

$$T(n) = \begin{cases} \Theta(1), & n=1 \ ή \ n=2 \\ T(n-1)+T(n-2)+\Theta(1), & n>2 \end{cases}$$

<u>Κάτω Φράγμα:</u>

$$\mathrm{K}(n) = 2\mathrm{K}(n-2) + \Theta(1)$$

… Μέθοδος Επανάληψης

$$T(n) = \Omega(2^{\frac{n}{2}})$$

<u>ΔΥΝΑΜΙΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ:</u>
- Υπολόγισε την λύση επαναληπτικά από 1…n

<u>ΨΕΥΔΟΚΩΔΙΚΑΣ (ΜΕ ΕΠΑΝΑΛΗΨΗ)</u>

```
procedure FibSeq(n)
   A[1]=1
   A[2]=1
   for i=3 to n
      A[i]=A[i-1]+A[i-2]
   end for
   return A[n]
end procedure
```

<u>ΠΑΡΑΔΕΙΓΜΑ ΕΚΤΕΛΕΣΗΣ:</u>

1	2	3	4	5	6	…	n
1	1	2	3	5	8	…	fib(n)

<u>ΠΟΛΥΠΛΟΚΟΤΗΤΑ Αλγόριθμου Δ.Π.:</u>
T(n)=Θ(n)

<u>Άνω Φράγμα:</u>

$$\mathrm{A}(n) = 2\mathrm{A}(n-1) + \Theta(1)$$

… Μέθοδος Επανάληψης

$$T(n) = O(2^n)$$

ΑΛΥΣΙΔΩΤΟΣ ΠΟΛΛΑΠΛΑΣΙΑΣΜΟΣ ΠΙΝΑΚΩΝ

ΕΙΣΟΔΟΣ: $A_1, A_2, \dots, A_n$ όπου ο πίνακας A_i είναι διάστασης $d_{i-1} \times d_i$.
ΕΞΟΔΟΣ: Η σειρά που πολλαπλασιασμών του γινομένου $A_1 \times A_2 \times \dots \times A_n$

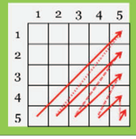
<u>ΑΝΑΔΡΟΜΙΚΗ ΣΧΕΣΗ</u> (υπολογισμού της βέλτιστης λύσης):

$$M[i,j] = \begin{cases} 0, & i=j \\ \min\limits_{i \le k < j}\{M[i,k]+M[k+1,j]+d_{i-1}d_k d_j\}, & i<j \end{cases}$$

<u>ΠΑΡΑΔΕΙΓΜΑ ΕΚΤΕΛΕΣΗΣ:</u> για τον πολλαπλασιασμό πινάκων $A_1A_2A_3A_4$, όταν
A_1: 5x3, A_2: 3x4, A_3: 4x8, A_4:8x2, A_5: 2x3

	1	2	3	4
5				
4				A4 0
3			A3 0	A3A4 4x8x2=64
2		A2 0	A2A3 3x4x8=96	A2A3A4 (A2A3)A4=96+3x8x2=96+48=144 <u>A2(A3A4)</u>=64+3x4x2=64+24=88 88
1	A1 0	A1A2 5x3x4=60	A1A2A3 (A1A2)A3=60+5x4x8=220 <u>A1(A2A3)</u>=96+5x3x8=216 216	A1A2A3A4 <u>A1(A2A3A4)</u>:88+5x3x2=88+30=118 (A1A2)(A3A4):60+64+5x4x2=164 (A1A2A3)A4:216+5x8x2=296 118

<u>ΔΥΝΑΜΙΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ:</u>
- Υπολόγισε την αναδρομή επαναληπτικά με βάση τη σειρά σύμφωνα με το ακόλουθο σχήμα:



<u>ΨΕΥΔΟΚΩΔΙΚΑΣ (ΜΕ ΕΠΑΝΑΛΗΨΗ)</u>

```
procedure DP_MatMult(A1,A2,…,An)
    for i=1 to n
       m[i,i]=0
    end for
    for p=2 to n
       for i=2 to n-p+1
          j=i+p-1
          m[i,j]=+∞
          for k=1 to j-1
             q=M[i,k]+M[k+1,j]+d[i-1]*d[k]*d[j]
             if (q>M[i,j]) then M[i,j]=q ,
                                 s[i,j]=k
          end for
       end for
    end for
    return M[1,n]
end procedure
```

<u>ΠΟΛΥΠΛΟΚΟΤΗΤΑ Αλγόριθμου Δ.Π.:</u>
$T(n)=O(n^3)$

ΜΕΓΙΣΤΗ ΚΟΙΝΗ ΥΠΑΚΟΛΟΥΘΙΑ

ΕΙΣΟΔΟΣ: Δίδονται ακολουθίες χαρακτήρων $X=x_1x_2x_3\dots x_n$ και $Y=y_1y_2\dots y_m$

ΕΞΟΔΟΣ: Το μέγιστο μήκος κοινής τους υπακολουθίας

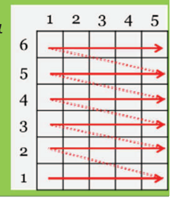
<u>ΑΝΑΔΡΟΜΙΚΗ ΣΧΕΣΗ</u> (υπολογισμού της βέλτιστης λύσης):

$$f_n = \begin{cases} 0, & i=0 \ ή \ j=0 \\ c[i-1,j-1]+1, & i,j>0 \ και \ x_i=y_j \\ \max\{c[i,j-1],c[i-1,j]\}, & i,j>0 \ και \ x_i \neq y_j \end{cases}$$

<u>ΠΑΡΑΔΕΙΓΜΑ ΕΚΤΕΛΕΣΗΣ:</u> για τις συμβολοσειρές
X=a**b**c**df** και Y=d**bd**a**f**

	1	2	3	4	5
5	X=abcdf Y=d c=1	X=abcdf Y=db c=1	X=abcdf Y=dbd c=2	X=abcdf Y=dbda c=2	X=abcd**f** Y=dbda**f** c=3
4	X=abcd Y=d c=1	X=abcd Y=db c=1	X=abc**d** Y=db**d** c=2	X=abcd Y=dbda c=2	X=abcd Y=dbdaf c=2
3	X=abc Y=d c=0	X=abc Y=db c=1	X=abc Y=dbd c=1	X=abc Y=dbda c=1	X=abc Y=dbdaf c=1
2	X=ab Y=d c=0	X=a**b** Y=d**b** c=1	X=ab Y=dbd c=1	X=ab Y=dbda c=1	X=ab Y=dbdaf c=1
1	X=a Y=d c=0	X=a Y=db c=0	X=a Y=dbd c=0	X=a Y=dbda c=1	X=a Y=dbdaf c=1

<u>ΔΥΝΑΜΙΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ:</u>
- Υπολόγισε την αναδρομή επαναληπτικά με βάση τη σειρά σύμφωνα με το ακόλουθο σχήμα:



<u>ΨΕΥΔΟΚΩΔΙΚΑΣ (ΔΥΝΑΜΙΚΟΥ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ)</u>

```
procedure LCS(X,Y)
    for i=1 to n : c[i,0]=0
    for j=1 to m : c[0,j]=0
    for i=1 to n
       for j=1 to m
          if xi=yj then
             c[i,j]=c[i-1,j-1]+1
          else
             if (c[i-1,j]>c[i,j-1]) then
                c[i,j]=c[i-1,j]
             else
                c[i,j]=c[i,j-1]
             end if
          end if
       end for
    end for
    return c[n,m]
end procedure
```

<u>ΠΟΛΥΠΛΟΚΟΤΗΤΑ Αλγόριθμου Δ.Π.:</u>
T(n)=O(nm)

ΣΥΝΤΟΜΟΤΕΡΟ ΜΟΝΟΠΑΤΙ ΣΕ ΑΚΥΚΛΟ ΓΡΑΦΗΜΑ

ΕΙΣΟΔΟΣ: Δίνεται άκυκλο κατευθυνόμενο γράφημα G=(V,E,W)

ΕΞΟΔΟΣ: Το συντομότερο μονοπάτι από την αφετηρία στον προορισμό

ΑΝΑΔΡΟΜΙΚΗ ΣΧΕΣΗ (προϋποθέτει τοπολογική ταξινόμηση των κόμβων 1,2,…,n):

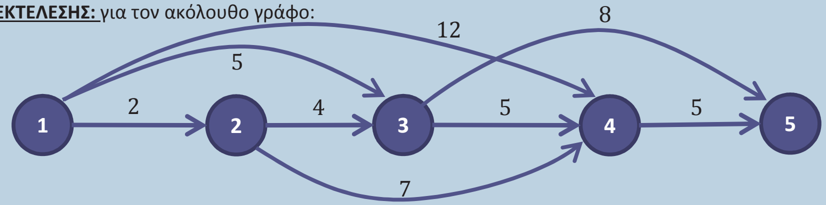
$$OPT[n] = \begin{cases} 0, & n = 1 \\ min\{OPT[j] + W[j,n] \mid (j,n) \in E\} & n > 1 \end{cases}$$

ΔΥΝΑΜΙΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ:

- Αφού πρώτα γίνει μία ταξινόμηση των κόμβων ώστε στην διάταξη τους κάθε ακμή να είναι (v_i,v_j) με i<j (τοπολογική ταξινόμηση)
- Ο δυναμικός προγραμματισμός υπολογίζει επαναληπτικά την αναδρομική σχέση για i=1,…,n.

ΠΟΛΥΠΛΟΚΟΤΗΤΑ Αλγόριθμου Δ.Π.:
T(n)=O(n+m)

ΠΑΡΑΔΕΙΓΜΑ ΕΚΤΕΛΕΣΗΣ: για τον ακόλουθο γράφο:



	1: **0** + 2 = 2	1: **0** + 5 = 5 2: **2** + 4 = 6	1: **0** + 12 = 12 2: **2** + 7 = 9 3: **5** + 5 = 10	3: **5** + 8 = 13 4: **9** + 5 = 14
OPT[1]=**0**	OPT[2]=**2**	OPT[3]=**5**	OPT[4]=**9**	OPT[5]=**13**