Αλγόριθμοι Τύπου
Μείωσης Προβλήματος

Περίληψη

- Αλγόριθμοι Τύπου Μείωσης Προβλήματος ("Decrease and Conquer")
  - Μείωση κατά μια σταθερά (decrease by a constant)
  - Μείωση κατά ένα ποσοστό (decrease by a constant factor)
  - Μείωση κατά μεταβλητό μέγεθος (variable-size-decrease)

Αλγόριθμοι Μείωσης Προβλήματος κατά μια Σταθερά

- Γενική Μεθοδολογία
  - Αντί να λύσουμε το πιο δύσκολο πρόβλημα μεγέθους n, λύνουμε το πιο εύκολο πρόβλημα μεγέθους n-m.
    - Συνήθως το m=1.
  - Από τη λύση του μικρότερου προβλήματος μπορούμε να βρούμε τη λύση του προβλήματος μεγέθους n.
- Απλό Παράδειγμα: Υπολογίστε τη συνάρτηση a^n

```
power(a, n)
  if(n = 1) return a;
  else return power(a, n-1)*a;
```

Προβλήματα Ταξινόμησης

- Είσοδος: Λίστα $A[0,\ldots,n-1]$
- Έξοδος: Ταξινομημένη λίστα $A[0,\ldots,n-1]$

- InsertionSort:
- **Βασική Ιδέα**
  - Υποθέτουμε πως τα πρώτα k στοιχεία της λίστας είναι ταξινομημένα και δημιουργούμε μια ταξινομημένη λίστα με $k+1$ στοιχεία.

Προβλήματα Ταξινόμησης

- Είσοδος: Λίστα $A[0,\ldots,n-1]$
- Έξοδος: Ταξινομημένη λίστα $A[0,\ldots,n-1]$

```
insertionsort(A[0,…,n-1])
  for i=0 to n-1
      new= A[i];
      j=i-1;
      while j>=0 and A[j]> new
          A[j+1]= A[j];
          j= j-1;
      A[j+1]= new;
```

Απόδοση InsertionSort()

```
insertionsort(A[0,…,n-1])
   for i=0 to n-1
        new= A[i];
        j=i-1;
        while j>=0 and A[j]> new
                A[j+1]= A[j];
                j= j-1;
        A[j+1]= new;
```

- Χειρότερη Περίπτωση:

- Καλύτερη Περίπτωση:

Εξερεύνηση Γράφων (Tree Traversal)

- Δεδομένου ενός γράφου, πως μπορεί κάποιος να επισκεφτεί όλους τους κόμβους του γράφου.
  - Αυτό είναι ένα πρόβλημα το οποίο παρουσιάζεται συχνά σε γράφους.
  - Μπορεί να υπάρξουν πολλές λύσεις. Θα μελετηθούν δύο τρόποι που εμπίπτουν κάτω από την κατηγορία «μείωσης» του προβλήματος κατά μια σταθερά (ένα).
  - Η βασική ιδέα είναι ότι το πιο δύσκολο πρόβλημα εξερεύνησης γράφου με n κόμβους «σπάζει» σε ένα πρόβλημα για n-1 κόμβους αφού πρώτα επισκεφτούμε τον αμέσως επόμενο κόμβο.
    - Με το ίδιο σκεπτικό θα μπορούσε κάποιος να κατατάξει τους αλγόριθμους αυτούς κάτω από την κατηγορία των greedy αλγορίθμων.

Depth-First Search (DFS) και Breadth-First Search (BFS)

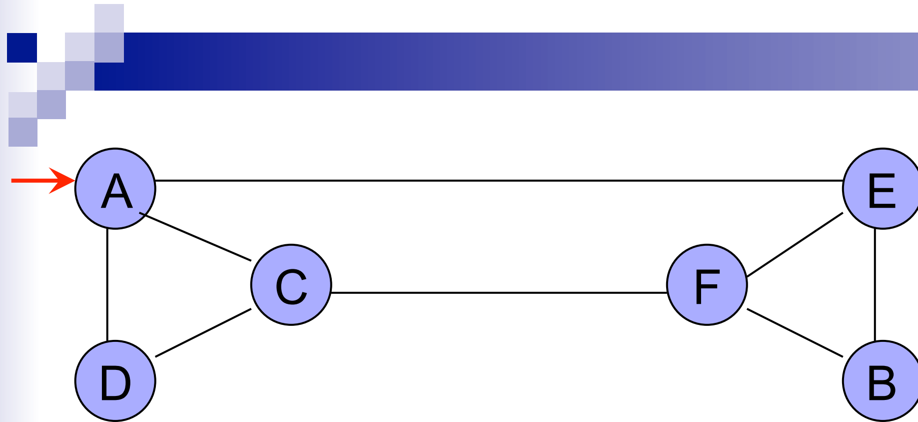
- Η Γενική προσέγγιση του DFS είναι αφού επισκεφτεί ένα κόμβο, να ψάξει για τον αμέσως επόμενο γειτονικό κόμβο.
  - Πολύ γρήγορα ο αλγόριθμος «απομακρύνεται» από ένα κόμβο, πριν επισκεφτεί όλους τους γείτονες.
  - Υλοποιείται αναδρομικά ή με στοίβα.
- Η γενική προσέγγιση του BFS είναι να επισκεφτεί όλους τους γείτονες ενός κόμβου πριν «απομακρυνθεί».
  - Υλοποιείτε με ουρά
- Οι δύο αλγόριθμοι έχουν την ίδια απόδοση διαφέρουν στην σειρά με την οποία θα επισκεφτούν τους κόμβους.
- Εφαρμογές: Έλεγχος συνεκτικότητας, έλεγχος για τυχόν βρόγχους, έλεγχος για «τρωτά» σημεία, κλπ.

DFS

```
DFS(G=<V,E>)
  for all v in V
     if v.visited= false;
          dfs(v)
```

```
dfs(v)
  v.visited= true;
  for w= 1 to wn
     if w.visited= false;
          dfs(w)
```

DFS: Παράδειγμα



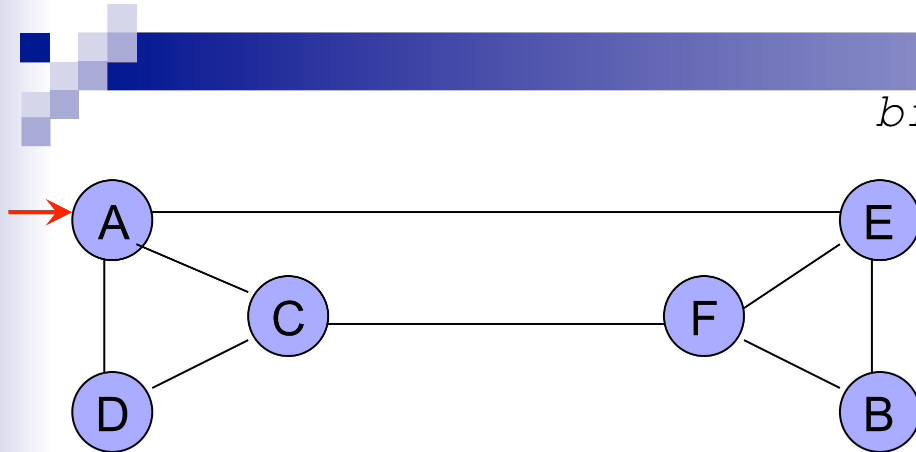

```
dfs(v)
  v.visited= true;
  for w= 1 to wn
     if w.visited= false;
        dfs(w)
```

Στοίβα

BFS

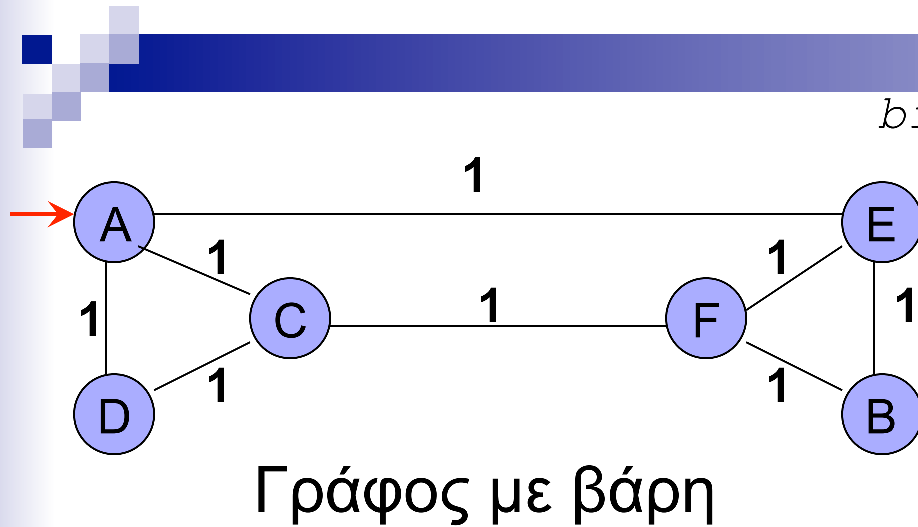
```
BFS(G=<V,E>)
  for all v in V
      if v.visited= false;
            bfs(v)

                   bfs(v)
                     v.visited= true;
                     while Q not empty
                         for w= 1 to wn
                               if w.visited= false;
                                     w.visited= true;
                                     Q.enqueue(w);
                         Q.dequeue(Q.head);
```

BFS: Παράδειγμα




```
bfs(v)
  v.visited= true;
  while Q not empty
      for w= 1 to wn
          if w.visited= false;
              w.visited= true;
              Q.enqueue(w);
      Q.dequeue(Q.head);
```

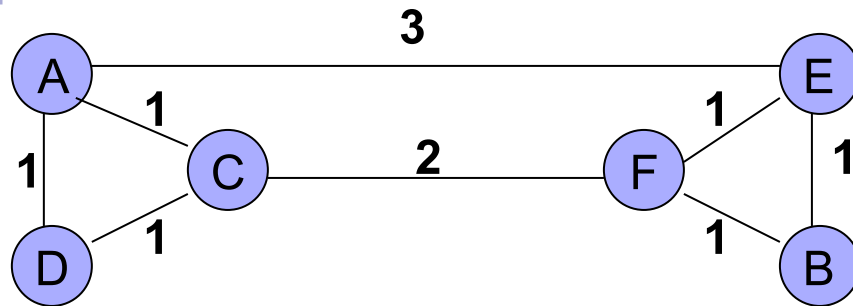
BFS: Παράδειγμα 2



Γράφος με βάρη

```
bfs(v)
  v.visited= true;
  while Q not empty
      for w= 1 to wn
          if w.visited= false;
              w.visited= true;
              Q.enqueue(w);
      Q.dequeue(Q.head);
```

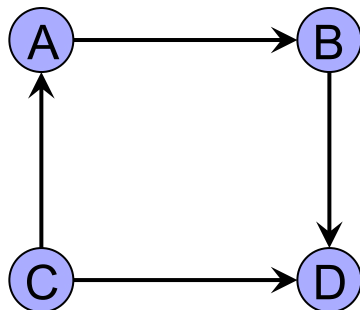
BFS: Παράδειγμα 3

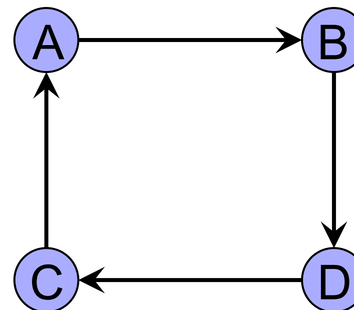


Γράφος με ακέραια βάρη

Τοπολογική Ταξινόμηση

- **Πρόβλημα:** Δεδομένου ενός κατευθυνόμενου γράφου, υπάρχει τρόπος να «ταξινομήσουμε» **όλους** τους κόμβους έτσι ώστε εάν υπάρχει ακμή από τον κόμβο u στον v, στην «ταξινομημένη» σειρά, ο u να εμφανίζεται πριν από τον v;
- **Παραδείγματα:**

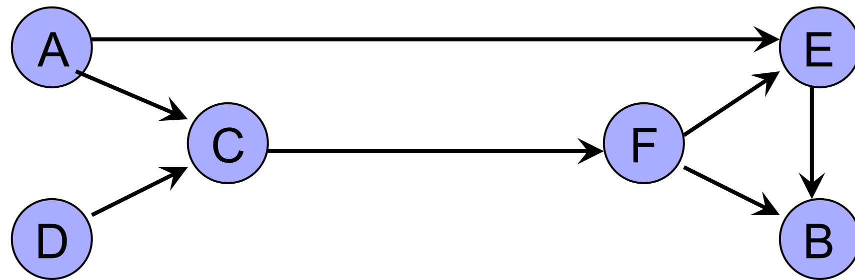




Τοπολογική Ταξινόμηση

- **Θεώρημα:** Το πρόβλημα της τοπολογικής ταξινόμησης έχει λύση εάν και μόνον αν ο κατευθυνόμενος γράφος που περιγράφει το πρόβλημα **δεν** έχει κύκλους (κατευθυνόμενος άκυκλος γράφος – directed acyclic graph (dag)).
- Εφαρμογές τοπολογικής ταξινόμησης:
  - Προαπαιτούμενα μαθήματα
  - Χρονοπρογραμματισμός (scheduling)
    - Διεργασιών σε υπολογιστή
    - Εργασιών για την συμπλήρωση μεγάλων έργων.
- Εξειδικευμένες λύσεις
  - Critical Path Method (CPM)
  - PERT (Program Evaluation and Review)

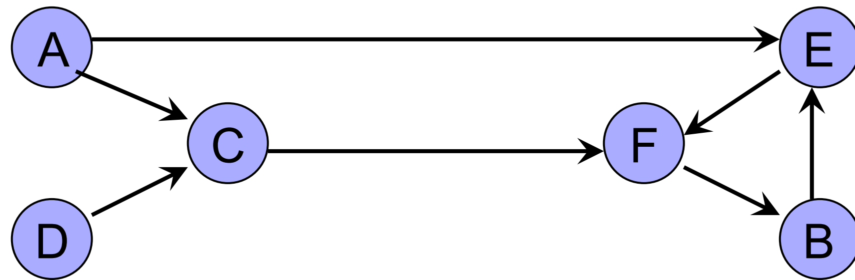
Λύση με DFS




```
dfs(v)
  v.visited= true;
  for w= 1 to wn
     if w.visited= false;
         dfs(w)
```

Στοίβα

Λύση με DFS




```
dfs(v)
  v.visited= true;
  for w= 1 to wn
     if w.visited= false;
        dfs(w)
```

Στοίβα

Τοπολογική Ταξινόμηση: Λύση βασισμένη σε μείωση κατά ένα

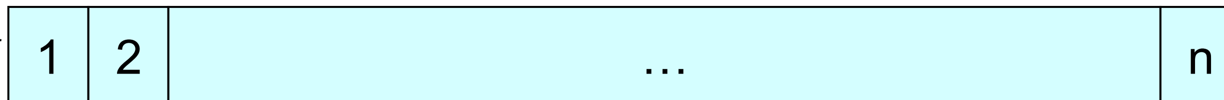
- **Βασική Ιδέα:**
  - Σε κάθε βήμα ψάχνουμε για κύκλο σε ένα γράφο ο οποίος είναι **κατά ένα** μικρότερος από ότι ο προηγούμενος.
  - Από τον μεγαλύτερο γράφο, αφαιρούμε ένα κόμβο στον οποίο δεν καταλήγει πάνω του καμιά ακμή καθώς και όλες τις ακμές που ξεκινούν από το εν λόγω κόμβο.
- **Παράδειγμα:**

Δημιουργία όλων των πιθανών υποσυνόλων και permutations

- **Πρόβλημα**: Σχηματίστε όλα τα πιθανά υποσύνολα (power set) που μπορούμε να δημιουργήσουμε από ένα σύνολο n στοιχείων.
- **Πρόβλημα**: Σχηματίστε όλες τις πιθανές διατάξεις (permutations) τις οποίες μπορούμε να δημιουργήσουμε από ένα σύνολο n στοιχείων.
- Από τη φύση τους τα προβλήματα αυτά δεν μπορούν να έχουν «αποδοτική» λύση! Γιατί;
- Οι διάφοροι συνδυασμοί και διατάξεις δημιουργούνται «σειριακά» οπόταν πολλές φορές είναι επιθυμητό να μην υπάρχουν μεγάλες εναλλαγές από το ένα υποσύνολο στο επόμενο ή τη μια διάταξη στην επόμενη.

Power Set

Διάνυσμα n στοιχείων όπου το κάθε στοιχείο μπορεί να πάρει τη τιμή 0 (εάν το στοιχείο δεν περιλαμβάνεται στο υποσύνολο) και 1 (εάν περιλαμβάνεται)



- Πιθανά υποσύνολα για n=5:

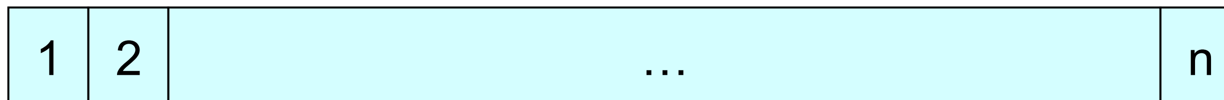
1	2	3	4	5
0	0	0	0	0
0	0	0	0	1
0	0	0	1	0

1	2	3	4	5
0	0	0	1	1
0	0	1	0	0
…	…	…	…	…

- Πόσα πιθανά υποσύνολα;

Permutations

Διάνυσμα n στοιχείων όπου το κάθε στοιχείο μπορεί να πάρει τη τιμή ενός από τα στοιχεία του συνόλου.



- Πιθανά permutations για n=5:

1	2	3	4	5
1	2	3	5	4
1	2	5	3	4

1	5	2	3	4
5	1	2	3	4
…	…	…	…	…

- Πόσα πιθανά υποσύνολα;

Permutations: Αλγόριθμος βασισμένος σε μείωση κατά ένα

- Υποθέστε πως ξέρουμε πώς να δημιουργήσουμε όλα τα permutations για n-1, πως μπορούμε να δημιουργήσουμε τα permutations για n;
- Απλά τοποθετούμε το νέο στοιχείο μεταξύ όλων των στοιχείων από τις υπάρχουσες διατάξεις

n=1

1				

n=2

1	2			
2	1			

n=3

3	1	2		
1	3	2		
1	2	3		
3	2	1		
2	3	1		
2	1	3		

- Ο αλγόριθμος Johnson-Trotter δημιουργεί όλα τα permutations χωρίς να πρέπει να δημιουργήσει τα n-1, n-2, … 1, permutations και σε κάθε βήμα αλλάζουν μόνο 2 στοιχεία θέση.

Αλγόριθμοι μείωσης του προβλήματος κατά ένα ποσοστό

- Γενική Μεθοδολογία
  - Αντί να λύσουμε το πιο δύσκολο πρόβλημα μεγέθους n, λύνουμε το πιο εύκολο πρόβλημα μεγέθους n/m.
    - Συνήθως το $m=2$.
  - Από τη λύση του μικρότερου προβλήματος μπορούμε να βρούμε τη λύση του προβλήματος μεγέθους n.
  - Πολύ αποδοτικοί αλγόριθμοι (αναζήτηση σε δυαδικό δέντρο)!
- Απλό Παράδειγμα: Υπολογίστε τη συνάρτηση a^n

$$a^n = \begin{cases} a^{\frac{n}{2}} \cdot a^{\frac{n}{2}} & \text{if } n > 0, \text{ even} \\ a \cdot a^{\frac{n-1}{2}} \cdot a^{\frac{n-1}{2}} & \text{if } n > 1, \text{ odd} \\ a & \text{if } n = 1 \end{cases}$$

$$a^n = \begin{cases} a^{\lfloor \frac{n}{2} \rfloor} \cdot a^{\lceil \frac{n}{2} \rceil} & \text{if } n > 1 \\ a & \text{if } n = 1 \end{cases}$$

Κάλπικο νόμισμα

- Υποθέστε ότι έχετε n νομίσματα εκ των οποίο το ένα κάλπικο. Κάθε νόμισμα ζυγίζει w γραμμάρια ενώ το κάλπικο ζυγίζει $q<w$ γραμμάρια.
- Υποθέστε πως έχετε στη διάθεση σας μία ζυγαριά.
- Πως μπορείτε να εντοπίσετε το κάλπικο νόμισμα κάνοντας λιγότερα από n ζυγίσματα;
- Πόσα ζυγίσματα χρειάζεστε;

Κάλπικο νόμισμα

- Πόσα ζυγίσματα χρειάζεστε;

```
findCoin(C[1,…,n])
   if(n = 1) return C[];  %fake found
   f=floor(n/2);
   P1=C[1,…,f]; P2=C[f+1,…,2f]; P3=C[n]-C[2f];
   X= weight(P1, P2);
   if X=1 return findCoin(P2);  %P1 weights more
   if X=0 return P3;            %P1 & P2 weigh the same
   if X=-1 return findCoin(P1); %P2 weights more
```

Πολλαπλασιασμός Ακεραίων σε Υλικό (Hardware)

- Πως μπορεί κάποιος να πολλαπλασιάσει δύο ακέραιους αριθμούς m, n;

$$p = m \cdot n$$

- Υποθέστε ότι το n είναι άρτιο, τότε

- Υποθέστε ότι το n είναι περιττό, τότε

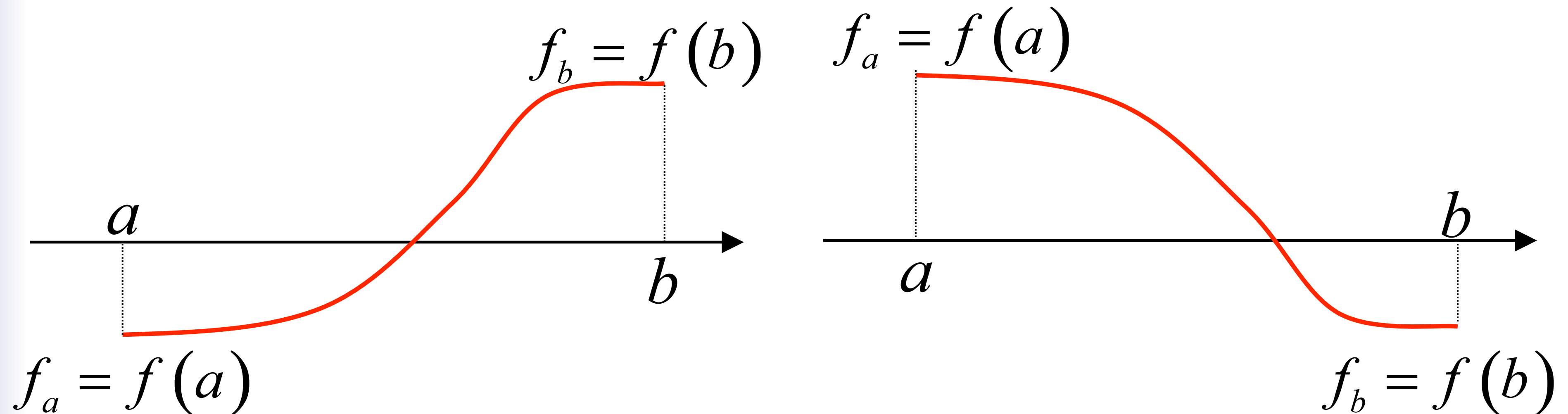
- Κάποιος μπορεί να υλοποιήσει ένα πολλαπλασιαστή με μόνο αθροίσματα και μετατοπίσεις!

Επίλυση μη γραμμικών εξισώσεων μιας μεταβλητής

- Πώς μπορείτε να βρείτε τη λύση της πιο κάτω μη γραμμικής εξίσωσης στο διάστημα $[a,b]$

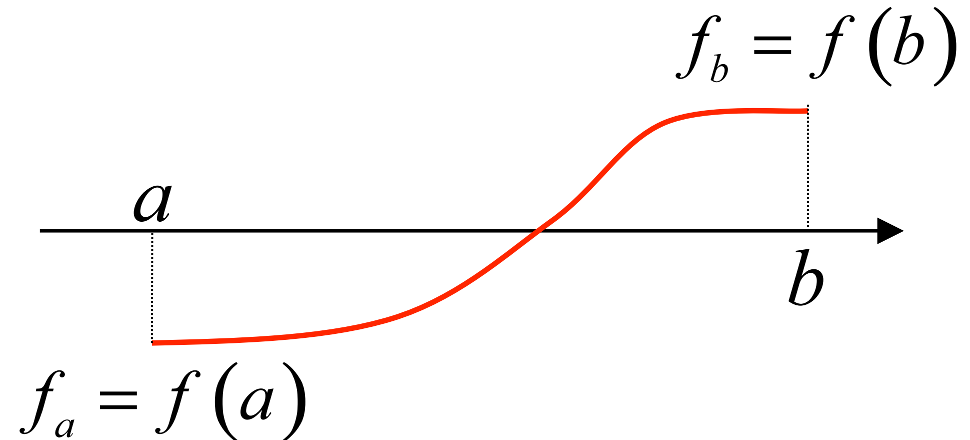
$$f\left(x\right)=0$$

- Υποθέστε πως μας ενδιαφέρουν μόνο τα πιο κάτω σενάρια



- Βρείτε ένα αλγόριθμο που να προσεγγίζει τη λύση

Bisection Method: Divide-and-Conquer




```
solve(f(),a,b)
   x=(a+b)/2;
   fx=f(x);
   if fx=0 return x;
   if fx*f(a)>0 return solve(f,x,b);
   return solve(f,a,x);
```

- Είναι ορθός ο αλγόριθμος;
- Πότε τερματίζει;

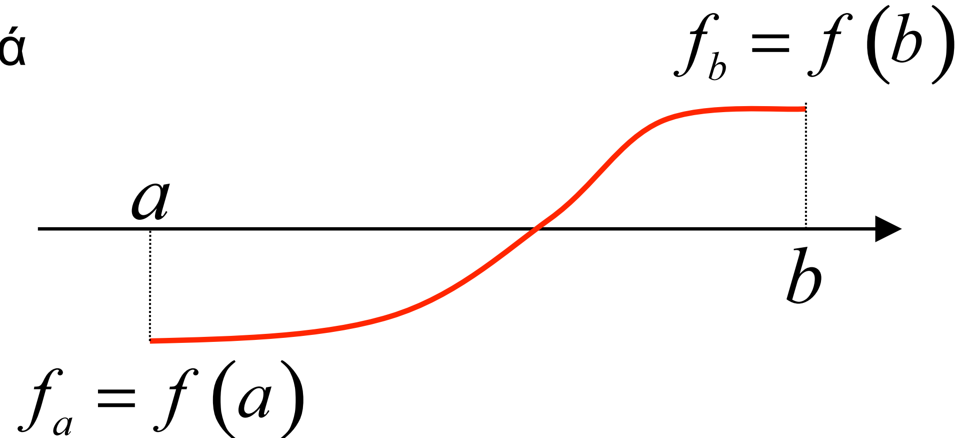
Bisection Method: Divide-and-Conquer

```
solve(f(),a,b,e,iter,N)
   x=(a+b)/2;
   if (b-a)<e or iter>N return x;
   fx=f(x);
   if fx=0 return x;
   if fx*f(a)>0 return solve(f,x,b,e,iter+1,N);
   return solve(f,a,x,e,iter+1,N);
```

- Πως μεταβάλλεται το μέγιστο δυνατό σφάλμα του αλγορίθμου από επανάληψη σε επανάληψη.

False Position Method: Divide-and-Conquer

- Εξίσωση της ευθείας που περνά από τα σημεία (a,f_a), (b,f_b):



- Υποψήφια λύση

- Στη συνέχεια ελέγχουμε τη τιμή της συνάρτησης στο σημείο x και συνεχίζουμε ανάλογα την αναζήτηση.
- Ο αλγόριθμος αυτός εμπίπτει στην κατηγορία μείωσης του προβλήματος αλλά κατά μεταβλητό μέγεθος (variable size decrease).
- Μπορείτε να σκεφτείτε «αντίστοιχο» τρόπο με τον οποίο να μεταβάλετε το αλγόριθμο αναζήτησης σε δυαδικά δέντρα (binary search);