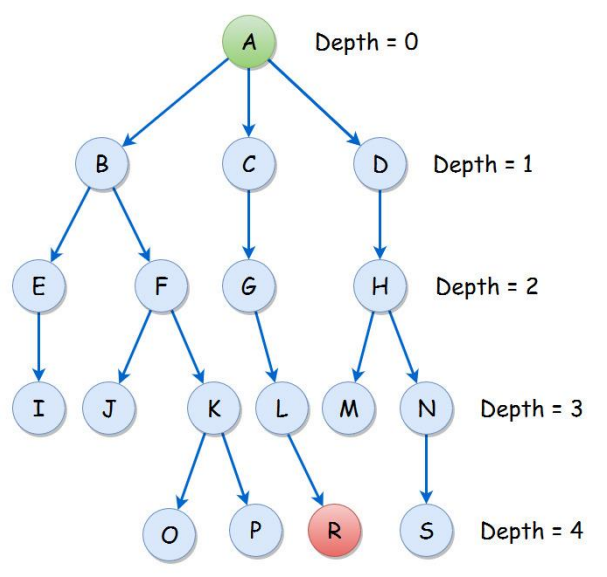
CN6005

Τεχνητή Νοημοσύνη

Lecture 2

Αλγόριθμοι Αναζήτησης



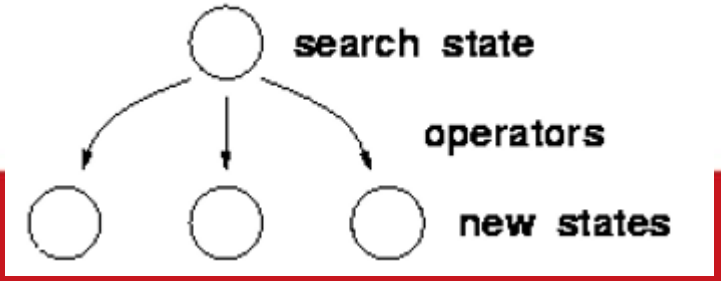
University of East London

METROPOLITAN COLLEGE

(ας θυμηθούμε...) Χώρος αναζήτησης ως δένδρο αναζήτησης

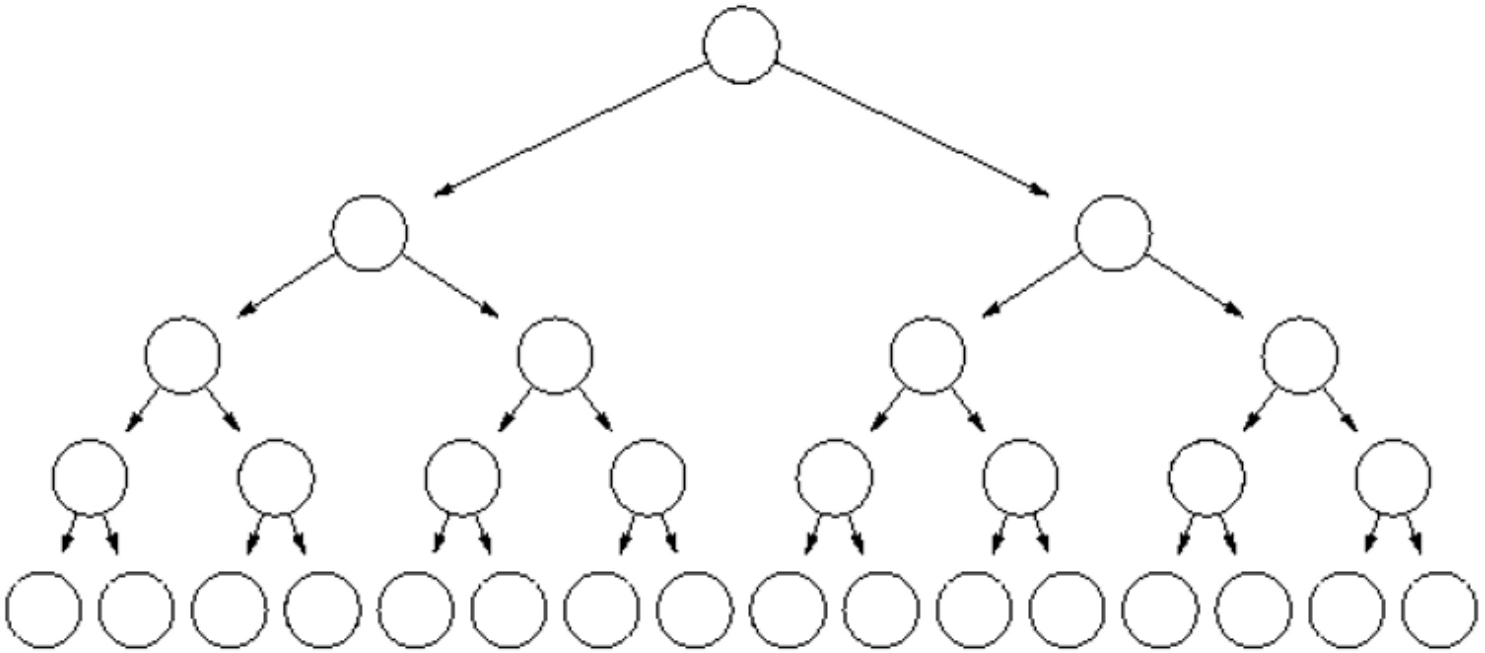
- Ο χώρος αναζήτησης μπορεί να αναπαρασταθεί με γράφο.

Τμήμα Δένδρου	**Αναπαράσταση**
Κόμβος (Node)	Κατάσταση
Ρίζα (Root)	Αρχική Κατάσταση
Φύλλο (Tip, Leaf) ή Τερματικός κόμβος	Τελική Κατάσταση ή Αδιέξοδο (Dead Node), δηλαδή κατάσταση στην οποία δεν μπορεί να εφαρμοστεί κανένας τελεστής μετάβασης.
Κλαδί (Branch)	Τελεστής Μετάβασης που μετατρέπει μια κατάσταση-Γονέα (Parent State) σε μία άλλη κατάσταση-Παιδί (Child State).
Λύση (Solution)	Μονοπάτι (Path) που ενώνει την αρχική με μία τελική κατάσταση
Επέκταση (Expansion)	Η διαδικασία παραγωγής όλων των καταστάσεων-παιδιών ενός κόμβου.
Παράγοντας Διακλάδωσης (Branching Factor)	Ο αριθμός των καταστάσεων-Παιδιών που προκύπτουν από την επέκταση μιας κατάστασης. Επειδή δεν είναι σταθερός αριθμός, αναφέρεται και ως Μέσος Παράγοντας Διακλάδωσης (Average Branching Factor).



Χώρος αναζήτησης ως δένδρο αναζήτησης

level	*states at this level*	*total states in tree*
0	1	1
1	2	3
2	4	7
3	8	15
4	16	31



Search Spaces

The number of states at a level is 2^n, where n is the level of the tree, and the total number of states up to and including level n is $2^{(n+1)-1}$. If there are more operators applicable to each state, the '2' in the above formulae is replaced by however many operators are applicable.

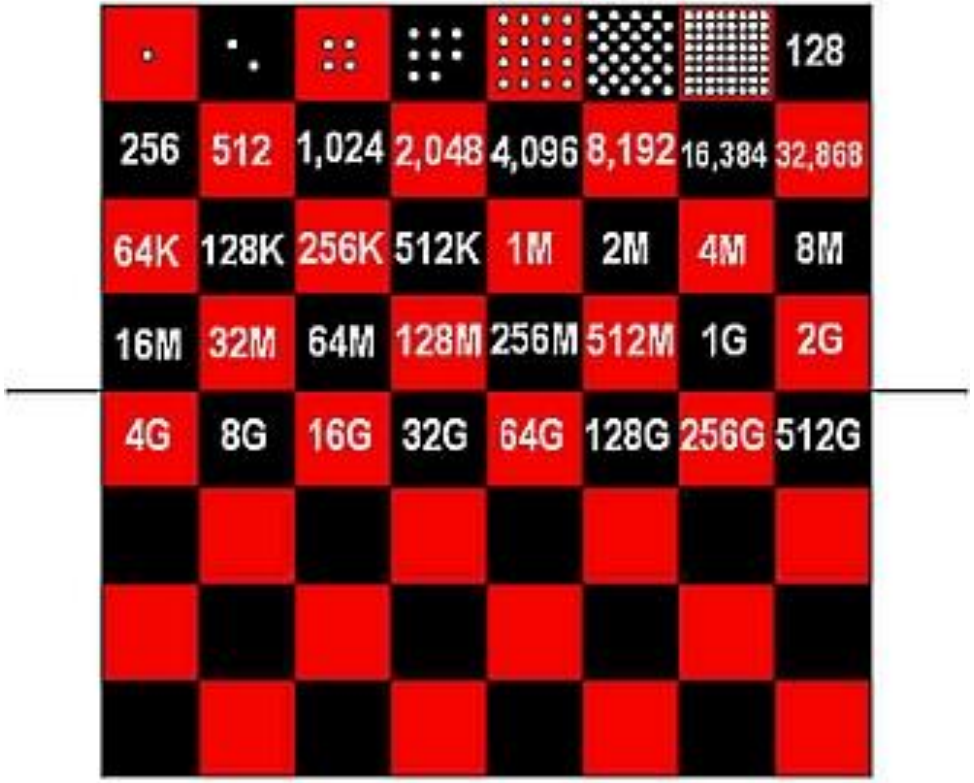
These formulae are important to remember because they give an idea of how hard the general problem-solving problem is. Here are some values:

N	2^n	2^(n+1) - 1
2	4	7
4	16	31
6	64	128
10	1024	2047
15	32768	65536
20	1048576	2097151
30	1073741824	2147483647

Note that this is for the case where only two operators apply. In general, a larger number of operators apply and so the number of states is even bigger. This sort of '**exponential growth**' is a general issue in the design of search programs, and since many AI programs involve search, it is an issue which is constantly important. Many of the techniques of AI programming involve figuring out how to manage these huge search spaces.

The calculation above does not take into account the fact that applying an operator to a state in the search space might yield a state that is identical to one already in the tree. For some domains this can make a big difference. In the Tower of Hanoi, for example, there are only 72 possible legal states. As we will see, for some search techniques it is important to keep track of which states have been seen before. For others, it is not worth the trouble, and the programs will do fine anyway.

Search Spaces – exponential growth

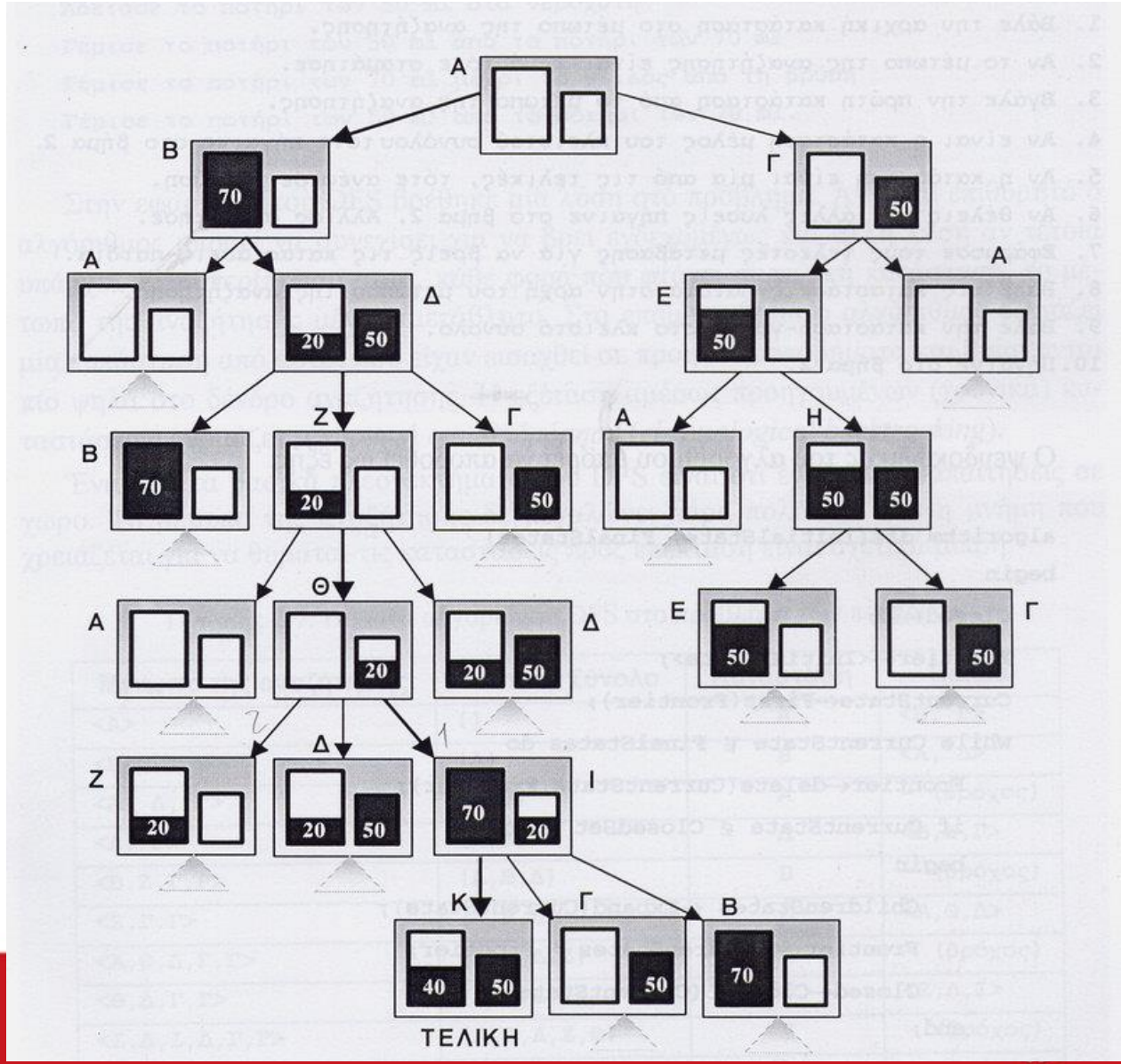


"...On the fortieth square the king would have had to put 1,000,000,000 grains of rice. And, finally on the sixty fourth square the king would have had to put more than 18,000,000,000,000,000,000 grains of rice which is equal to about 210 billion tons and is allegedly sufficient to cover the whole territory of India with a meter thick layer of rice. At ten grains of rice per square inch, the above amount requires rice fields covering twice the surface area of the Earth, oceans included..."

Γενικός Αλγόριθμος Αναζήτησης

1. Βάλε την αρχική κατάσταση στο **μέτωπο της αναζήτησης**.
2. Αν το μέτωπο αναζήτησης είναι άδειο τότε σταμάτησε.
3. Πάρε την πρώτη σε σειρά κατάσταση του μετώπου της αναζήτησης.
4. Αν είναι η κατάσταση αυτή μέρος του **κλειστού συνόλου** τότε πήγαινε στο βήμα 2.
5. Αν είναι η κατάσταση αυτή τελική κατάσταση τότε τύπωσε τη λύση και πήγαινε στο βήμα 2.
6. **Εφάρμοσε τους τελεστές μετάβασης** για να παράγεις τις καταστάσεις-παιδιά.
7. **Βάλε τις νέες καταστάσεις-παιδιά στο μέτωπο της αναζήτησης.**
8. **Κλάδεψε τις καταστάσεις** που δε χρειάζονται *(σύμφωνα με κάποιο κριτήριο)*, βγάζοντάς τες από το μέτωπο της αναζήτησης.
9. Κάνε αναδιάταξη στο μέτωπο της αναζήτησης *(σύμφωνα με κάποιο κριτήριο)*.
10. Βάλε την κατάσταση-γονέα στο κλειστό σύνολο.
11. Πήγαινε στο βήμα 2.

Πρόβλημα των ποτηριών



algorithm general(Initial, Final)

```
Begin
    Closed←∅;
    Frontier←{Initial};
    while (Frontier≠∅) do
       CurrentState←First(Frontier);
       Frontier←Delete(CurrentState,Frontier);
       if (CurrentState ∈ Final)
            return CurrentState;
       if (CurrentState ∉ Closed)
            Next←Expand(CurrentState);
            Frontier←Frontier ∪ Next;
            Frontier←prune(Frontier);
            Frontier←reorder(Frontier);
            Closed←Closed ∪ CurrentState;
    endwhile;
    return failure;
End.
```

Μέτωπο αναζήτησης

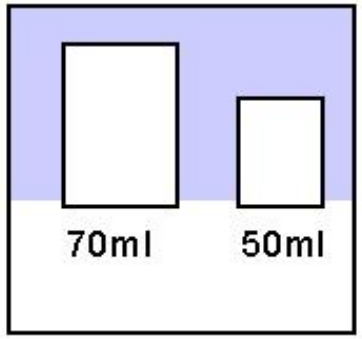
κλαδεύω

Εισαγωγή στους Τυφλούς Αλγορίθμους

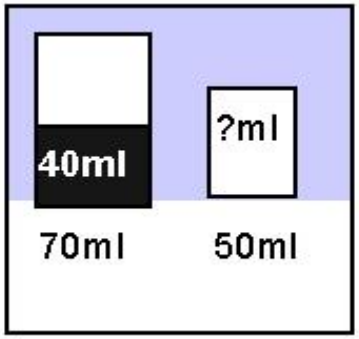
- Οι αλγόριθμοι **τυφλής αναζήτησης** (blind search algorithms) εφαρμόζονται όταν:
  - *Το πρόβλημα έχει μικρό χώρο αναζήτησης*
  - *δεν υπάρχει πληροφορία αξιολόγησης των καταστάσεων*
- Οι αλγόριθμοι αυτοί αντιμετωπίζουν με τον ίδιο ακριβώς τρόπο οποιοδήποτε πρόβλημα καλούνται να λύσουν
- Οι αλγόριθμοι είναι ανεξάρτητοι από το περιεχόμενο των καταστάσεων
  - Καθορίζουν μόνο την σειρά επέκτασης των καταστάσεων

Όνομα Αλγορίθμου	Συντομογραφία	Ελληνική Ορολογία
Depth-First Search	DFS	Αναζήτηση Πρώτα σε Βάθος
Breadth-First Search	BFS	Αναζήτηση Πρώτα σε Πλάτος
Iterative Deepening	ID	Επαναληπτική Εκβάθυνση
Bi-directional Search	BiS	Αναζήτηση Διπλής Κατεύθυνσης
Branch and Bound	B&B	Επέκταση και Οριοθέτηση
Beam Search	BS	Ακτινωτή Αναζήτηση

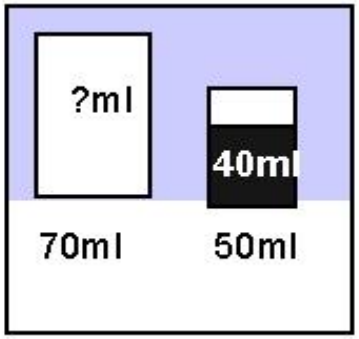
Παράδειγμα: Το πρόβλημα των ποτηριών



Αρχική Κατάσταση



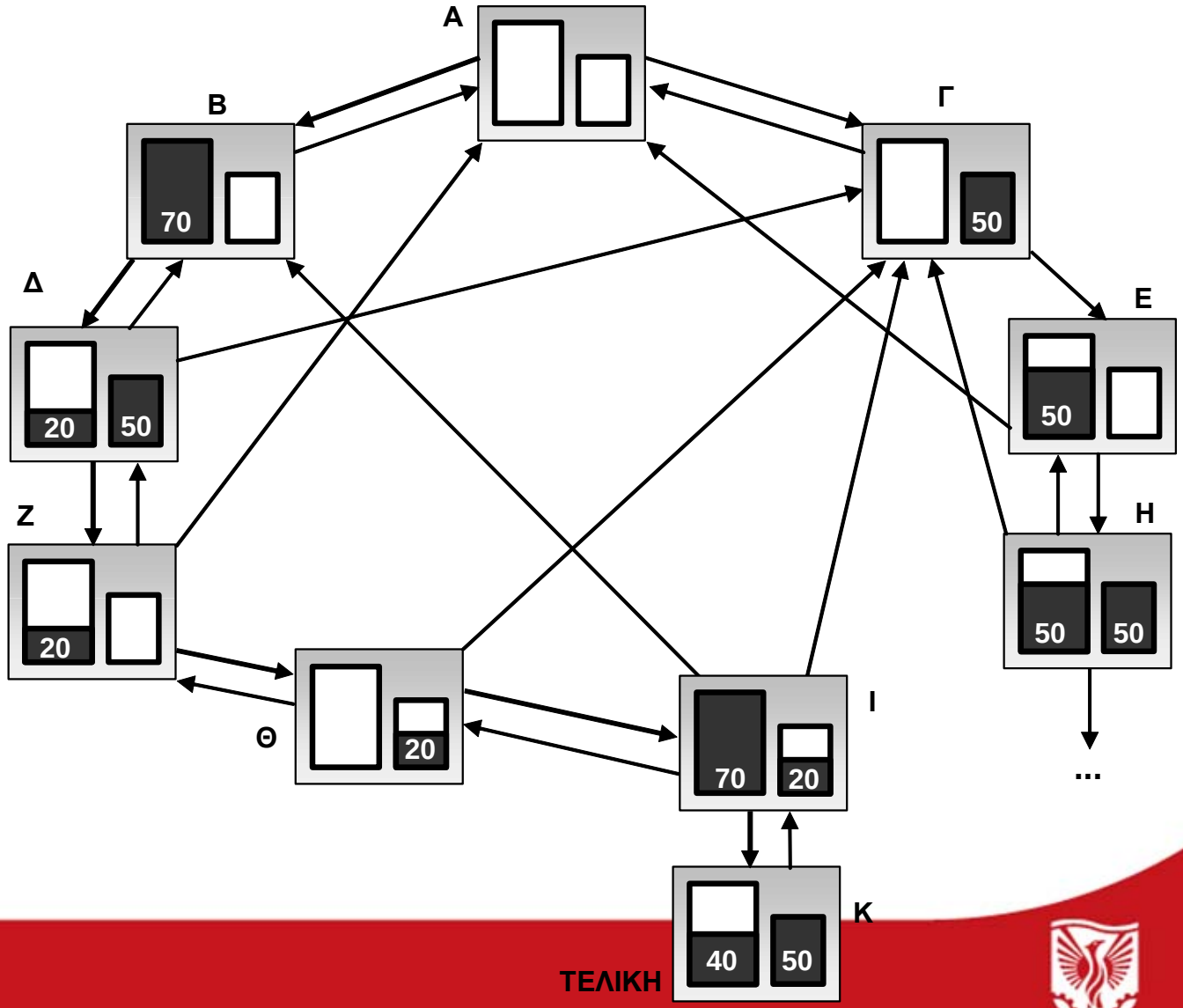
ή



Τελικές Καταστάσεις

Τελεστής (1): Γέμισε το ποτήρι των X ml μέχρι το χείλος από τη βρύση
Προϋποθέσεις
Το ποτήρι των X ml έχει 0 ml
Αποτελέσματα
Το ποτήρι των X ml έχει X ml
Τελεστής (2):Γέμισε το ποτήρι των X ml από το ποτήρι των Y ml
Προϋποθέσεις
Το ποτήρι των X ml έχει Z ml Το ποτήρι των Y ml έχει W ml
Αποτελέσματα
Το ποτήρι των X ml έχει X ml και Το ποτήρι των Y ml έχει W-(X-Z), αν W≥X-Z ή Το ποτήρι των X ml έχει Z+W ml και Το ποτήρι των Y ml έχει 0, αν W<X-Z
Τελεστής (3): Άδειασε το ποτήρι των X ml στο νεροχύτη
Προϋποθέσεις
Το ποτήρι των X ml έχει Z ml
Αποτελέσματα
Το ποτήρι των X ml έχει 0 ml

Παράδειγμα 1: Μέρος του Χώρου Αναζήτησης



Αναζήτηση πρώτα σε **βάθος**
Depth First Search (DFS)

University of East London

METROPOLITAN COLLEGE

Αναζήτηση Πρώτα σε Βάθος

- **Ο αλγόριθμος *πρώτα σε βάθος* (*DFS*) επιλέγει προς επέκταση την κατάσταση που βρίσκεται πιο βαθιά στο δένδρο**
  - Αν υπάρχουν πολλές καταστάσεις στο ίδιο βάθος επιλέγει αυθαίρετα μία
  - Ο αλγόριθμος εξετάζει το δένδρο σε κλαδιά

Ο αλγόριθμος DFS:

1. Βάλε την αρχική κατάσταση στο μέτωπο της αναζήτησης
2. Αν το μέτωπο της αναζήτησης είναι κενό τότε σταμάτησε
3. Βγάλε την 1η κατάσταση από το μέτωπο της αναζήτησης, έστω S
4. Αν η S ανήκει στο κλειστό σύνολο τότε πήγαινε στο βήμα 2
5. Αν η S είναι μία από τις τελικές, τότε ανέφερε τη λύση
6. Αν θέλεις άλλες λύσεις πήγαινε στο βήμα 2 Αλλιώς σταμάτησε
7. Εφάρμοσε τους τελεστές για να βρεις τις καταστάσεις-παιδιά
8. Βάλε τα παιδιά στην αρχή του μετώπου της αναζήτησης
9. Βάλε την S στο κλειστό σύνολο
10. Πήγαινε στο βήμα 2

algorithm DFS(Initial, Final)

```
algorithm dfs(InitialState, FinalStates)
begin
  Closed←∅;
  Frontier←<InitialState>;
  CurrentState←First(Frontier);
  while CurrentState ∉ FinalStates do
    Frontier←delete(CurrentState,Frontier);
    if CurrentState ∉ ClosedSet then
    begin
      ChildrenStates ←Expand(CurrentState);
      Frontier←ChildrenStates ^ Frontier;
      Closed←Closed∪{CurrentState};
    end;
    if Frontier=∅ then exit;
    CurrentState←First(Frontier);
  endwhile;
return success;
 end.
```

DFS - Σχόλια

- Το μέτωπο της αναζήτησης είναι μια **δομή στοίβας** (Stack LIFO)
- **Για να εξετάσει μία κατάσταση θα πρέπει πρώτα να έχει ολοκληρώσει την εξέταση όλων των καταστάσεων που βρίσκονται στα "*αριστερότερα*" κλαδιά.**
- **Πλεονεκτήματα**:
  - *Μικρές απαιτήσεις σε χώρο – Το μέτωπο της αναζήτησης δεν περιέχει μεγάλο αριθμό καταστάσεων*
- **Μειονεκτήματα**:
  - *Δεν εγγυάται ότι η πρώτη λύση που θα βρεθεί είναι η βέλτιστη*
  - *Εν γένει θεωρείται **μη-πλήρης** (αν δεν υπάρχει έλεγχος βρόχων ή αν ο χώρος αναζήτησης είναι μη πεπερασμένος γιατί μπορεί να μπλεχτεί σε κλαδιά άπειρου μήκους) (δείτε επόμενη διαφάνεια για σχετική ορολογία)*
    - *Αν ο χώρος αναζήτησης είναι πεπερασμένος και χρησιμοποιείται κλειστό σύνολο, ο DFS είναι πλήρης*

Χαρακτηριστικά αλγορίθμων αναζήτησης

- Ένας αλγόριθμος ονομάζεται **εξαντλητικός** *(exhaustive)* όταν το σύνολο των καταστάσεων που εξετάζει ο αλγόριθμος για να βρει τις απαιτούμενες λύσεις είναι ίσο με το χώρο αναζήτησης, δηλαδή V=SP.

- Ένας αλγόριθμος δεν λύνει πάντα κάποιο πρόβλημα, έστω και αν υπάρχει κάποια λύση. Τότε τα σύνολα G_s και F είναι κενά.

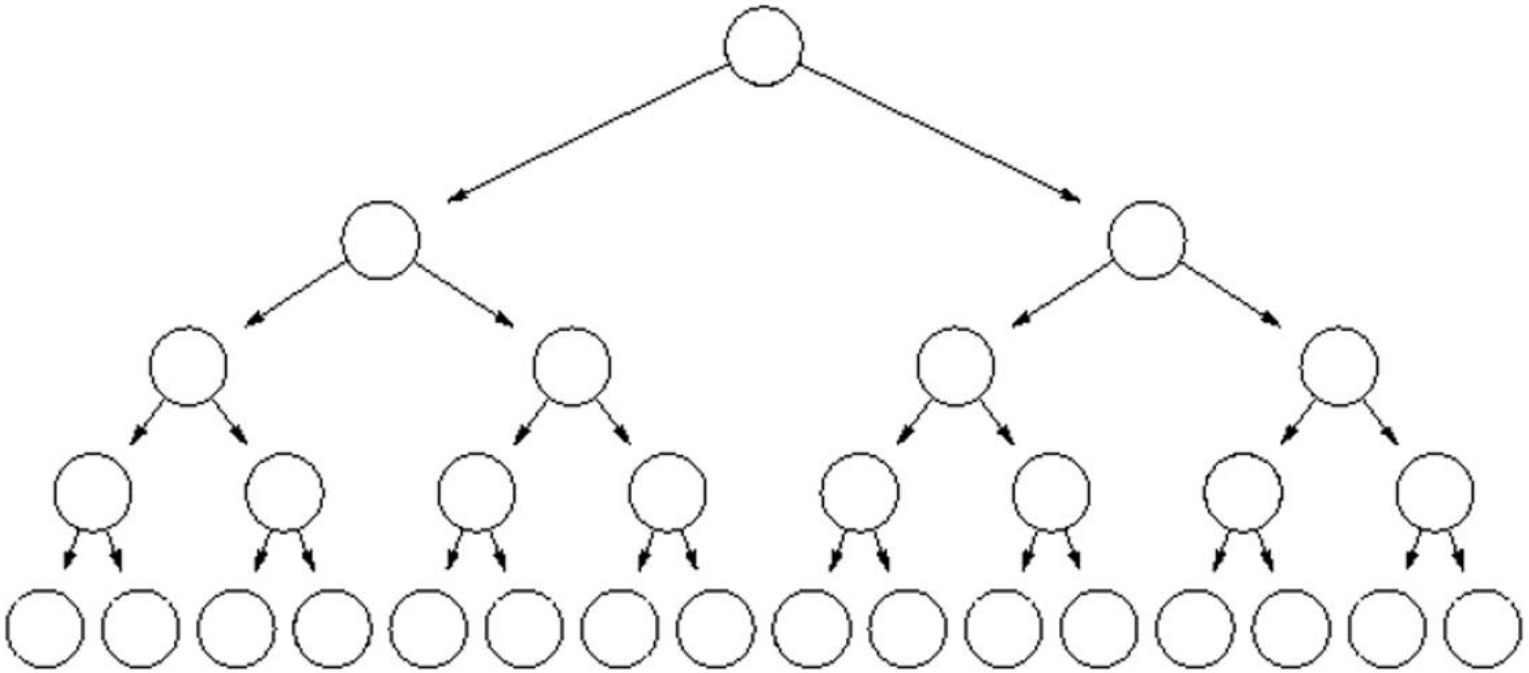
- Ένας αλγόριθμος αναζήτησης ονομάζεται **πλήρης** *(complete)* αν εγγυάται ότι θα βρει μία λύση για οποιαδήποτε τελική κατάσταση, αν τέτοια λύση υπάρχει. Σε αντίθετη περίπτωση, ο αλγόριθμος ονομάζεται μη-πλήρης *(incomplete)*.

- Μία λύση ονομάζεται **βέλτιστη** *(optimal)* αν οδηγεί στην καλύτερη, σύμφωνα με τη διάταξη, τελική κατάσταση. Όταν δεν υπάρχει διάταξη, μία λύση ονομάζεται βέλτιστη αν είναι η συντομότερη (shortest).

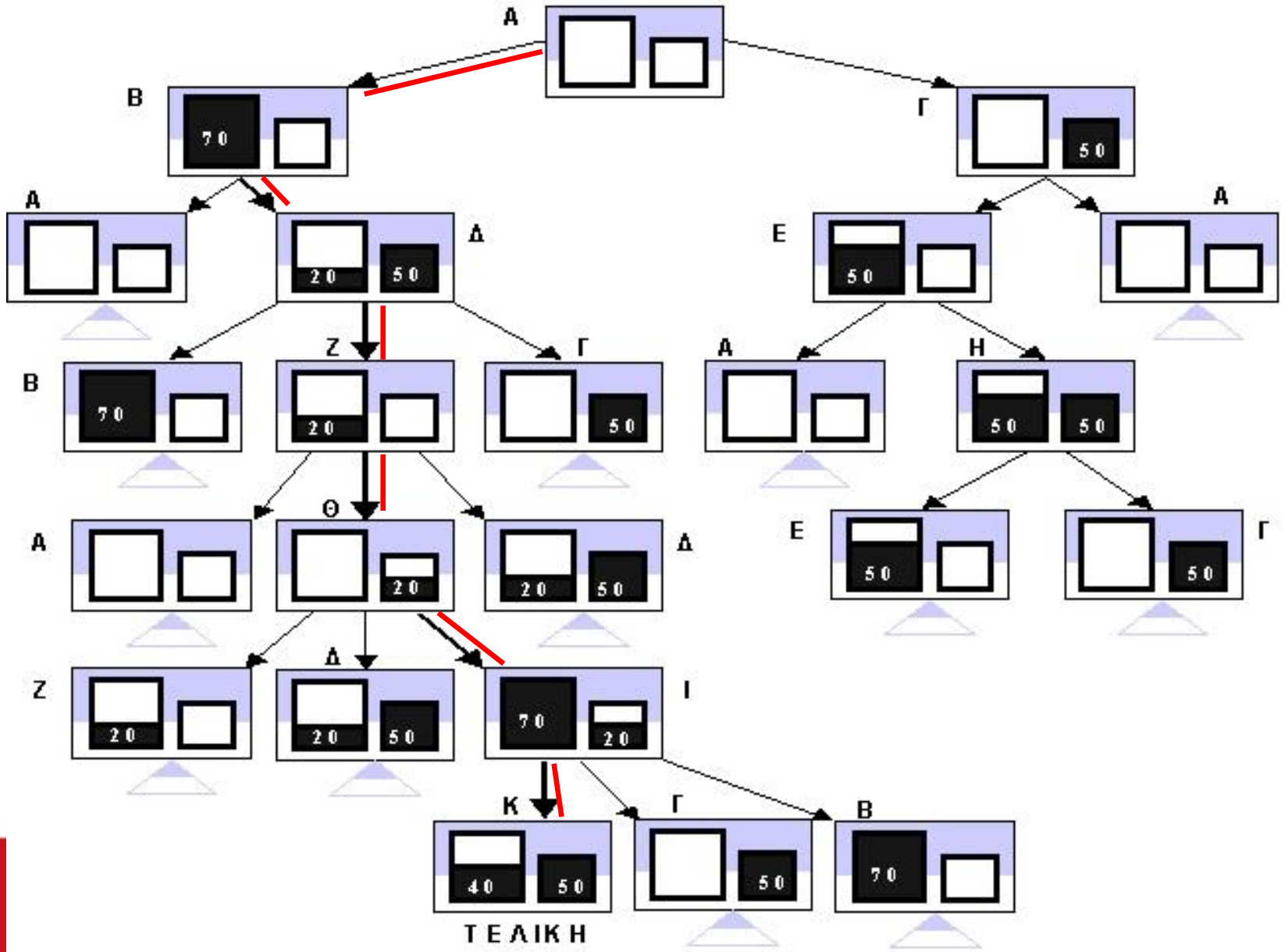
- Ένας αλγόριθμος αναζήτησης καλείται **αποδεκτός** *(admissible)* αν εγγυάται ότι θα βρει τη βέλτιστη λύση, αν μια τέτοια λύση υπάρχει.

όπου: ***F*** *είναι το σύνολο των λύσεων που βρέθηκαν.*
Gs *είναι το σύνολο των τελικών καταστάσεων που εξετάστηκαν.*
V *είναι το σύνολο των καταστάσεων που εξέτασε ο αλγόριθμος αναζήτησης.*
SP *είναι ο χώρος αναζήτησης*

DFS - Σχόλια



Παράδειγμα 1 – Δένδρο Αναζήτησης



Παράδειγμα 1 – Εφαρμογή DFS

Μέτωπο αναζήτησης	Κλειστό Σύνολο	Κατάσταση	Παιδιά
<Α>	{}	Α	<Β, Γ>
<Β, Γ>	{Α}	Β	<Α, Δ>
<Α, Δ, Γ>	{Α,Β}	Α	- (βρόχος)
<Δ, Γ>	{Α,Β}	Δ	<Β,Ζ,Γ>
?	?	?	?

Παράδειγμα 1 – Εφαρμογή DFS

Μέτωπο αναζήτησης	Κλειστό Σύνολο	Κατάσταση	Παιδιά
<Α>	{}	Α	<Β, Γ>
<Β, Γ>	{Α}	Β	<Α, Δ>
<Α, Δ, Γ>	{Α,Β}	Α	- (βρόχος)
<Δ, Γ>	{Α,Β}	Δ	<Β,Ζ,Γ>
<Β,Ζ,Γ,Γ>	{Α,Β,Δ}	Β	- (βρόχος)
<Ζ,Γ,Γ>	{Α,Β,Δ}	Ζ	<Α,Θ,Δ>
<Α,Θ,Δ,Γ,Γ>	{Α,Β,Δ,Ζ}	Α	- (βρόχος)
<Θ,Δ,Γ,Γ>	{Α,Β,Δ,Ζ}	Θ	<Ζ,Δ,Ι>
<Ζ,Δ,Ι,Δ,Γ,Γ>	{Α,Β,Δ,Ζ,Θ}	Ζ	- (βρόχος)
<Δ,Ι,Δ,Γ,Γ>	{Α,Β,Δ,Ζ,Θ}	Δ	- (βρόχος)
<Ι,Δ,Γ,Γ>	{Α,Β,Δ,Ζ,Θ}	Ι	<Κ,Γ,Β>
<Κ,Γ,Β,Δ,Γ,Γ>	{Α,Β,Δ,Ζ,Θ,Ι}	Κ	ΤΕΛΙΚΗ

Αναζήτηση πρώτα σε **πλάτος**
Breadth First Search (BFS)

University of East London

METROPOLITAN COLLEGE

Αναζήτηση Πρώτα σε Πλάτος

- **Ο αλγόριθμος *πρώτα σε πλάτος* (*BFS*) επιλέγει προς επέκταση την κατάσταση που βρίσκεται πιο κοντά στη ρίζα του δένδρου**
  - Αν υπάρχουν πολλές καταστάσεις στο ίδιο βάθος επιλέγει αυθαίρετα μία
  - Ο αλγόριθμος εξετάζει το δένδρο σε επίπεδα

Ο αλγόριθμος BFS:

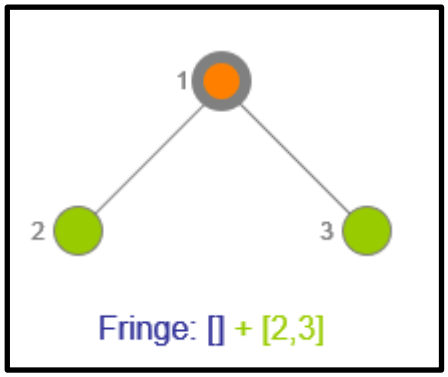
1. Βάλε την αρχική κατάσταση στο μέτωπο της αναζήτησης
2. Αν το μέτωπο της αναζήτησης είναι κενό τότε σταμάτησε
3. Βγάλε την 1η κατάσταση από το μέτωπο της αναζήτησης, έστω S
4. Αν η S ανήκει στο κλειστό σύνολο τότε πήγαινε στο βήμα 2
5. Αν η S είναι μία από τις τελικές, τότε ανέφερε τη λύση
6. Αν θέλεις άλλες λύσεις πήγαινε στο βήμα 2 Αλλιώς σταμάτησε
7. Εφάρμοσε τους τελεστές για να βρεις τις καταστάσεις-παιδιά
8. Βάλε τα παιδιά στο τέλος του μετώπου της αναζήτησης
9. Βάλε την S στο κλειστό σύνολο
10. Πήγαινε στο βήμα 2

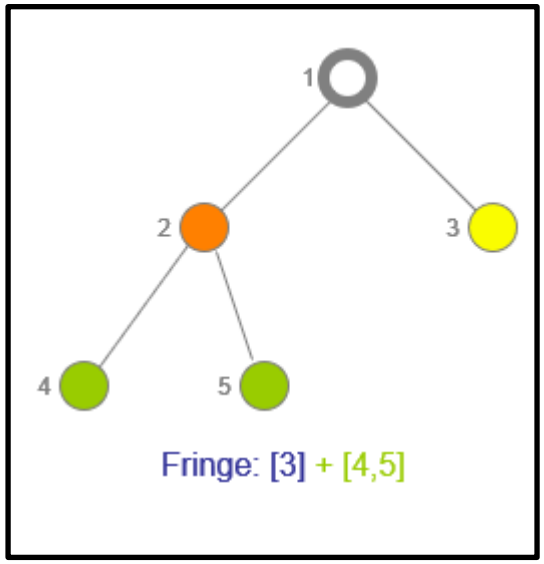
Αναζήτηση Πρώτα σε Πλάτος

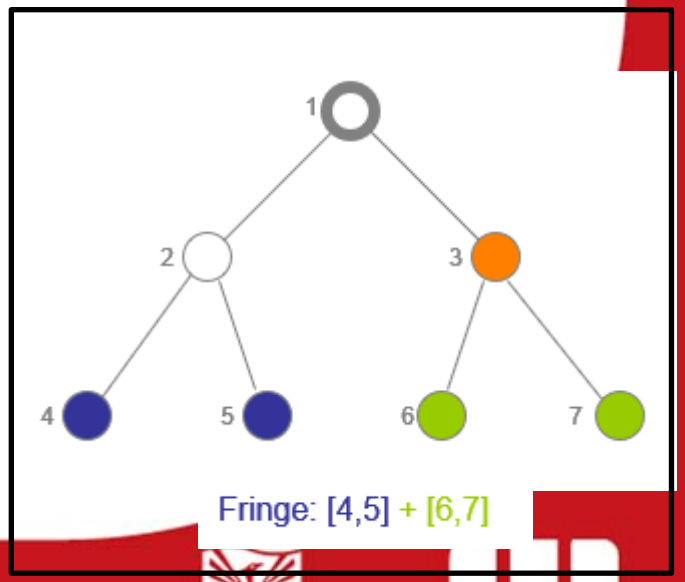
```
algorithm bfs(InitialState, FinalStates)
begin
Closed←∅;
Frontier←<InitialState>;
CurrentState←First(Frontier);
while CurrentState ∉ FinalStates do
    Frontier←delete(CurrentState,Frontier);
    if CurrentState ∉ ClosedSet
    begin
      ChildrenStates ←Expand(CurrentState);
      Frontier← Frontier ^ ChildrenStates;
      Closed←Closed∪{CurrentState};
    end;
  if Frontier= ∅ then exit;
  CurrentState←First(Frontier);
endwhile;
return success;
end.
```

algorithm BFS (στιγμιότυπα)

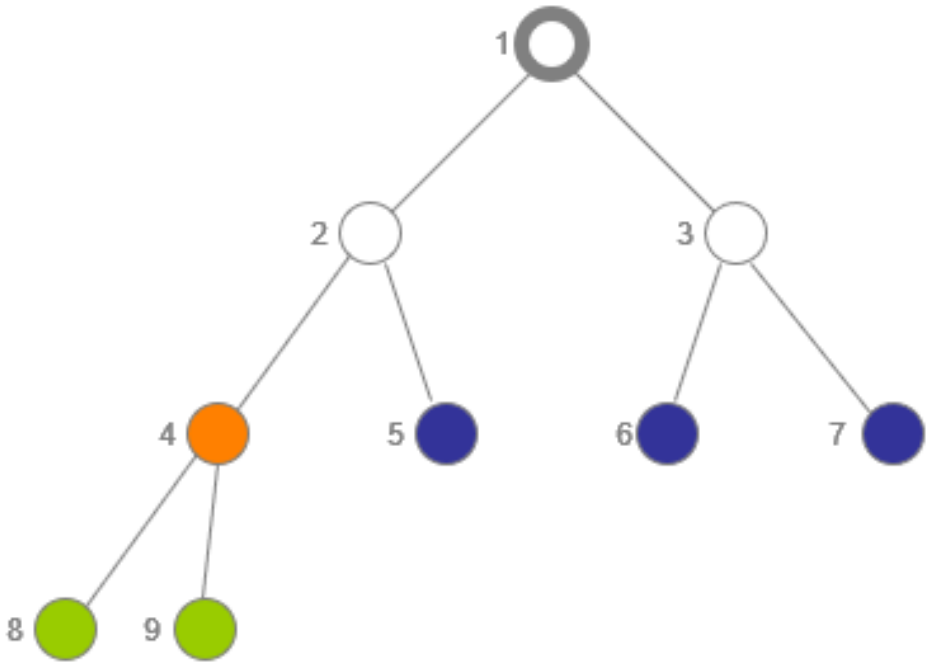
ΤΥΧΑΙΟ ΠΑΡΑΔΕΙΓΜΑ





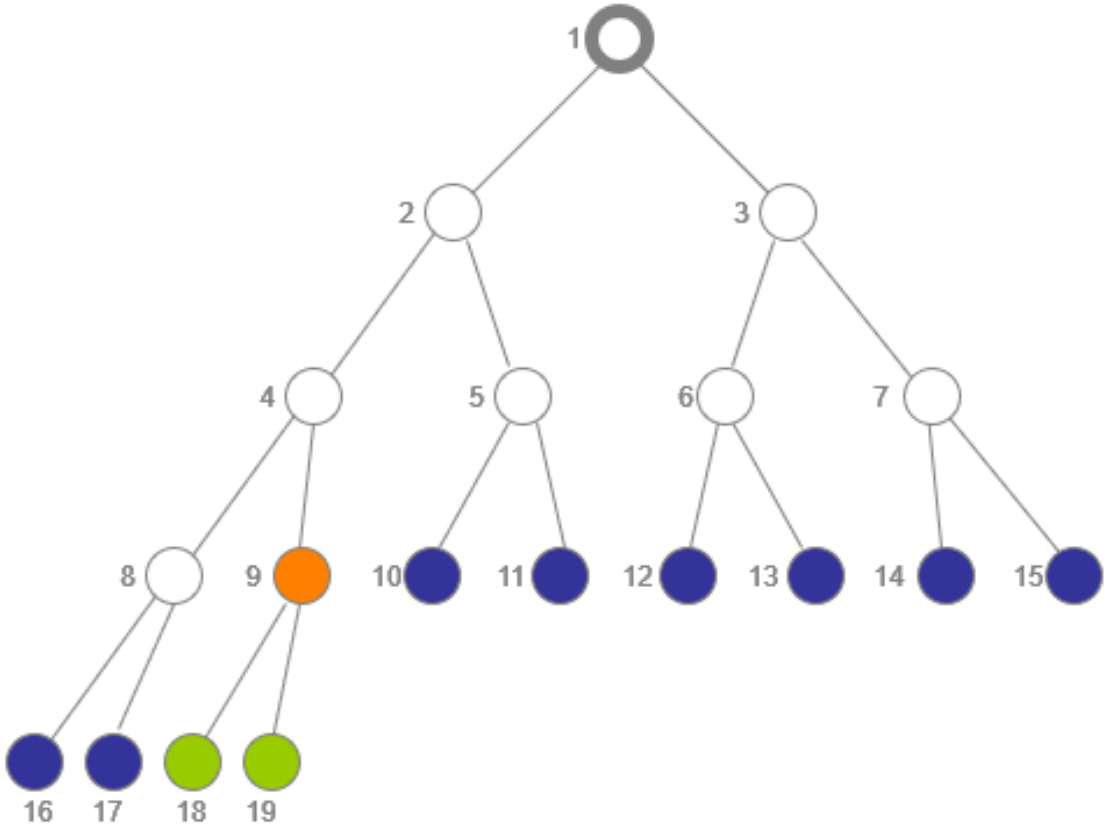


algorithm BFS (στιγμιότυπα)



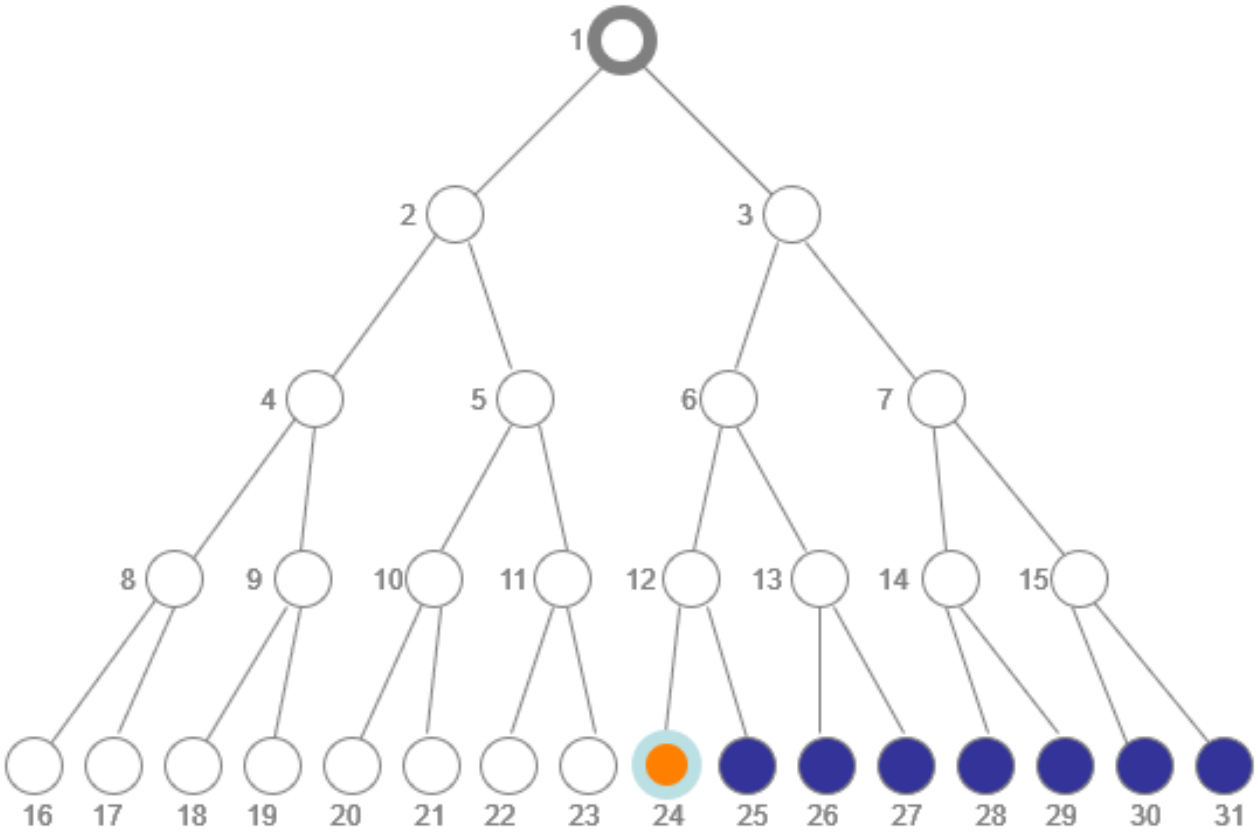
Fringe: [5,6,7] + [8,9]

algorithm BFS (στιγμιότυπα)



Fringe: [10,11,12,13,14,15,16,17] + [18,19]

algorithm BFS (στιγμιότυπα)



Fringe: [25,26,27,28,29,30,31]

Ο στόχος βρέθηκε

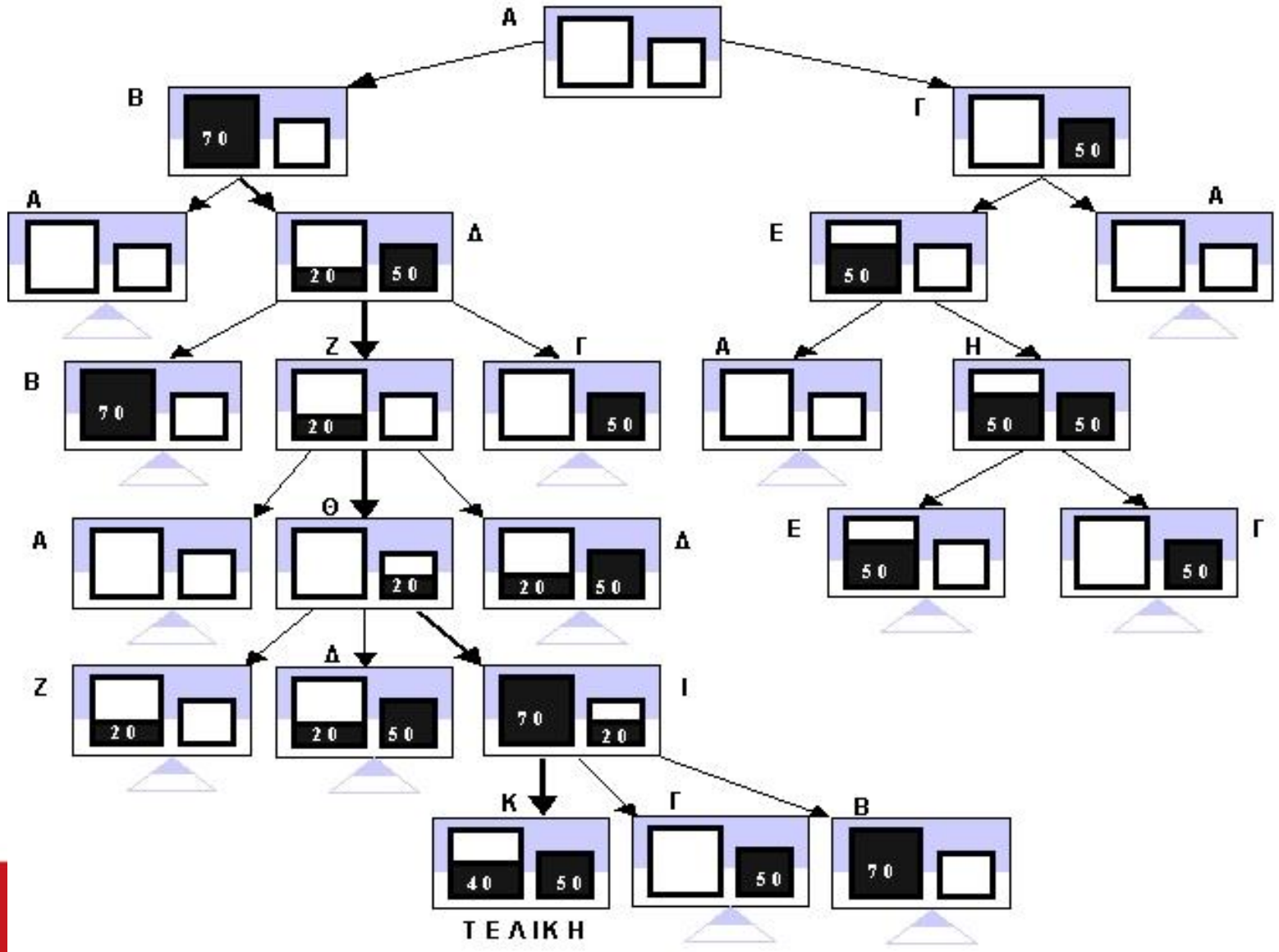
University of East London

METROPOLITAN COLLEGE

BFS - Σχόλια

- Το μέτωπο της αναζήτησης είναι μια **δομή ουράς** (Queue FIFO)
- Για να εξετάσει μία κατάσταση θα πρέπει πρώτα να έχει ολοκληρώσει την εξέταση όλων των καταστάσεων που βρίσκονται σε "υψηλότερα" επίπεδα
- **Πλεονεκτήματα**:
  - *Είναι **πλήρης***
  - Είναι αποδεκτός – βρίσκει πάντα την βέλτιστη λύση
- **Μειονεκτήματα**:
  - *Έχει πολύ μεγάλες απαιτήσεις σε μνήμη*
    - *Το μέτωπο αναζήτησης περιλαμβάνει κατά μέσο όρο όλες τις καταστάσεις ενός επιπέδου (συνδυαστική έκρηξη)*
    - *Σε πρακτικά προβλήματα τερματίζει λόγω έλλειψης μνήμης*

Παράδειγμα 1 – Δένδρο Αναζήτησης



Εφαρμογή BFS

Μέτωπο αναζήτησης	Κλειστό Σύνολο	Κατάσταση	Παιδιά

Δοκιμάστε να δημιουργήσετε μόνοι τον πίνακα (η λύση στην επόμενη διαφάνεια)…

Εφαρμογή BFS (1/2)

Μέτωπο αναζήτησης	Κλειστό Σύνολο	Κατάσταση	Παιδιά
<Α>	{}	Α	<Β, Γ>
<Β, Γ>	{Α}	Β	<Α, Δ>
<Γ,Α,Δ>	{Α,Β}	Γ	<Ε,Α>
<Α,Δ,Ε,Α>	{Α,Β,Γ}	Α	- (βρόχος)
<Δ,Ε,Α>	{Α,Β,Γ}	Δ	<Β,Ζ,Γ>
<Ε,Α,Β,Ζ,Γ>	{Α,Β,Γ,Δ}	Ε	<Α,Η>
<Α,Β,Ζ,Γ,Α,Η>	{Α,Β,Γ,Δ,Ε}	Α	- (βρόχος)
<Β,Ζ,Γ,Α,Η>	{Α,Β,Γ,Δ,Ε}	Β	- (βρόχος)
<Ζ,Γ,Α,Η>	{Α,Β,Γ,Δ,Ε}	Ζ	<Α,Θ,Δ>
<Γ,Α,Η,Α,Θ,Δ>	{Α,Β,Γ,Δ,Ε,Ζ}	Γ	- (βρόχος)
<Α,Η,Α,Θ,Δ>	{Α,Β,Γ,Δ,Ε,Ζ}	Α	- (βρόχος)
<Η,Α,Θ,Δ>	{Α,Β,Γ,Δ,Ε,Ζ}	Η	<Ε,Γ>
<Α,Θ,Δ,Ε,Γ>	{Α,Β,Γ,Δ,Ε,Ζ,Η}	Α	- (βρόχος)
<Θ,Δ,Ε,Γ>	{Α,Β,Γ,Δ,Ε,Ζ,Η}	Θ	<Ζ,Δ,Ι>

Εφαρμογή BFS (2/2)

Μέτωπο αναζήτησης	Κλειστό Σύνολο	Κατάσταση	Παιδιά
<Δ,Ε,Γ,Ζ,Δ,Ι>	{Α,Β,Γ,Δ,Ε,Ζ,Η}	Δ	- (βρόχος)
<Ε,Γ,Ζ,Δ,Ι>	{Α,Β,Γ,Δ,Ε,Ζ,Η}	Ε	- (βρόχος)
<Γ,Ζ,Δ,Ι>	{Α,Β,Γ,Δ,Ε,Ζ,Η}	Γ	- (βρόχος)
<Ζ,Δ,Ι>	{Α,Β,Γ,Δ,Ε,Ζ,Η}	Ζ	- (βρόχος)
<Δ,Ι>	{Α,Β,Γ,Δ,Ε,Ζ,Η}	Δ	- (βρόχος)
<Ι>	{Α,Β,Γ,Δ,Ε,Ζ,Η}	Ι	<Κ,Γ,Β>
<Κ,Γ,Β>	{Α,Β,Γ,Δ,Ε,Ζ,Η,Ι}	Κ	ΤΕΛΙΚΗ