Κεφάλαιο 3

Αλγόριθμοι Τυφλής Αναζήτησης

Τεχνητή Νοημοσύνη - Β' Έκδοση

Ι. Βλαχάβας, Π. Κεφαλάς, Ν. Βασιλειάδης, Φ. Κόκκορας, Η. Σακελλαρίου

Αλγόριθμοι Τυφλής Αναζήτησης

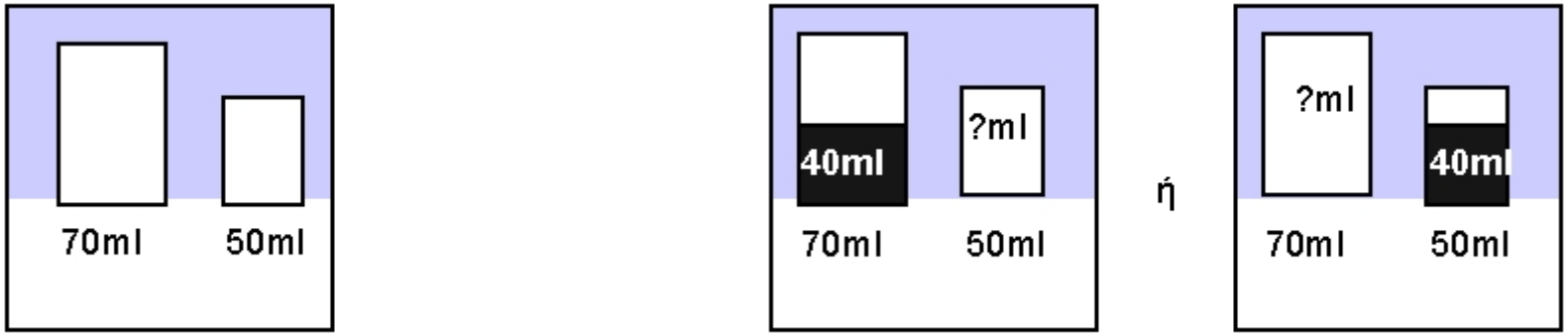
- Οι *αλγόριθμοι τυφλής αναζήτησης* (*blind search algorithms*) εφαρμόζονται σε προβλήματα στα οποία δεν υπάρχει πληροφορία που να επιτρέπει αξιολόγηση των καταστάσεων.

- Στους αλγορίθμους τυφλής αναζήτησης έχει σημασία η χρονική σειρά με την οποία παράγονται οι καταστάσεις από το μηχανισμό επέκτασης.

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 (1/2)



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

Παράδειγμα

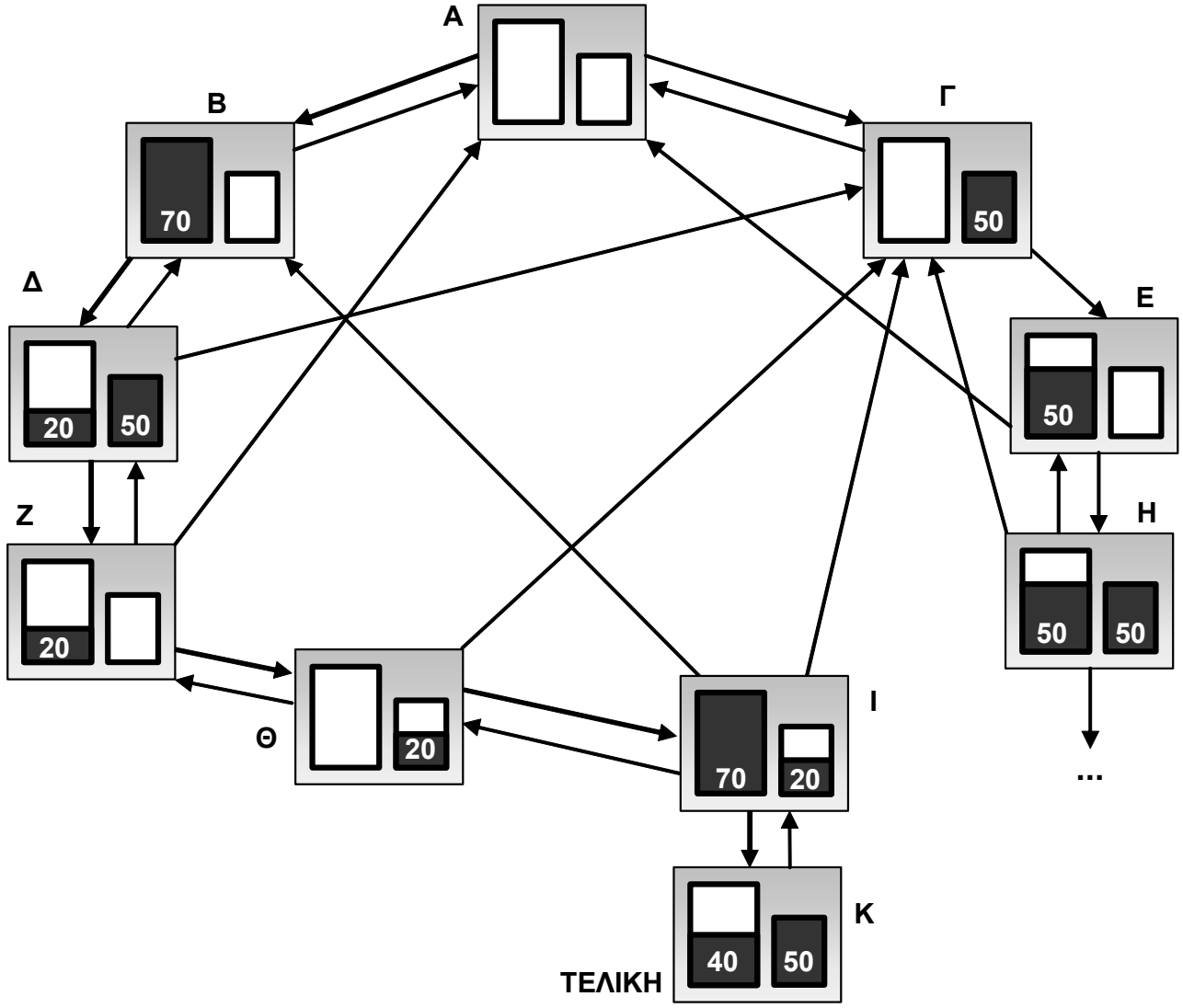
Το πρόβλημα των ποτηριών (2/2)

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 (W≠0)
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 έχει περιεχόμενο
Αποτελέσματα
Το ποτήρι των X ml έχει 0 ml

Μέρος του χώρου αναζήτησης

Στο πρόβλημα με τα ποτήρια



Αναζήτηση Πρώτα σε Βάθος

Ο αλγόριθμος *πρώτα σε βάθος* (*Depth-First Search - DFS*) επιλέγει προς επέκταση την κατάσταση που βρίσκεται πιο βαθιά στο δένδρο.

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

Ο αλγόριθμος DFS (Ψευδοκώδικας)

```
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.
```

Αναζήτηση Πρώτα σε Βάθος

Σχόλια

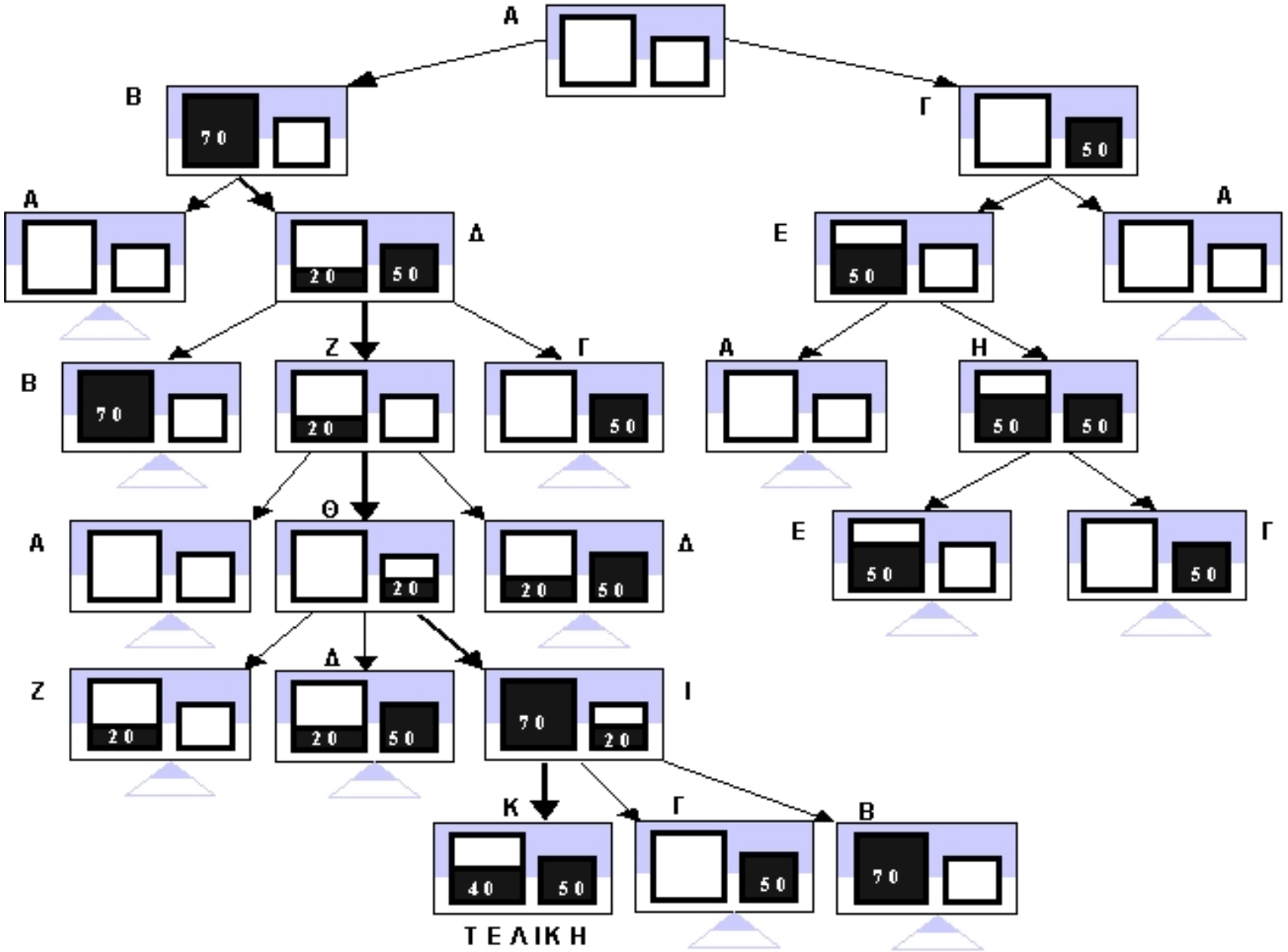
- Το μέτωπο της αναζήτησης είναι μια δομή στοίβας (Stack LIFO)
- Η εξέταση αμέσως προηγουμένων (χρονικά) καταστάσεων ονομάζεται χρονική οπισθοδρόμηση (*chronological backtracking*).

- Πλεονεκτήματα:
  - *το μέτωπο της αναζήτησης δε μεγαλώνει πάρα πολύ.*

- Μειονεκτήματα:
  - *Δεν εγγυάται ότι η πρώτη λύση που θα βρεθεί είναι η βέλτιστη.*
  - *Θεωρείται μη-πλήρης.*
  - *Όταν όμως ο χώρος αναζήτησης είναι πεπερασμένος και χρησιμοποιείται κλειστό σύνολο, ο DFS θα βρει λύση, εάν μια τέτοια υπάρχει.*

Αναζήτηση Πρώτα σε Βάθος (DFS)

Δένδρο αναζήτησης στο πρόβλημα των ποτηριών



Αναζήτηση Πρώτα σε Βάθος (DFS)

Πρόβλημα των ποτηριών

Μέτωπο της 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

Ο αλγόριθμος *αναζήτησης πρώτα σε πλάτος* (*Breadth First Search - BFS*) εξετάζει πρώτα όλες τις καταστάσεις που βρίσκονται στο ίδιο βάθος και μετά συνεχίζει στην επέκταση καταστάσεων στο αμέσως επόμενο επίπεδο.

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 πρώτη κατάσταση από το μέτωπο της αναζήτησης.
4. Αν είναι η κατάσταση ανήκει στο κλειστό σύνολο τότε πήγαινε στο βήμα 2.
5. Αν η κατάσταση είναι μία τελική τότε ανέφερε τη λύση.
6. Αν θέλεις και άλλες λύσεις πήγαινε στο βήμα 2. Αλλιώς σταμάτησε.
7. Εφάρμοσε τους τελεστές μεταφοράς για να βρεις τις καταστάσεις-παιδιά.
8. Βάλε τις καταστάσεις-παιδιά στο τέλος του μετώπου της αναζήτησης.
9. Βάλε την κατάσταση-γονέα στο κλειστό σύνολο.
10. Πήγαινε στο βήμα 2.

Ο αλγόριθμος BFS (Ψευδοκώδικας)

```
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.
```

Αναζήτηση Πρώτα σε Πλάτος

Σχόλια

- Το μέτωπο της αναζήτησης είναι μια δομή ουράς (Queue FIFO).
- Πλεονεκτήματα:
  - *Βρίσκει πάντα την καλύτερη λύση (μικρότερη σε μήκος).*
  - *Είναι πλήρης.*
- Μειονεκτήματα:
  - *Το μέτωπο της αναζήτησης μεγαλώνει πολύ σε μέγεθος.*

Αναζήτηση Πρώτα σε Πλάτος (BFS)

Πρόβλημα των ποτηριών (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>

Αναζήτηση Πρώτα σε Πλάτος (BFS)

Πρόβλημα των ποτηριών (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

Αλγόριθμος Επαναληπτικής Εκβάθυνσης

Ο αλγόριθμος επαναληπτικής εκβάθυνσης (Iterative Deepening - ID) συνδυάζει με τον καλύτερο τρόπο τους DFS και BFS.

Ο αλγόριθμος ID:

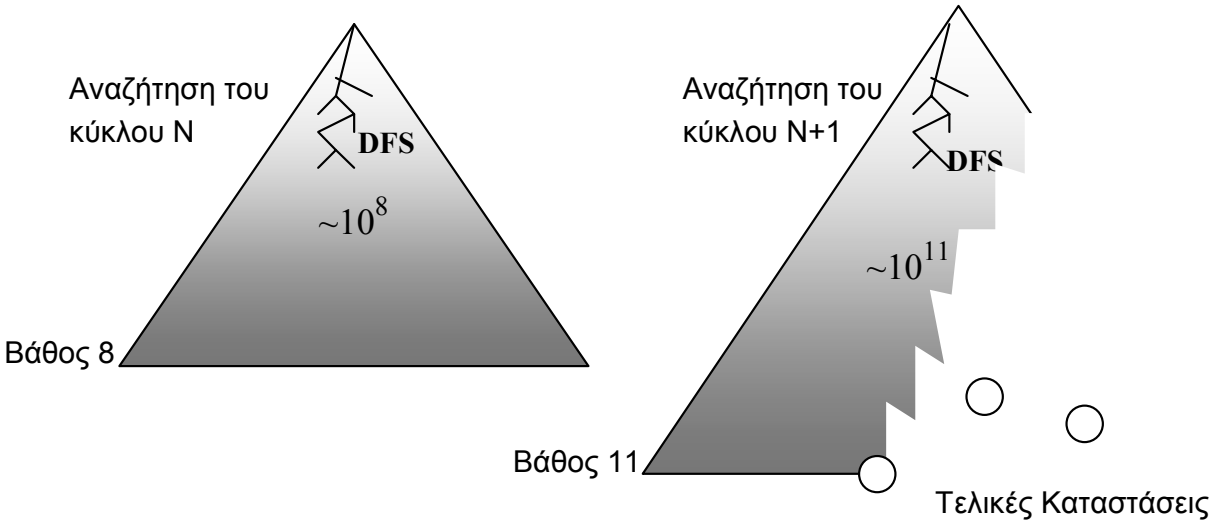
1. Όρισε το αρχικό βάθος αναζήτησης (συνήθως 1).
2. Εφάρμοσε τον αλγόριθμο DFS μέχρι αυτό το βάθος αναζήτησης.
3. Αν έχεις βρει λύση σταμάτησε.
4. Αύξησε το βάθος αναζήτησης (συνήθως κατά 1).
5. Πήγαινε στο βήμα 2.

Ο αλγόριθμος ID (Ψευδοκώδικας)

```
algorithm id(InitialState, FinalStates)
begin
  depth←1
  while solution is not found do
    bounded_dfs(InitialState,FinalStates,depth);
    depth←depth+1
  endwhile;
end.
```

Αναζήτηση ID
Σχόλια

- Μειονεκτήματα:
  - *Δε θυμάται τίποτα από τον προηγούμενο κύκλο αναζήτησης.*
- Πλεονεκτήματα:
  - *Είναι πλήρης.*
  - *Αν το βάθος αυξάνεται κατά 1 σε κάθε κύκλο και ο ID βρει λύση, τότε αυτή η λύση θα είναι η καλύτερη.*
  - *Έχει αποδειχθεί ότι έχει την ίδια πολυπλοκότητα σε χώρο και χρόνο με τους DFS και BFS.*
  - *Δεν κινδυνεύει να χαθεί σε κάποιο κλαδί απείρου μήκους.*



Αναζήτηση Διπλής Κατεύθυνσης (1/2)

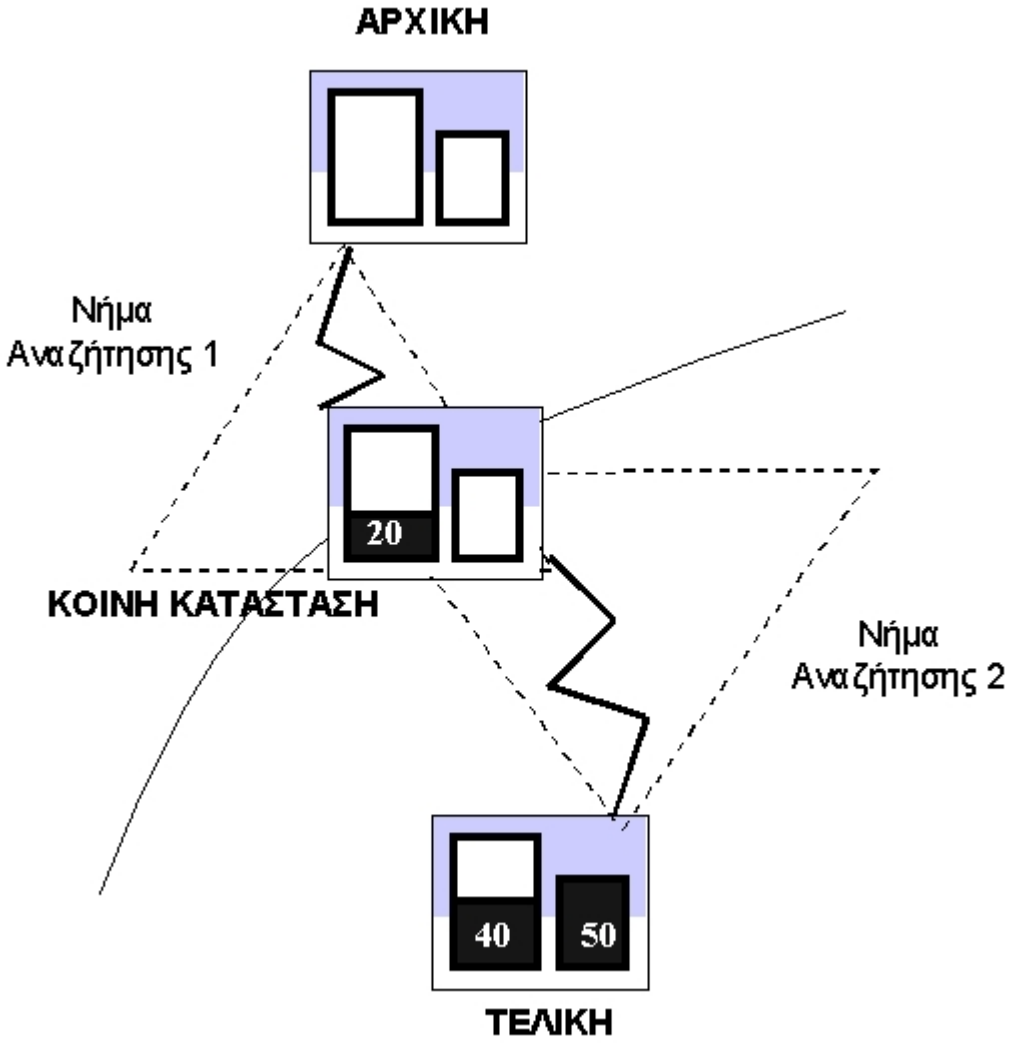
Η ιδέα της *αναζήτησης διπλής κατεύθυνσης* (Bidirectional Search - *BiS*) πηγάζει από τη δυνατότητα του παραλληλισμού (parallelism) στα υπολογιστικά συστήματα.

- Προϋποθέσεις κάτω από τις οποίες μπορεί να εφαρμοστεί:
  - *Οι τελεστές μετάβασης είναι αντιστρέψιμοι (reversible), και*
  - *Είναι πλήρως γνωστή η τελική κατάσταση.*

- Εφαρμογή
  - *Αρχίζει την αναζήτηση από την αρχική και τελική κατάσταση ταυτόχρονα.*
  - *Αν κάποια κατάσταση που επεκτείνεται είναι κοινή, τότε βρέθηκε λύση.*
  - *Λύση είναι η ένωση των μονοπατιών από την κοινή κατάσταση έως την αρχική και έως την τελική κατάσταση.*

- Μειονεκτήματα:
  - *Υπάρχει επιπλέον κόστος επικοινωνίας μεταξύ των δύο αναζητήσεων.*

Αναζήτηση Διπλής Κατεύθυνσης (2/2)



Επέκταση και Οριοθέτηση (B&B)

Ο *αλγόριθμος επέκτασης και οριοθέτησης* (Branch and Bound - *B&B*) εφαρμόζεται σε προβλήματα όπου αναζητείται η βέλτιστη λύση (ελάχιστο κόστος).

- Ο B&B κλαδεύει καταστάσεις (pruning) και μειώνει το χώρο αναζήτησης.
- Παράδειγμα: βρέθηκε μια λύση με κόστος 159. Κατά την αναζήτηση για άλλες λύσεις μια κατάσταση έχει ήδη κόστος 167. Άρα κλαδεύεται αυτή η κατάσταση.

Ο αλγόριθμος B&B:

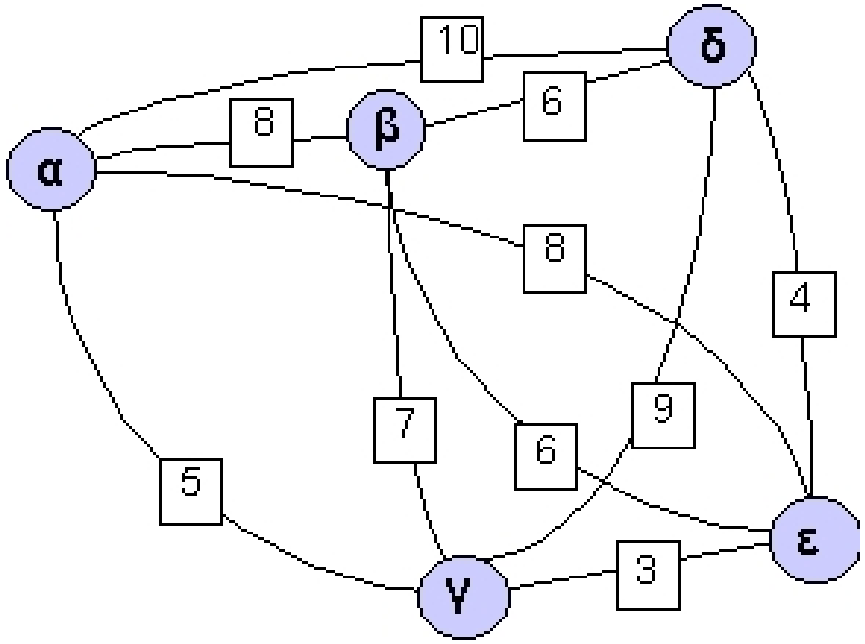
1. Βάλε την αρχική κατάσταση στο μέτωπο της αναζήτησης.
2. Αρχική τιμή της καλύτερης λύσης είναι το $+\infty$ (όριο).
3. Αν το μέτωπο της αναζήτησης είναι κενό, τότε σταμάτησε.
   Η καλύτερη μέχρι τώρα λύση είναι και η βέλτιστη.
4. Βγάλε την πρώτη σε σειρά κατάσταση από το μέτωπο της αναζήτησης.
5. Αν η κατάσταση ανήκει στο κλειστό σύνολο, τότε πήγαινε στο 3.
6. Αν η κατάσταση είναι τελική, τότε ανανέωσε τη λύση ως την καλύτερη μέχρι τώρα και ανανέωσε την τιμή του ορίου με την τιμή που αντιστοιχεί στην τελική κατάσταση. Πήγαινε στο 3.
7. Εφάρμοσε τους τελεστές μεταφοράς για να παράγεις τις καταστάσεις-παιδιά και την τιμή που αντιστοιχεί σε αυτές.
8. Βάλε τις καταστάσεις-παιδιά, των οποίων η τιμή δεν υπερβαίνει το όριο, μπροστά στο μέτωπο της αναζήτησης. **(*)**
9. Βάλε την κατάσταση-γονέα στο κλειστό σύνολο.
10. Πήγαινε στο 3.

(*) Αυτός είναι DFS-B&B γιατί οι νέες καταστάσεις μπαίνουν μπροστά στο μέτωπο αναζήτησης. Υπάρχει και BestFS-B&B.

Ο αλγόριθμος B&B (Ψευδοκώδικας)

```
algorithm b&b(InitialState, FinalStates)
begin
  Closed←∅;
  Frontier←<InitialState>;
  BestCost←∝;
  BestState←null;
  while Frontier≠ ∅ do
    CurrentState←First(Frontier);
    CurrentCost←Cost(Current_State);
    Frontier←delete(CurrentState,Frontier);
    if CurrentCost < BestCost then
  if CurrentState∈ FinalStates then
        BestState←CurrentState;
        BestCost←CurrentCost;
      else
        Next←Expand(CurrentState);
        ChildrenStates←
        {s| s∈Next ∧ s∉Frontier ∧ s∉Closed};
        Frontier←ChildrenStates ^ Frontier;
        Closed←Closed∪{CurrentState};
      endif;
    endif;
  endwhile;
   if BestState = null
      then return fail
    else return BestState and BestCost;
end.
```

Το Πρόβλημα του Πλανόδιου Πωλητή (TSP)



- Το πρόβλημα ανήκει στην κατηγορία *μη-πολυωνυμικού χρόνου* **(***NP-complete***)**.
- Το πρόβλημα είναι πρόβλημα ελαχιστοποίησης κόστους και έχει πολλές εφαρμογές.

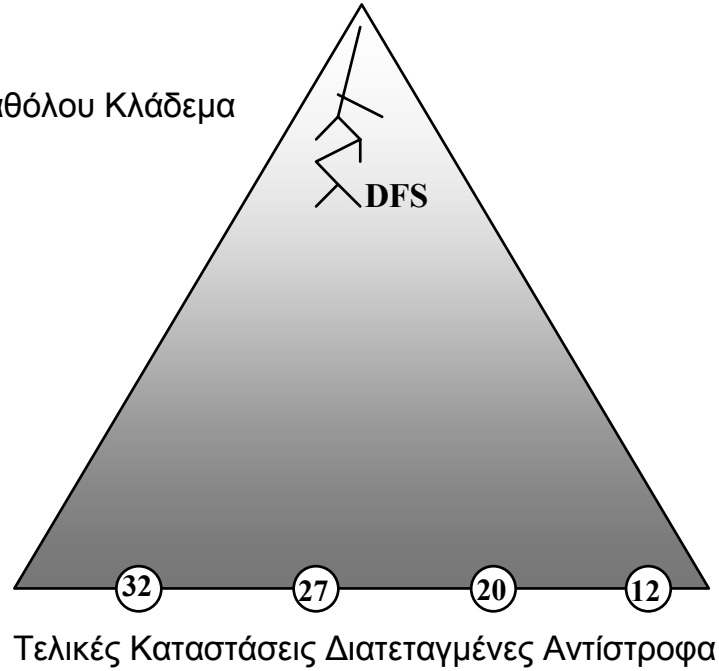
Ο αλγόριθμος B&B στο πρόβλημα TSP

Μέτωπο της αναζήτησης	Κόστος Λύσης	Κατάσταση	Παιδιά
<α>	+∞	α	$αβ^{8}$,$αγ^{5}$,$αδ^{10}$,$αε^{8}$
<$αβ^{8}$,$αγ^{5}$,$αδ^{10}$,$αε^{8}$>	+∞	αβ	$αβγ^{15}$,$αβδ^{14}$, $αβε^{14}$
<$αβγ^{15}$,$αβδ^{14}$,$αβε^{14}$,$αγ^{5}$,...>	+∞	αβγ	$αβγδ^{24}$, $αβγε^{18}$
<$αβγδ^{24}$,$αβγε^{18}$, $αβδ^{14}$,$αβε^{14}$ ...>	+∞	αβγδ	$αβγδε^{28}$
<$αβγδε^{28}$,$αβγε^{18}$, $αβδ^{14}$,...>	+∞	αβγδε	$αβγδεα^{36}$
< $αβγδεα^{36}$, $αβγε^{18}$, $αβδ^{14}$,..>	36	αβγδεα	Τελική Κατάσταση
<$αβγε^{18}$, $αβδ^{14}$,... >	36	αβγε	$αβγεδ^{22}$
<$αβγεδ^{22}$,$αβδ^{14}$,...>	36	αβγεδ	$αβγεδα^{32}$
< $αβγεδα^{32}$,$αβδ^{14}$,$αβε^{14}$ ...>	32	$αβγεδα^{32}$	Τελική Κατάσταση
...	...	...	...
<$αβδεγα^{26}$, ...>	26	αβδεγα	Τελική Κατάσταση
...	...	...	...
<$αβεγδ^{26}$, ...>	26	αβεγδ	Κλάδεμα
....	...	...	...
<$αεβγδ^{30}$, ...>	26	αεβγδ	Κλάδεμα
. . .	. . .	. . .	. . .
<>	ΕλάχιστηΤιμή	ΤΕΛΟΣ	

Ο αλγόριθμος B&B

- Ο B&B εφαρμόζεται όταν υπάρχει μια πραγματική εκτίμηση του κόστους.
- Το κέρδος από το κλάδεμα εξαρτάται από το πόσο γρήγορα θα βρεθεί μια καλή λύση.
- Υπάρχει περίπτωση να μη γίνει καθόλου κλάδεμα αν οι λύσεις είναι διατεταγμένες από τη χειρότερη προς την καλύτερη.
- Στη χειρότερη περίπτωση συμπεριφέρεται σαν τον DFS.

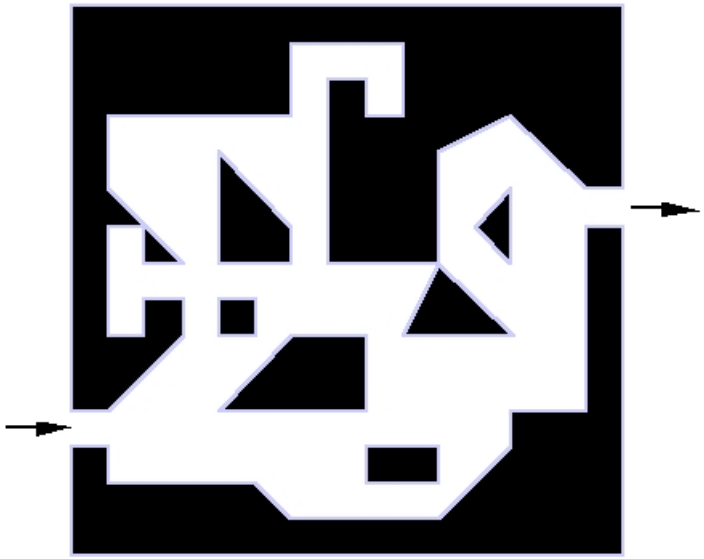
Τελικές Καταστάσεις Διατεταγμένες Ορθά

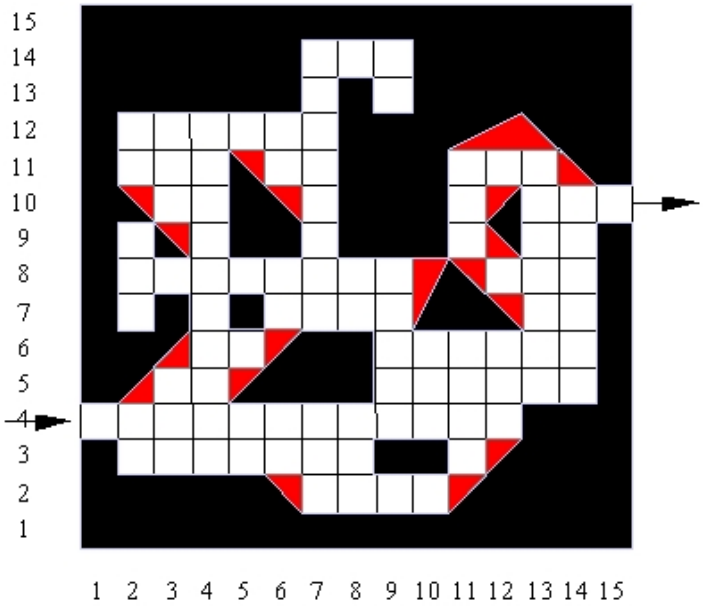


Τελικές Καταστάσεις Διατεταγμένες Αντίστροφα

Εφαρμογή των Αλγορίθμων Τυφλής Αναζήτησης

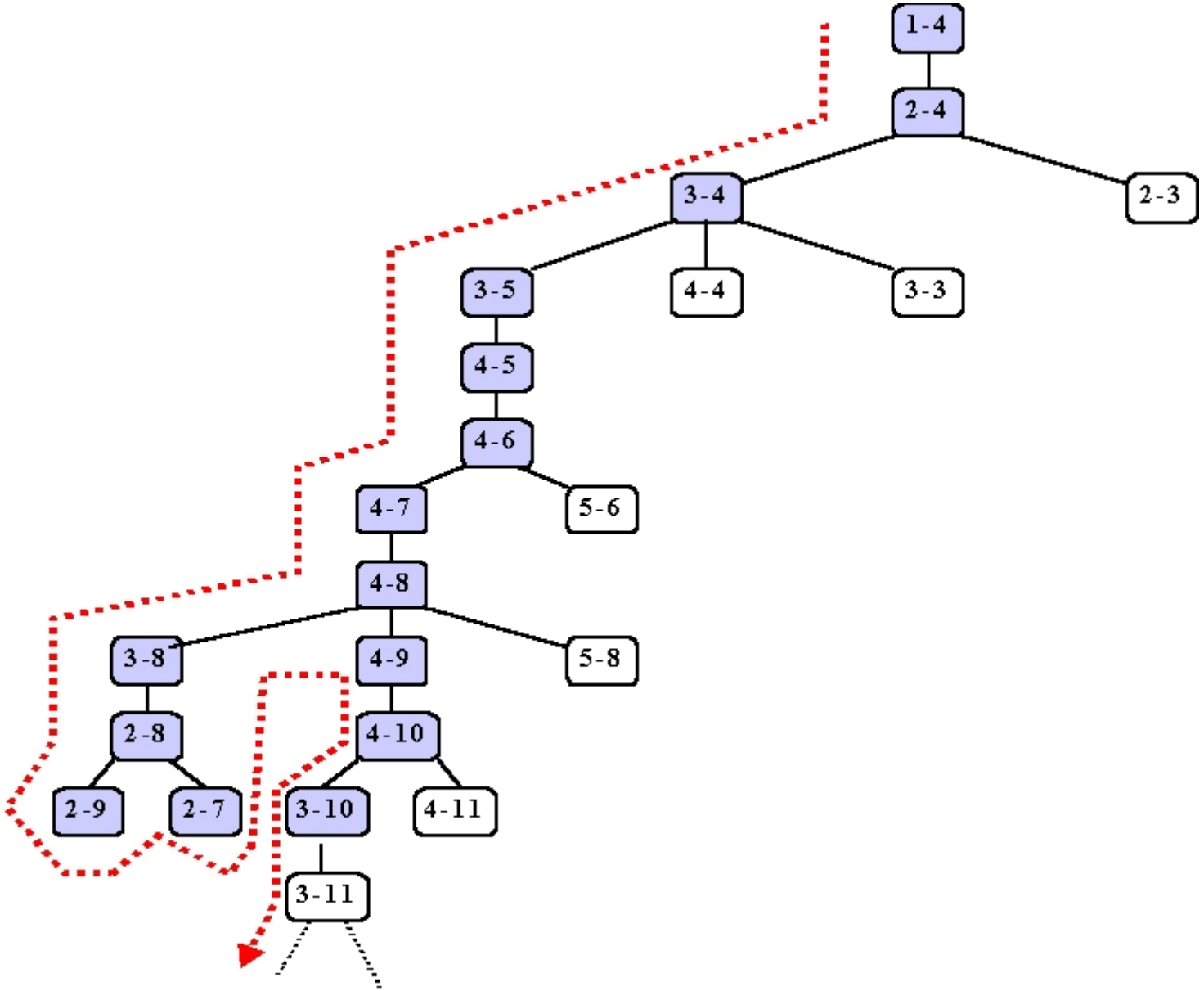
Το πρόβλημα του Λαβύρινθου - Ορισμός



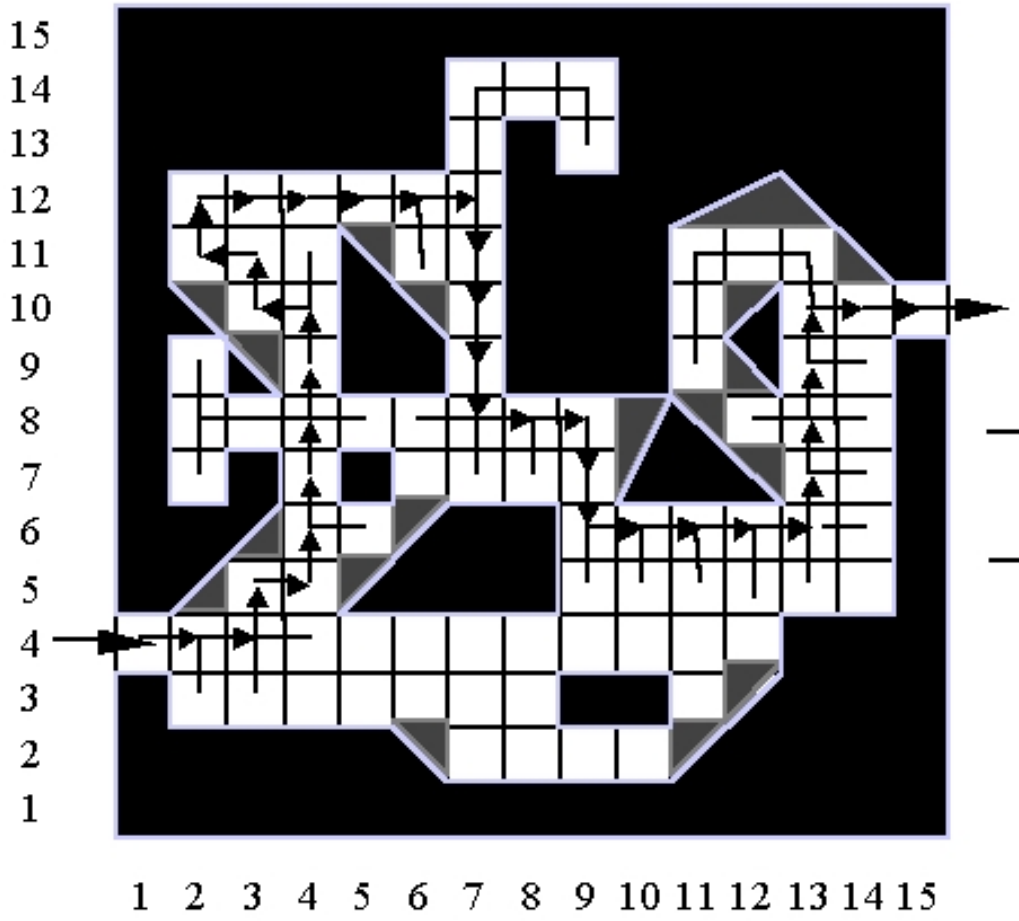


- Αρχική κατάσταση (1,4).
- Το σύνολο τελικών καταστάσεων περιέχει μόνο (15,10).
- Οι τελεστές μεταφοράς είναι οι εξής: πήγαινε μία θέση αριστερά, πήγαινε μία θέση επάνω, πήγαινε μία θέση δεξιά, πήγαινε μία θέση κάτω, εφόσον η θέση είναι ελεύθερη.
- Ο χώρος καταστάσεων είναι όλες οι ελεύθερες θέσεις, χωρίς εμπόδια, του πλέγματος.

Εφαρμογή του αλγορίθμου DFS



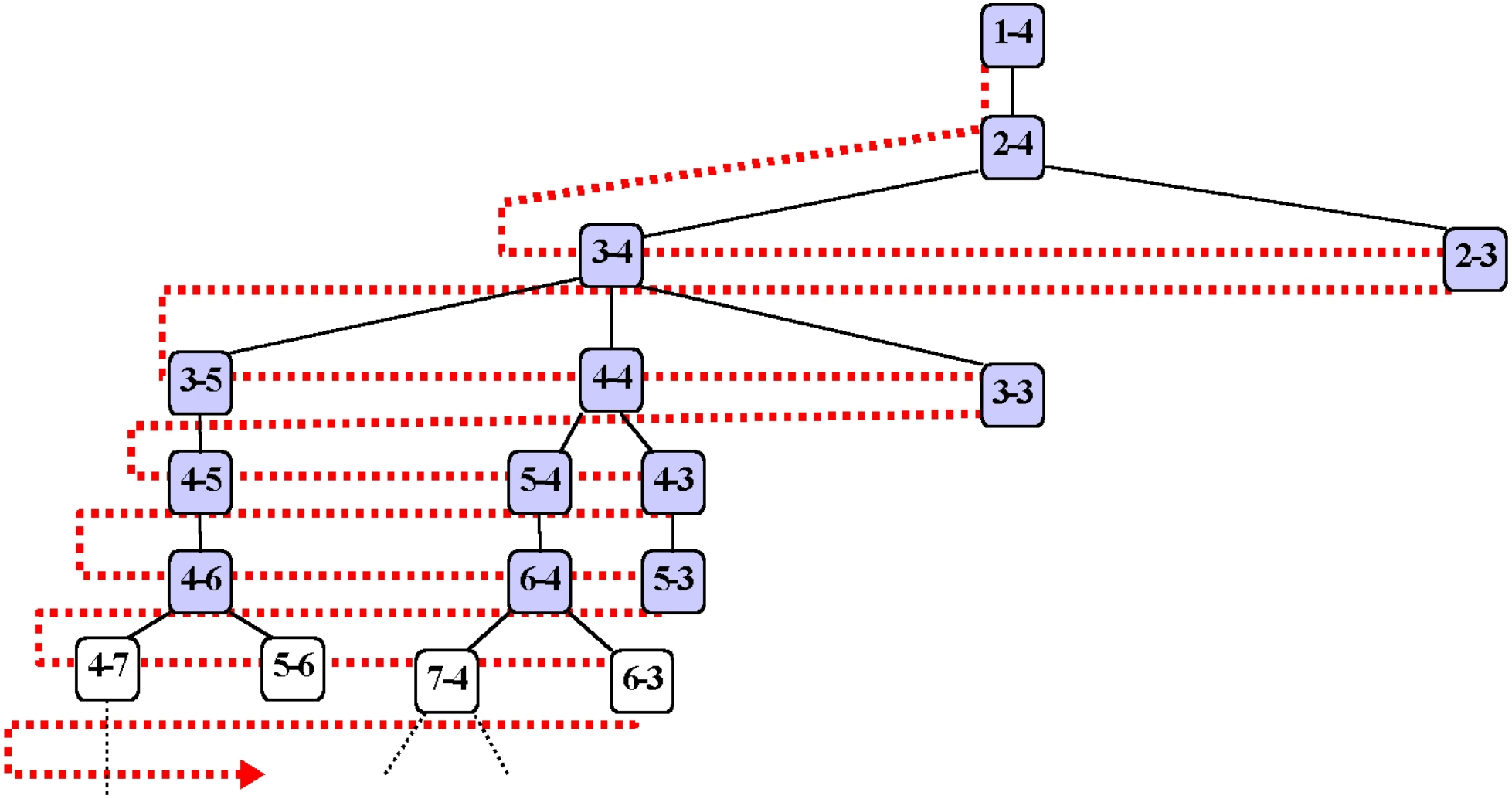
Λύση στο πρόβλημα του λαβύρινθου με χρήση DFS



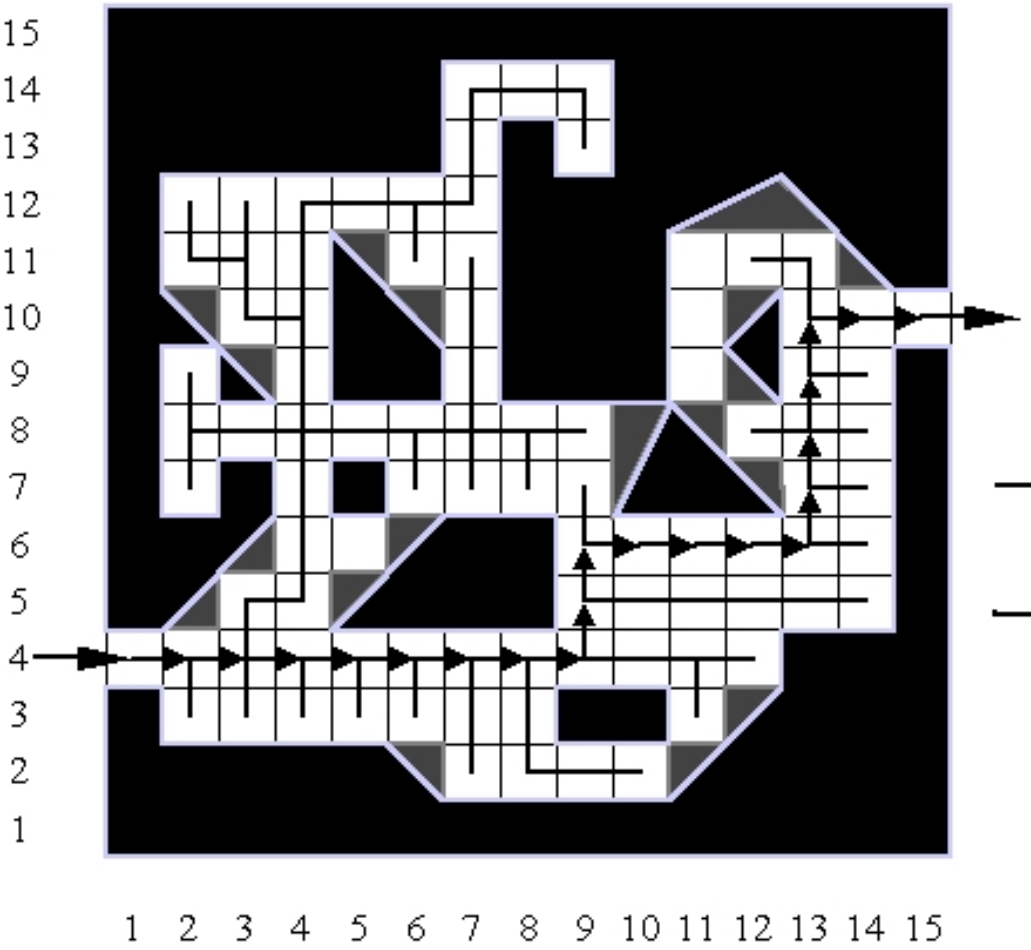
→ Τελεστής που ανήκει στη λύση

— Τελεστής που δεν ανήκει στη λύση αλλά εξετάζεται κατά την εκτέλεση του αλγορίθμου

Εφαρμογή αλγορίθμου BFS



Λύση στο πρόβλημα του λαβύρινθου με χρήση BFS

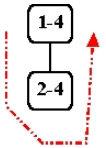


→ Τελεστής που ανήκει στη λύση

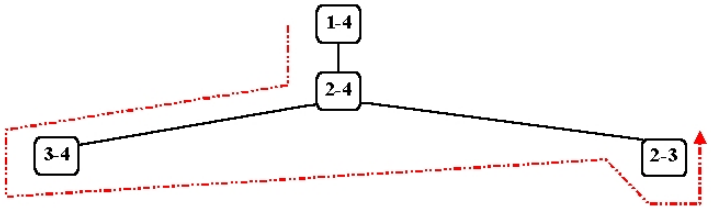
— Τελεστής που δεν ανήκει στη λύση αλλά εξετάζεται κατά την εκτέλεση του αλγορίθμου

Εφαρμογή του ID στο πρόβλημα του λαβυρίνθου

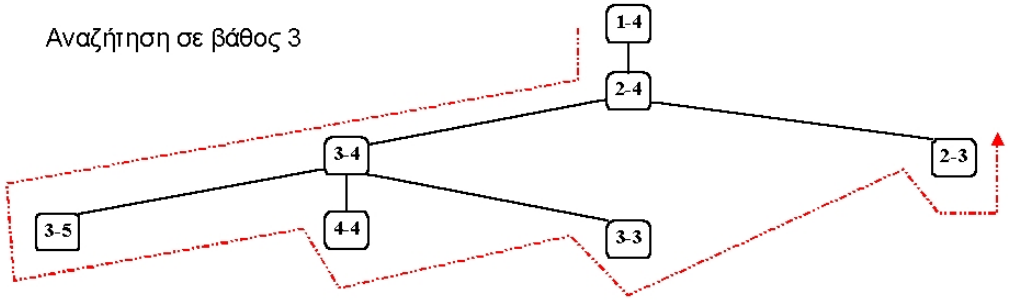
Αναζήτηση σε βάθος 1



Αναζήτηση σε βάθος 2



Αναζήτηση σε βάθος 3



Αναζήτηση σε βάθος 4