

# RICERCA IN UNO SPAZIO DI SOLUZIONI

# Outline

- ◇ Formulazione del problema
- ◇ Esempi di problemi
- ◇ Alcuni algoritmi di ricerca

## Esempio: Romania

In vacanza in Romania; ora ad Arad.  
Il volo parte domani da Bucharest

Formulare il goal:

essere a Bucharest

Formulate il problema:

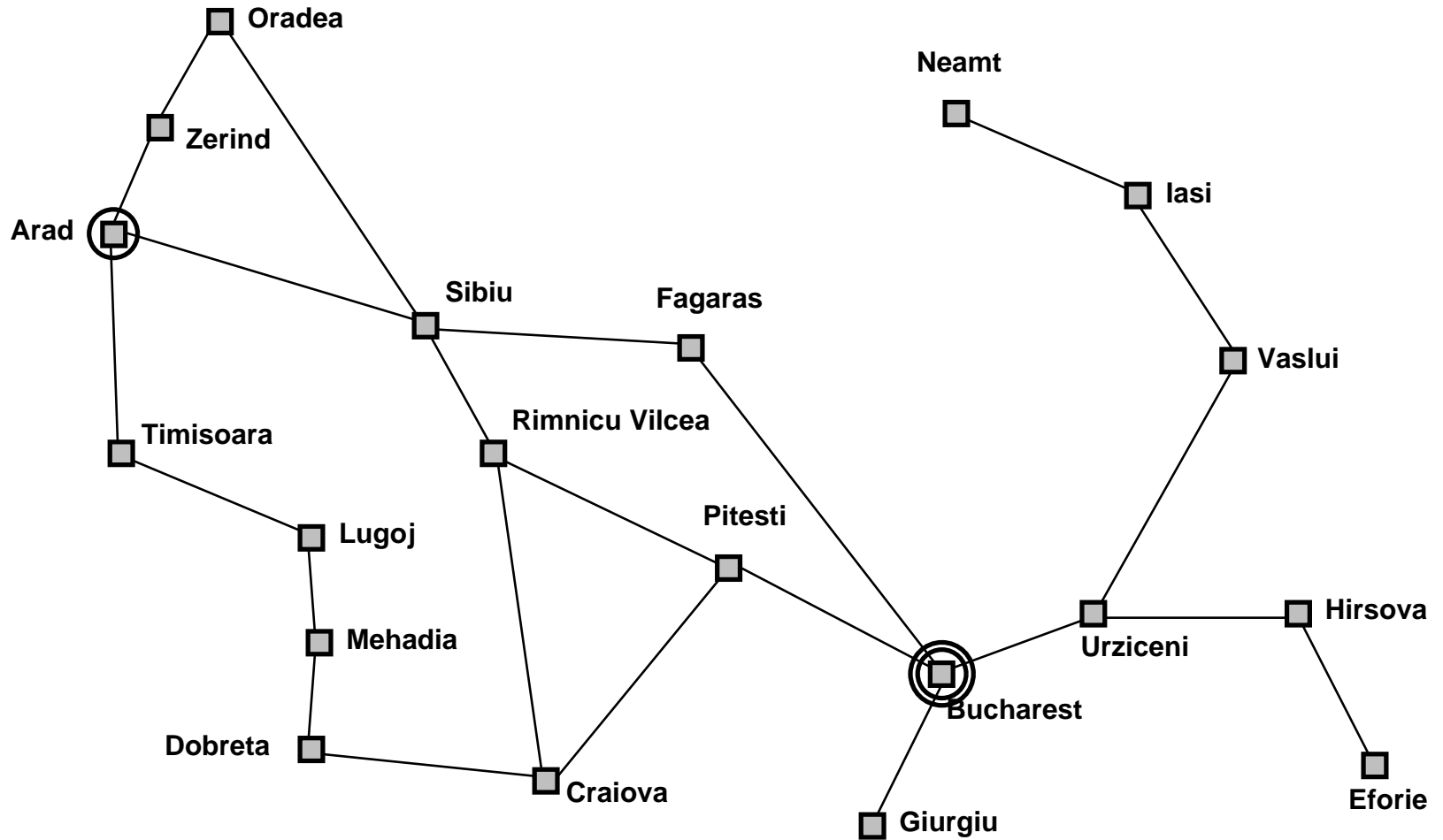
*stati:* varie citta'

*operatori:* viaggi tra le citta'

Tovare una soluzione:

sequenza di citta', esempio: Arad, Sibiu, Fagaras, Bucharest

# Esempio: Romania



## Formulazione del problema (singoli stati)

Un *problema* e' definito da quattro elementi:

stato iniziale es.: "ad Arad"

operatori (o *funzione successore*  $S(x)$ )

es.: Arad  $\rightarrow$  Zerind      Arad  $\rightarrow$  Sibiu      etc.

Un test per il goal, puo' essere

*esplicito*, es.:  $x =$  "a Bucharest"

*implicito*, es.: Bucharest( $x$ )

costo di un cammino (additivo)

es.: somma di distanze, numero di operatori eseguiti, ...

Una *soluzione* e' una sequenza di operatori  
che porta dallo stato iniziale ad uno stato di goal

## Selezionare uno spazio degli stati

Il mondo reale e' molto complesso

⇒ lo spazio degli stati deve essere *astratto* per risolvere il problema

Stato (astratto) = insieme di stati reali

Operatore (astratto) = combinazione complessa di azioni reali

es.: "Arad → Zerind" rappresenta un insieme complesso di possibili strade, detour, fermate, ...

Per garantire la realizzabilita', qualsiasi stato reale "in Arad" deve portare a *qualche* stato reale "in Zerind"

Soluzione (astratta) =

insieme di cammini reali che sono soluzioni in quel mondo reale

Ogni azione astratta dovrebbe essere "piu' semplice" del problema originale.

## Esempio: il puzzle

|   |   |   |
|---|---|---|
| 5 | 4 |   |
| 6 | 1 | 8 |
| 7 | 3 | 2 |

Start State

|   |   |   |
|---|---|---|
| 1 | 2 | 3 |
| 8 |   | 4 |
| 7 | 6 | 5 |

Goal State

stati??: caselle (ignora posizioni intermedie)

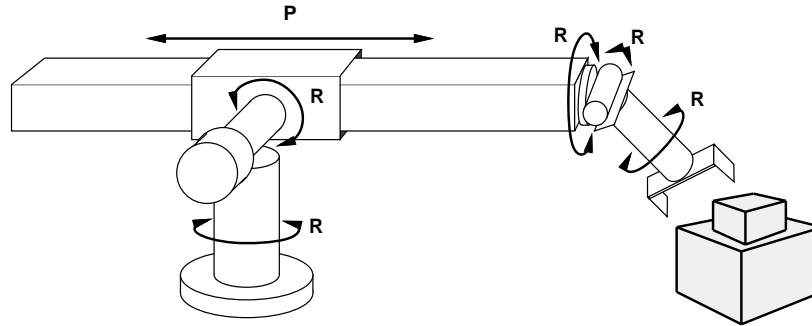
operatori??: mossa del bianco a sinistra, destra, alto, basso

test per il goal??: = stato di goal (dato)

costo di un cammino??: 1 per mossa

[Nota: soluzione ottima e' NP-hard]

## Esempio: assemblaggio robotico



stati??: coordinate reali

degli snodi del robot e delle parti dell'oggetto da assemblare

operatori??: mosse continue degli snodi del robot

test per il goal??: assemblaggio completo

costo di un cammino??: tempo per eseguire il tutto



# Algoritmi di ricerca

Idea di base:

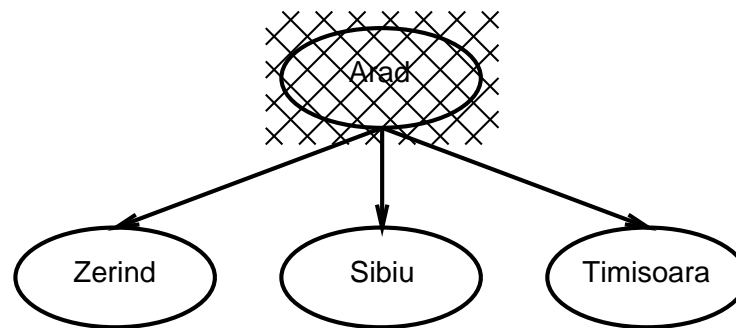
offline, esplorazione simulata dello spazio degli stati  
generando successori di stati già esplorati  
(cioè *espandendo* stati)

```
function GENERAL-SEARCH(problem, strategy) returns a solution, or failure
  initialize the search tree using the initial state of problem
  loop do
    if there are no candidates for expansion then return failure
    choose a leaf node for expansion according to strategy
    if the node contains a goal state then return the corresponding solution
    else expand the node and add the resulting nodes to the search tree
  end
```

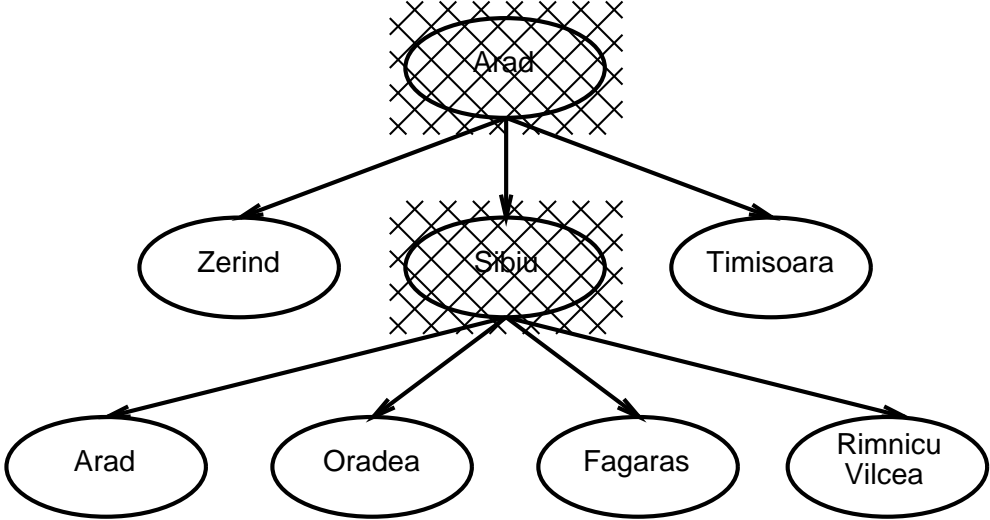
# Esempio di ricerca

Arad

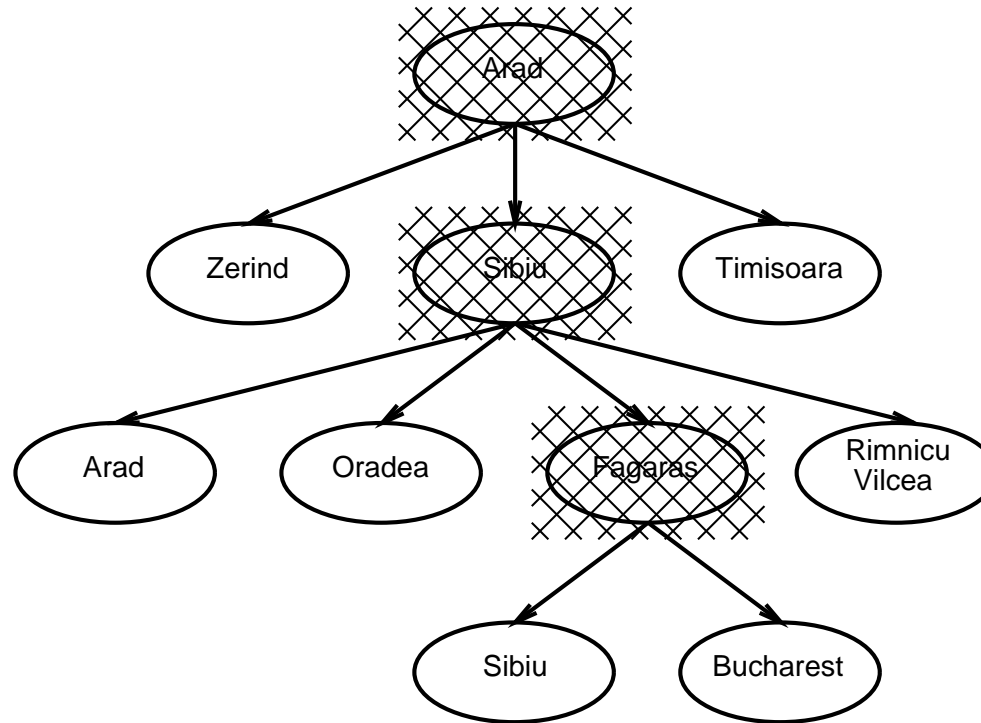
# Esempio di ricerca



# Esempio di ricerca



# Esempio di ricerca



# Implementazione degli algoritmi di ricerca

```
function GENERAL-SEARCH(problem, QUEUING-FN) returns a solution, or failure
  nodes ← MAKE-QUEUE(MAKE-NODE(INITIAL-STATE[problem]))
  loop do
    if nodes is empty then return failure
    node ← REMOVE-FRONT(nodes)
    if GOAL-TEST[problem] applied to STATE(node) succeeds then return node
    nodes ← QUEUING-FN(nodes, EXPAND(node, OPERATORS[problem]))
  end
```

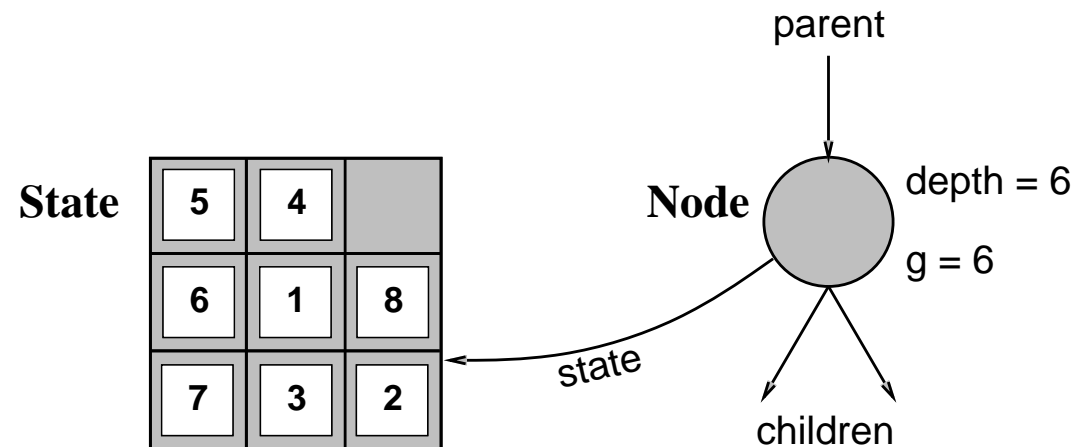
## Implementazione: stati vs. nodi

Uno *stato* e' una rappresentazione di una configurazione fisica

Un *nodo* e' una struttura dati che fa parte di un albero di ricerca

include *genitori, figli, profondita', costo del cammino  $g(x)$*

*Stati* non hanno genitori, figli, ...



La funzione `EXPAND` crea nuovi nodi, riempiendo i vari campi e usando gli `OPERATORS` (o `SUCCESSORFN`) del problema per creare gli stati corrispondenti.

# Strategie di ricerca

Una strategia e' definita scegliendo un *ordine di espansione dei nodi*

Le strategie sono valutate secondo le seguenti dimensioni:

completezza—trova sempre una soluzione se ne esiste una?

complessita' in tempo—numero di nodi generati/espansi

complessita' in spazio—massimo numero di nodi in memoria

ottimalita'—trova sempre una soluzione di costo minimo?

La complessita' in spazio e tempo sono misurate in termini di

$b$ —massimo fattore di branching dell'albero di ricerca

$d$ —profondita' della soluzione di costo minimo

$m$ —massima profondita' dello spazio degli stati (puo' essere  $\infty$ )



## Strategie di ricerca non informate

Le strategie *non informate* usano solo l'informazione disponibile nella definizione del problema

Ricerca breath-first

Ricerca con costo uniforme

Ricerca depth-first

Ricerca depth-limited

Ricerca iterative deepening

# Ricerca breadth-first

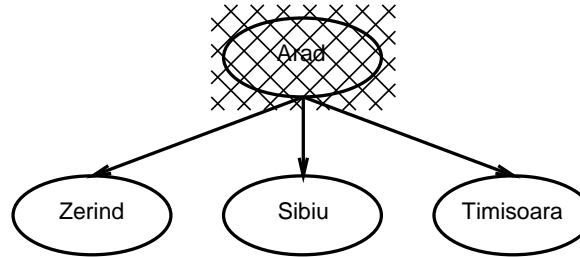
Espande sempre il nodo meno profondo



Arad

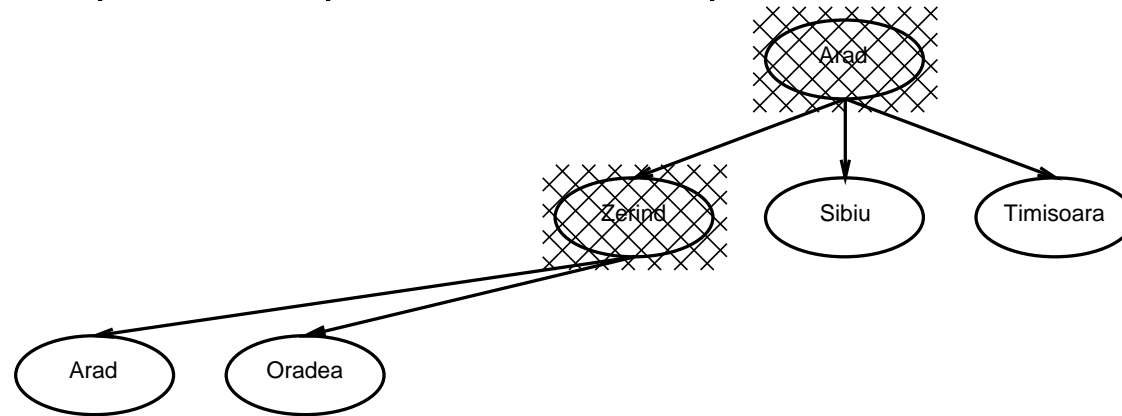
# Ricerca breadth-first

Espande sempre il nodo meno profondo



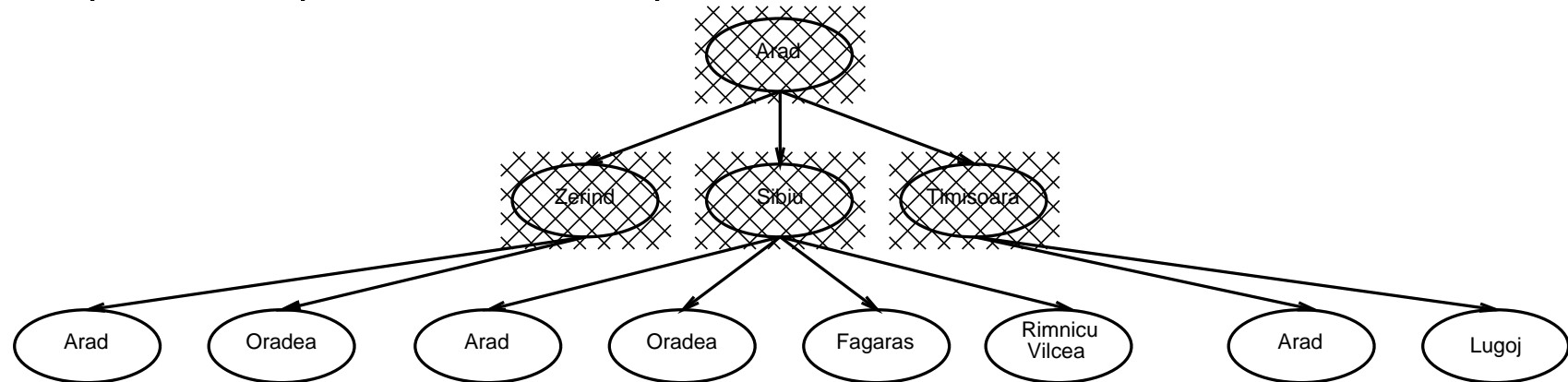
# Ricerca breadth-first

Espande sempre il nodo meno profondo



# Ricerca breadth-first

Espande sempre il nodo meno profondo



## Proprieta' della ricerca breadth-first

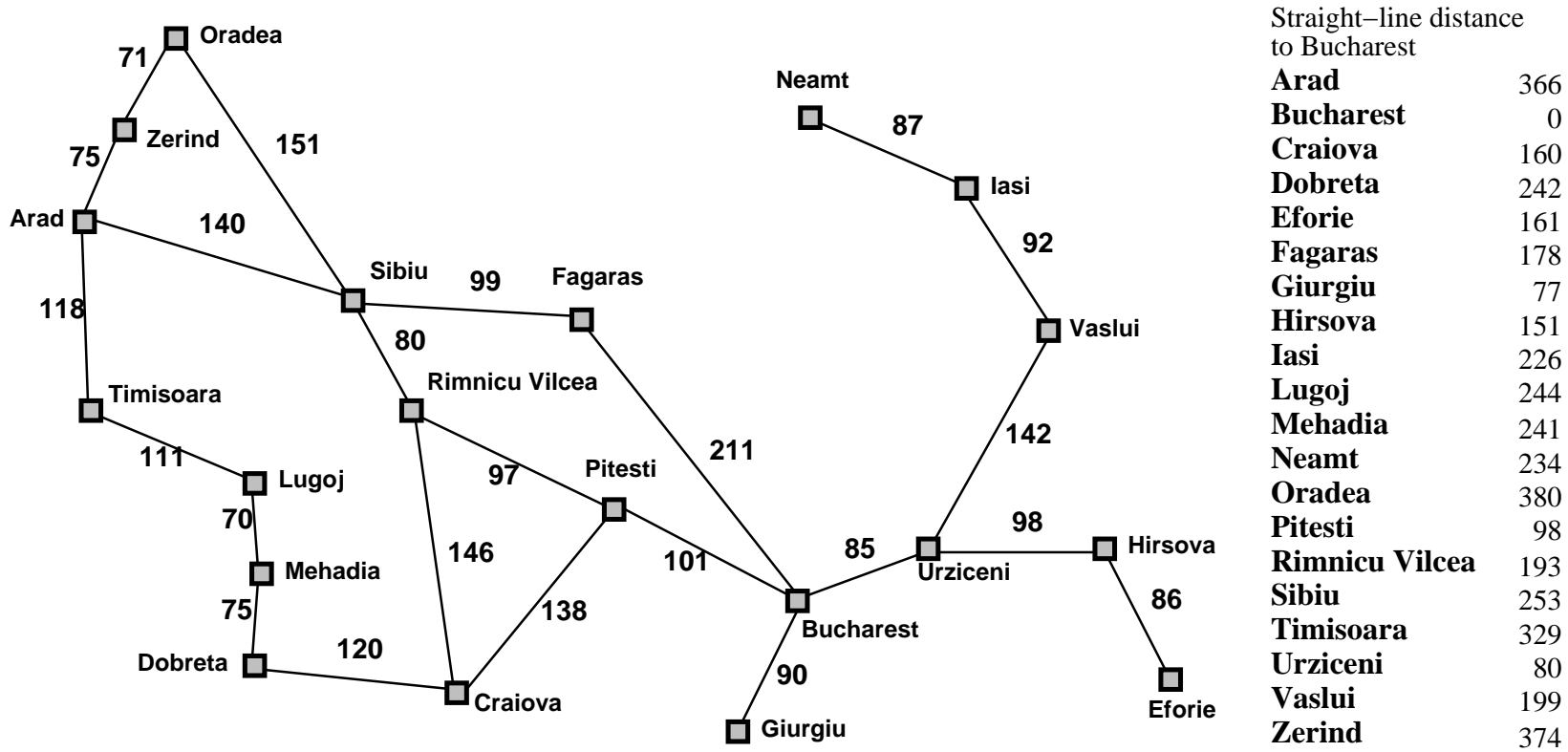
Completa?? Si (se  $b$  e' finito)

Tempo??  $1 + b + b^2 + b^3 + \dots + b^d = O(b^d)$ , cioe' esponenziale in  $d$

Spazio??  $O(b^d)$  (mantiene ogni nodo in memoria)

Ottima?? Si (se costo = 1 per passo); non ottima in generale

# Romania con costo dei passi in Km



# Ricerca a costo uniforme

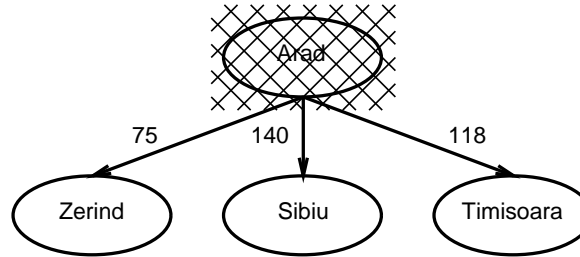
Espande il nodo con costo minimo





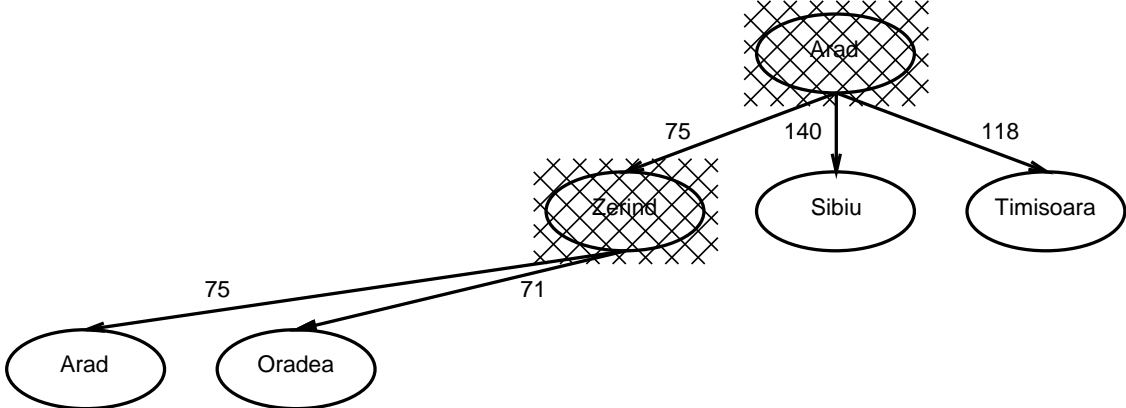
# Ricerca a costo uniforme

Espande il nodo con costo minimo



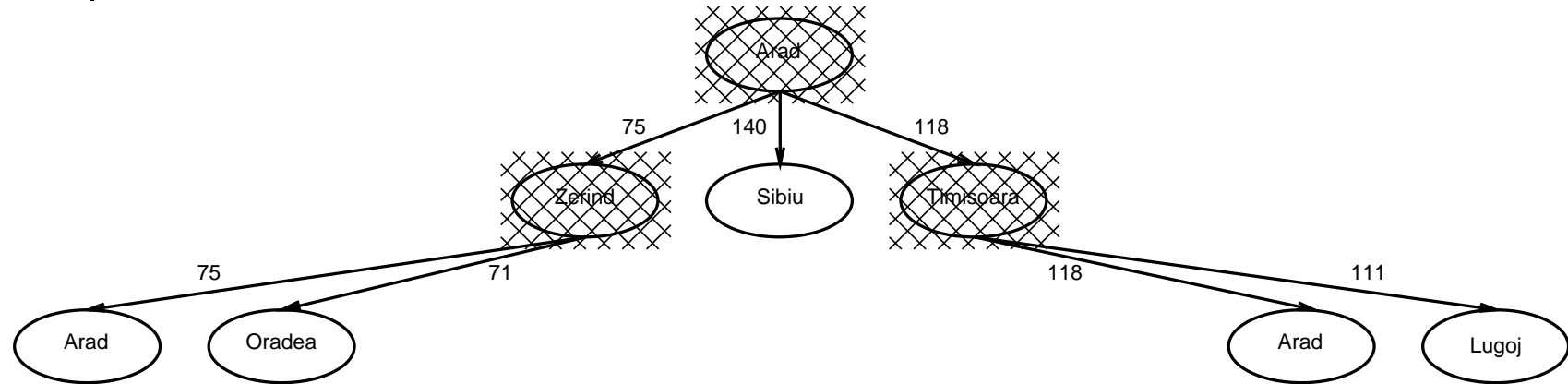
# Ricerca a costo uniforme

Espande il nodo con costo minimo



# Ricerca a costo uniforme

Espande il nodo con costo minimo



## Proprieta' della ricerca a costo uniforme

Completa?? Si, se costo di un passo  $\geq \epsilon$

Tempo?? # di nodi con  $g \leq$  costo della soluzione ottima

Spazio?? # di nodi con  $g \leq$  costo della soluzione ottima

Ottima?? Si

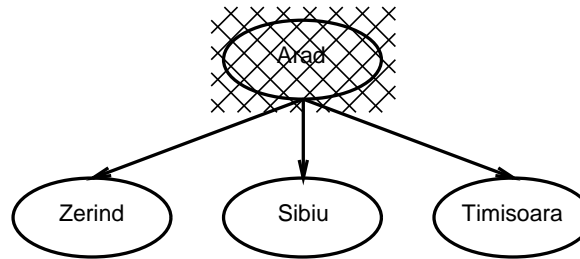
# Ricerca depth-first

Espande il nodo piu' profondo



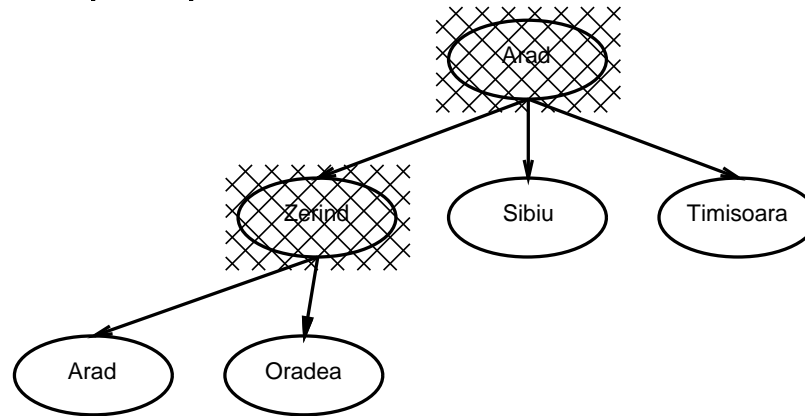
# Ricerca depth-first

Espande il nodo piu' profondo



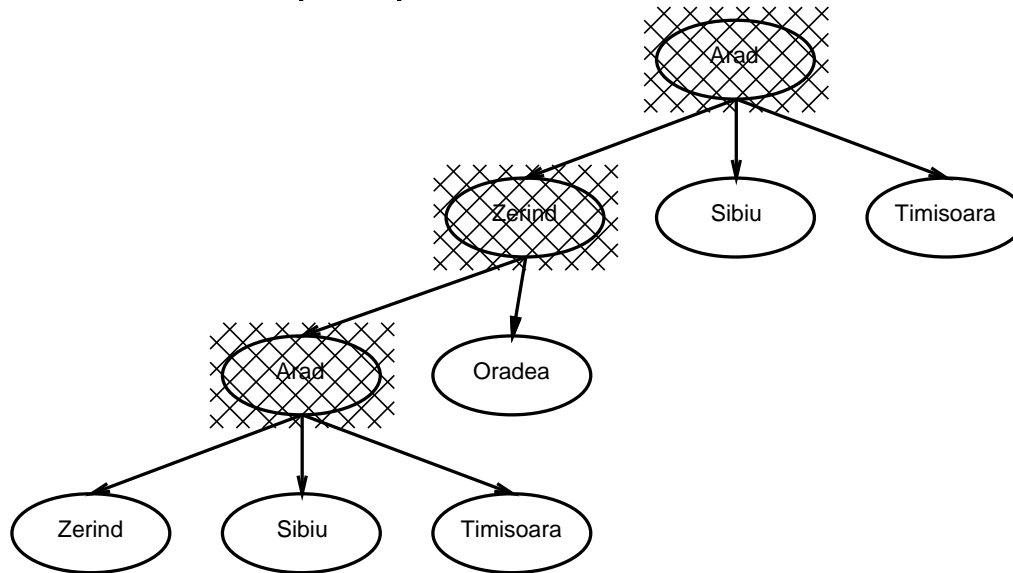
# Ricerca depth-first

Espande il nodo piu' profondo



# Ricerca depth-first

Espande il nodo piu' profondo



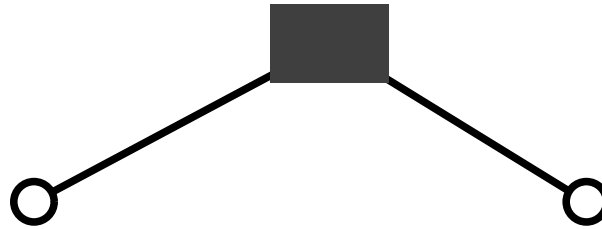
Nota: puo' avere dei cammini infiniti → ha bisogno di uno spazio di ricerca finito e non ciclico (o si deve controllare che gli stati non si ripetano)



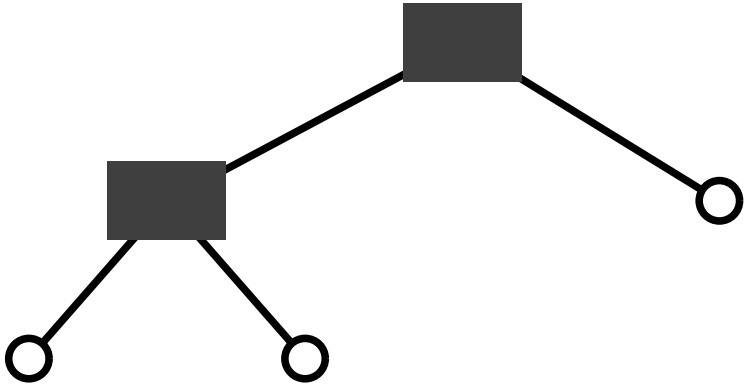
## DFS su un albero binario di profondita' 3



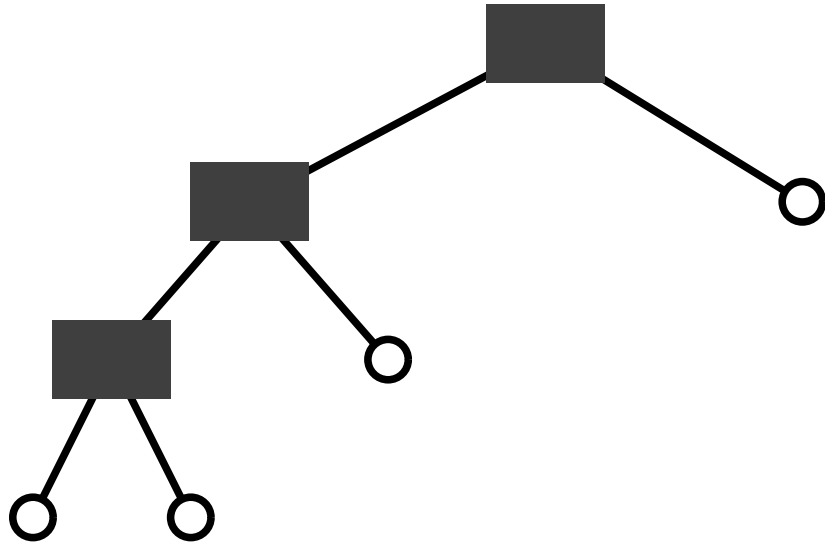
# DFS su un albero binario di profondita' 3



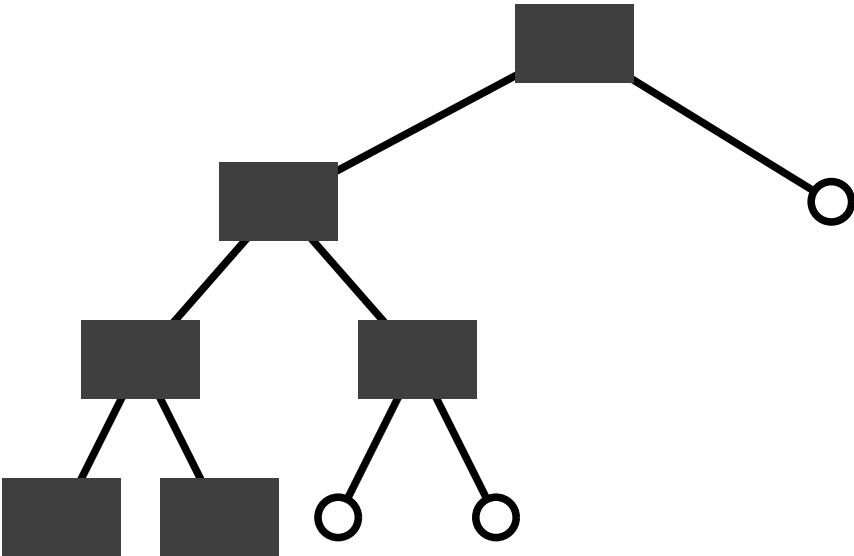
**DFS su un albero binario di profondita' 3**



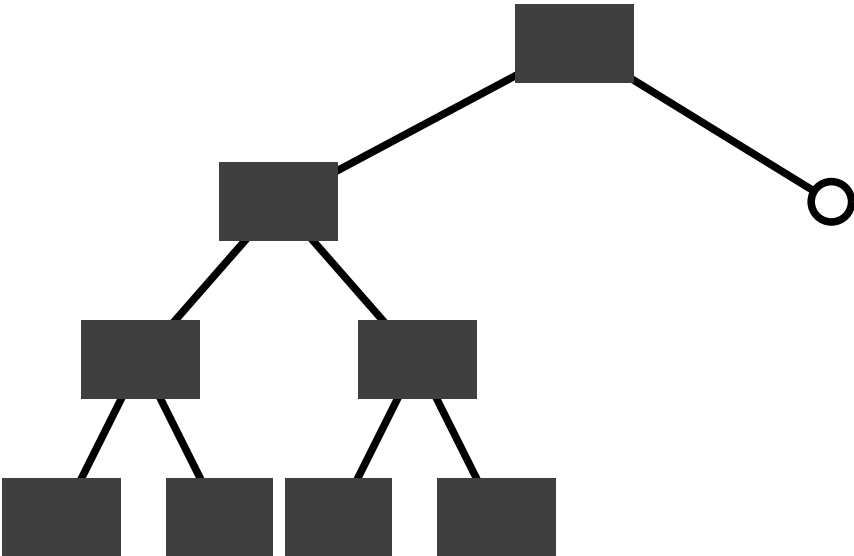
**DFS su un albero binario di profondita' 3**



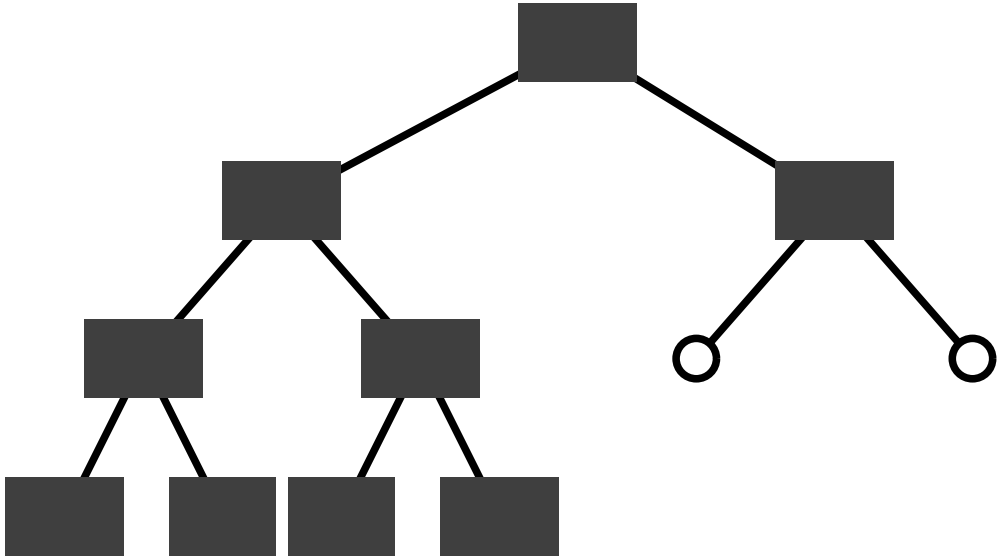
**DFS su un albero binario di profondita' 3**



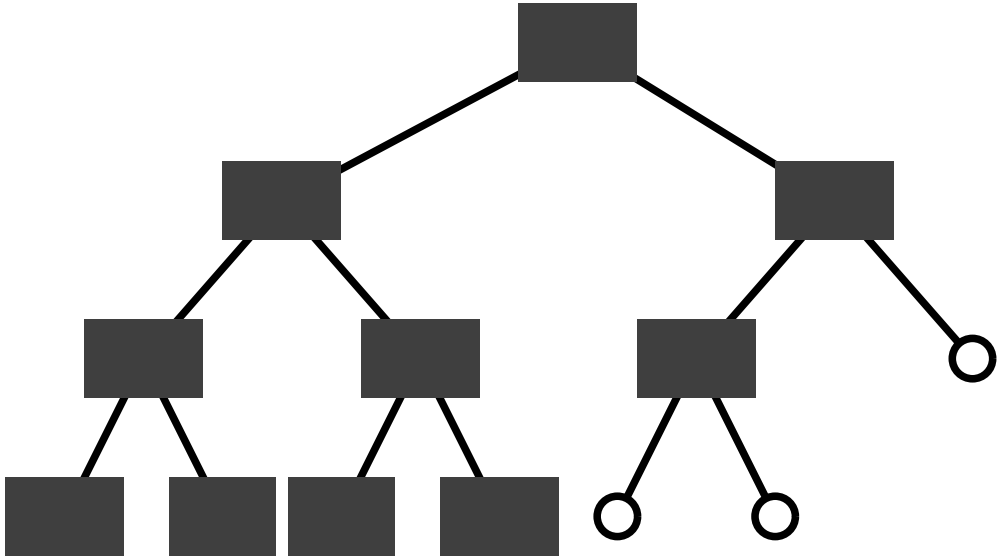
**DFS su un albero binario di profondita' 3**



**DFS su un albero binario di profondita' 3**

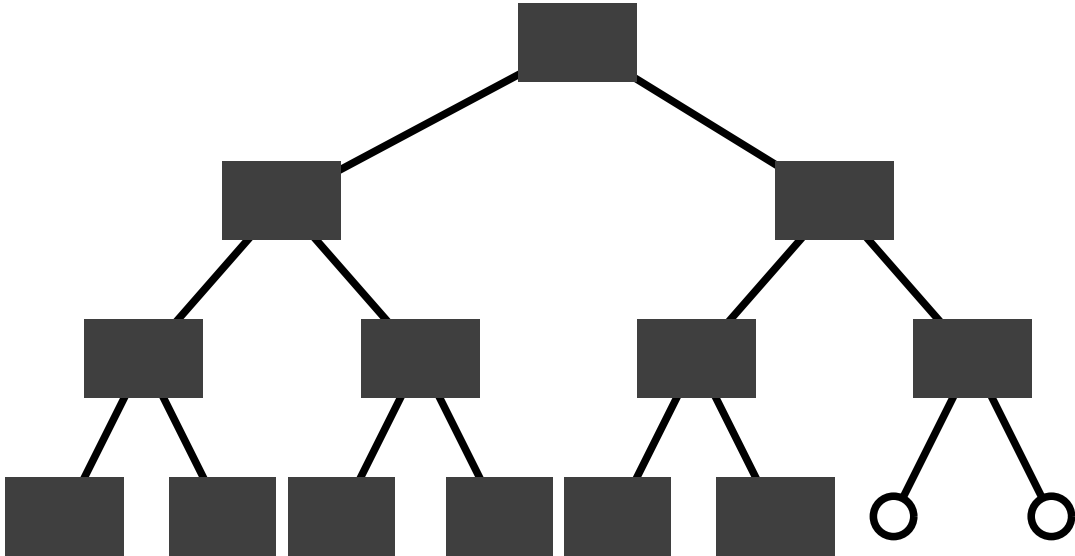


**DFS su un albero binario di profondita' 3**





**DFS su un albero binario di profondita' 3**



## Proprieta' della ricerca depth-first

Completa?? No: fallisce in spazi di profondita' infinita' e spazi con cicli

Modifica per evitare stati ripetuti lungo i cammini

⇒ completa in spazi finiti

Tempo??  $O(b^m)$ : terribile se  $m$  e' molto piu' grande di  $d$

ma se le soluzioni sono dense, puo' essere molto piu' veloce della ricerca breadth-first

Spazio??  $O(bm)$ , cioe' lineare!

Ottima?? No

## Ricerca depth-limited

= ricerca depth-first con profondita' limitata (a  $l$ )

Implementazione:

Nodi a profondita'  $l$  non hanno successori

Completa se  $l \geq d$ .

Non ottima.

Tempo:  $O(b^l)$

Spazio:  $O(b \times l)$

# Ricerca iterativa deepening

Prova tutti i possibili limiti di profondita'.

```
function ITERATIVE-DEEPENING-SEARCH(problem) returns a solution sequence
  inputs: problem, a problem
  for depth  $\leftarrow$  0 to  $\infty$  do
    result  $\leftarrow$  DEPTH-LIMITED-SEARCH(problem, depth)
    if result  $\neq$  cutoff then return result
  end
```

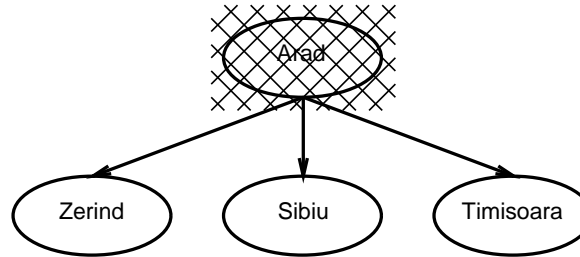
# Ricerca iterative deepening con $l = 0$

Arad

# Ricerca iterative deepening con $l = 1$

Arad

# Ricerca iterative deepening con $l = 1$

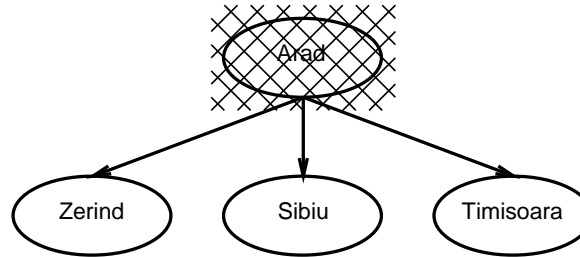


# Ricerca iterative deepening con $l = 2$

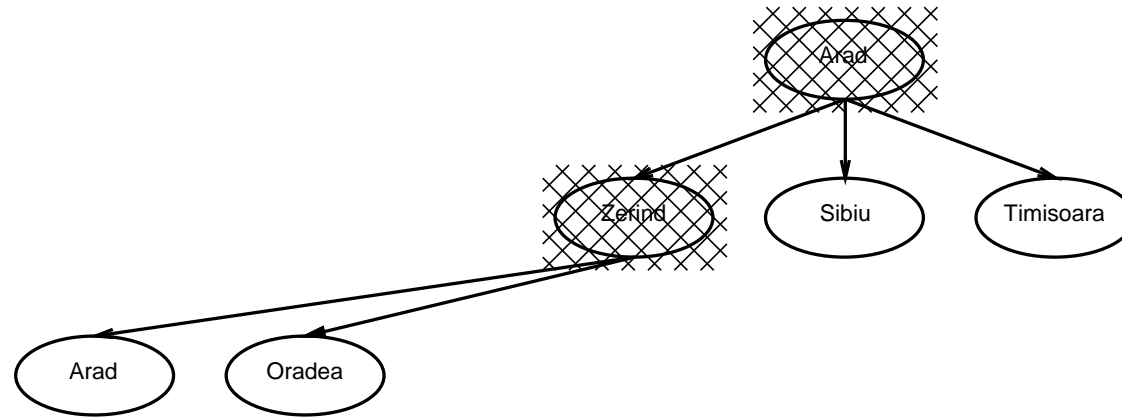
Arad



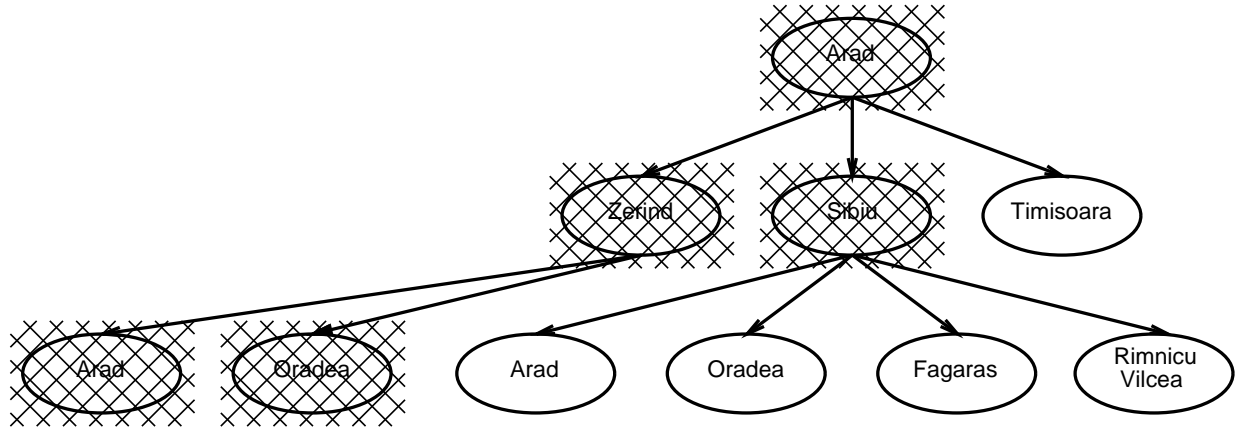
# Ricerca iterative deepening con $l = 2$



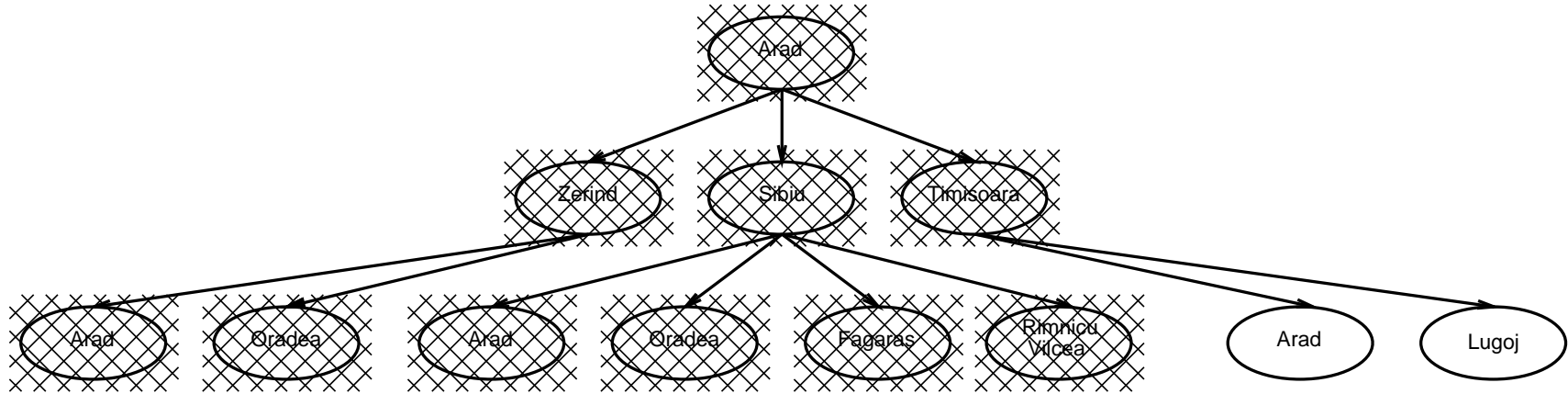
# Ricerca iterative deepening con $l = 2$



# Ricerca iterative deepening con $l = 2$



# Ricerca iterative deepening con $l = 2$



## Proprieta' della ricerca iterative deepening

Completa?? Si

Tempo??  $(d + 1)b^0 + db^1 + (d - 1)b^2 + \dots + b^d = O(b^d)$

Spazio??  $O(bd)$

Ottima?? Si, se il costo di un passo e' 1

Puo' essere modificata per esplorare l'albero a costo uniforme

## Confronto tra le strategie

| Critero    | Breadth-First | Uniform-Cost | Depth-First | Depth-Limited     | Iterative Deepening |
|------------|---------------|--------------|-------------|-------------------|---------------------|
| Tempo      | $b^d$         | $b^d$        | $b^m$       | $b^l$             | $b^d$               |
| Spazio     | $b^d$         | $b^d$        | $bm$        | $bl$              | $bd$                |
| Ottima ?   | si            | si           | no          | no                | si                  |
| Completa ? | si            | si           | no          | si, se $l \geq d$ | si                  |

## Stati ripetuti

In molti problemi è impossibile evitare stati che si ripetono

Es.: azioni reversibili (vedi puzzle)

Questo problema si affronta di solito in 3 possibili modi:

- evitare di generare il nuovo nodo se e' uguale al nodo corrente (spazio: costante)
- evitare di generare il nuovo nodo se e' uno degli avi (nel cammino dalla radice al nodo corrente) (spazio:  $O(d)$ )
- evitare di generare il nuovo nodo se e' stato gia' generato (spazio:  $O(b^d)$ , in realta'  $O(s)$ )

## Sommario

La formulazione del problema di solito richiede un'astrazione dai dettagli del mondo reale per definire uno spazio degli stati che possa essere esplorato

Varie strategie di ricerca non-informate

La ricerca iterative deepening usa spazio solo lineare e non ha bisogno di molto più tempo delle altre strategie non informate