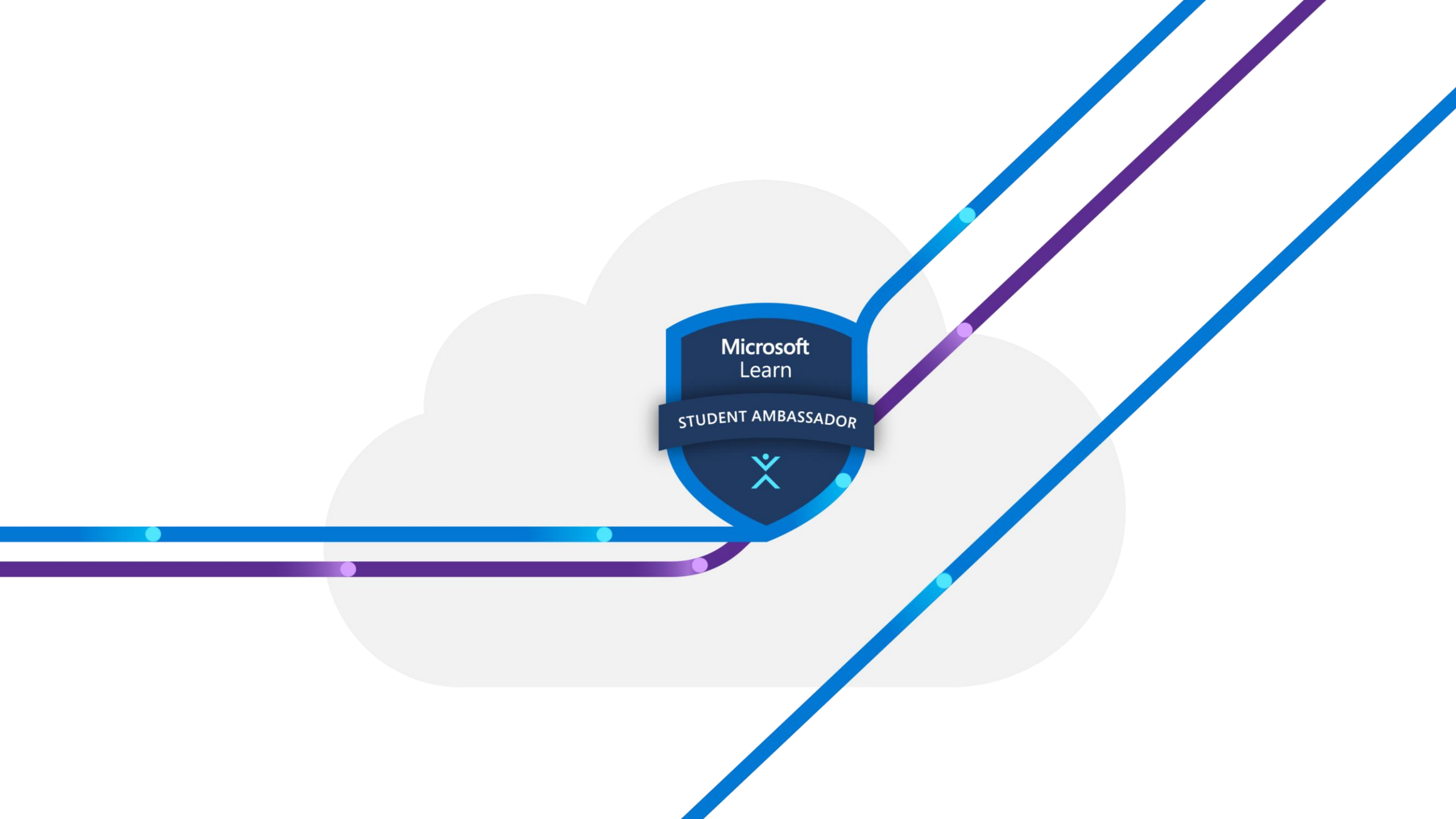

Microsoft
Learn
STUDENT AMBASSADOR

Microsoft Learn
Student Ambassadors

Search Path

Mario Cuomo

13.10.2021

Microsoft Learn
Student Ambassadors

MARIO CUOMO

mariocuomo.github.io
linkedin/in/mariocuomo
@mariocuomo.exe
@mariocuomoEXE

Intelligenza Artificiale

Intelligenza Artificiale

L’Intelligenza Artificiale studia come far eseguire al computer compiti che per l’uomo richiederebbero intelligenza.

Intelligenza Artificiale

L'Intelligenza Artificiale studia come far eseguire al computer compiti che per l'uomo richiederebbero intelligenza.

Nasce nel 1950.
Alan Turing - *Computing Machinery and Intelligence*

Search Path

Ricerca di cammino da un punto A a un punto B

Search Path

Ricerca di cammino da un punto A a un punto B

- Definizione dell'obiettivo

Search Path

Ricerca di cammino da un punto A a un punto B

- Definizione dell'obiettivo
- Formulare il problema

Search Path

Ricerca di cammino da un punto A a un punto B

- Definizione dell'obiettivo
- Formulare il problema
- Trovare una sequenza di passi

Search Path

Ricerca di cammino da un punto A a un punto B

- Definizione dell'obiettivo
  - Identificare un insieme di stati
- Formulare il problema
- Trovare una sequenza di passi

Search Path

Ricerca di cammino da un punto A a un punto B

- Definizione dell'obiettivo
  - Identificare un insieme di stati
- Formulare il problema
  - Identificare un insieme di azioni
- Trovare una sequenza di passi

Esempio (classico in letteratura)

Mi trovo ad Aarad e devo raggiungere Bucarest.

Esempio (classico in letteratura)

Mi trovo ad Aarad e devo raggiungere Bucarest.

Le città intermedie sono gli **stati**.

Esempio (classico in letteratura)

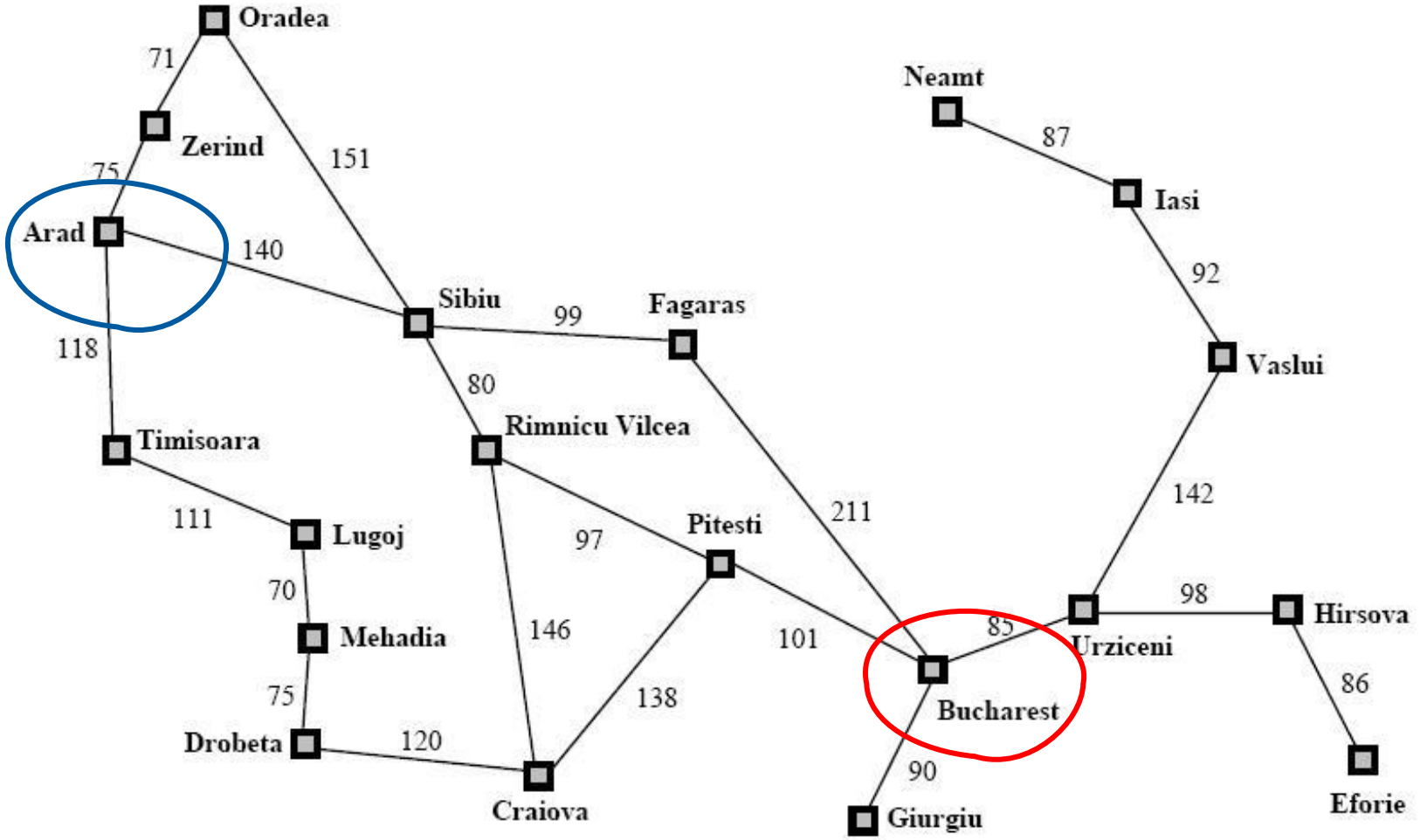
Mi trovo ad Aarad e devo raggiungere Bucarest.

Le città intermedie sono gli **stati**.
Gli spostamenti tra le città sono le **azioni**.

Esempio (classico in letteratura)

Mi trovo ad Aarad e devo raggiungere Bucarest.

Le città intermedie sono gli **stati**.
Gli spostamenti tra le città sono le **azioni**.
La sequenza di città visitate è la **soluzione**.


Oradea
71
Zerind
75
151
Arad
140
Sibiu
99
Fagaras
118
80
Timisoara
Rimnicu Vilcea
111
Lugoj
97
Pitesti
211
70
Mehadia
146
101
75
138
Drobeta
120
Craiova
Neamt
87
Iasi
92
Vaslui
142
98
Hirsova
85
Urziceni
86
Bucharest
90
Giurgiu
Eforie

Rappresentazione

Per rappresentare la sequenza delle operazioni da effettuare si utilizza un albero.

Rappresentazione

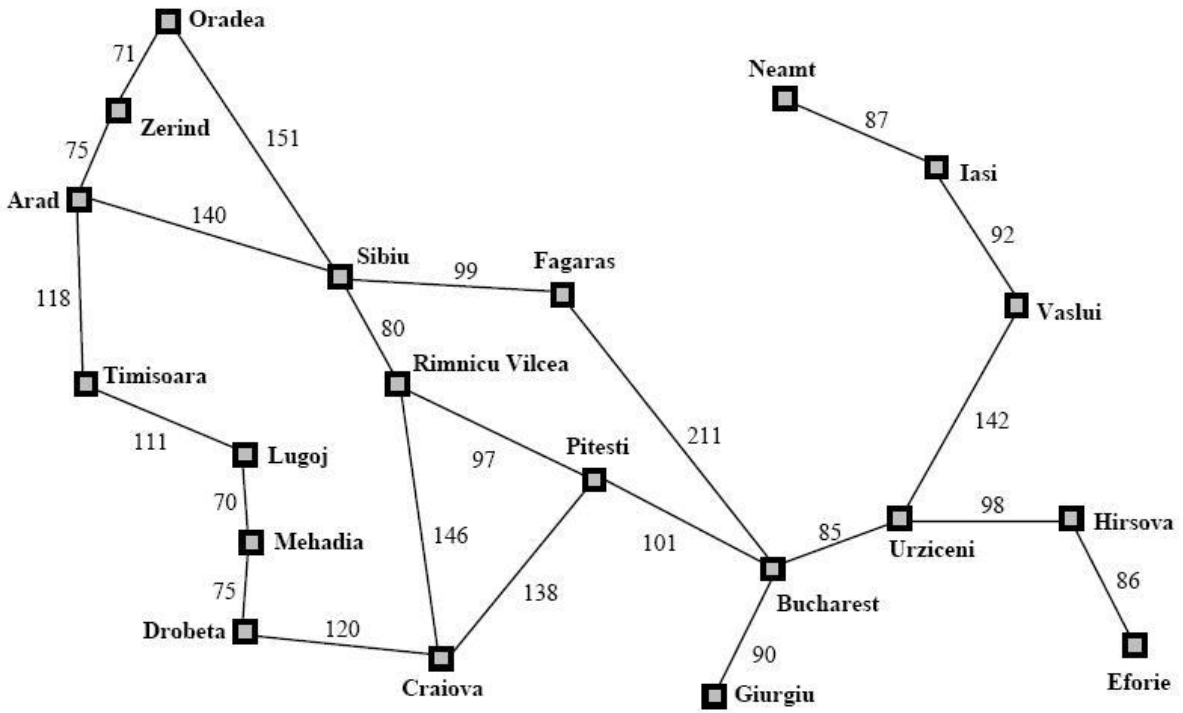
Per rappresentare la sequenza delle operazioni da effettuare si utilizza un albero.

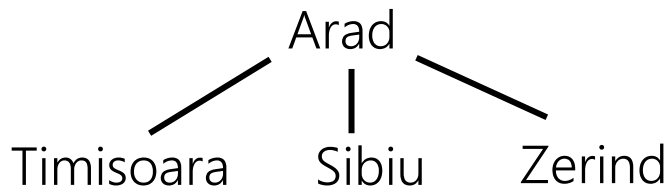
- Un **nodo** rappresenta uno stato *ammissibile* dello *spazio degli stati*

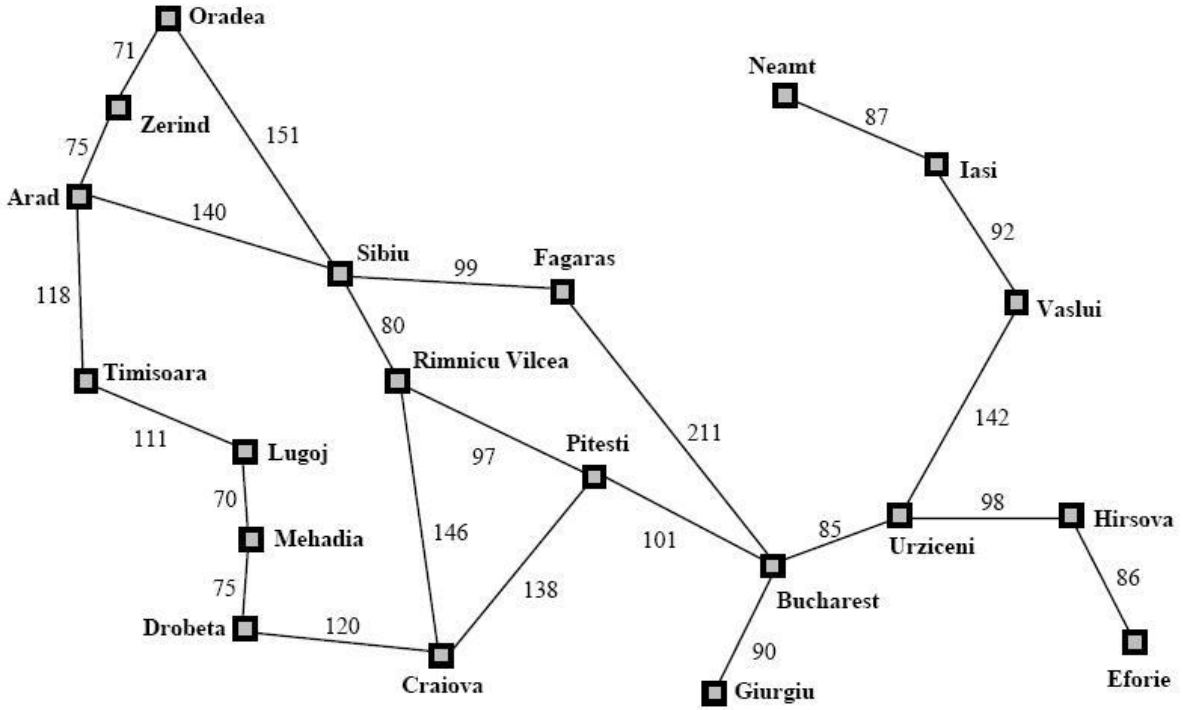
Rappresentazione

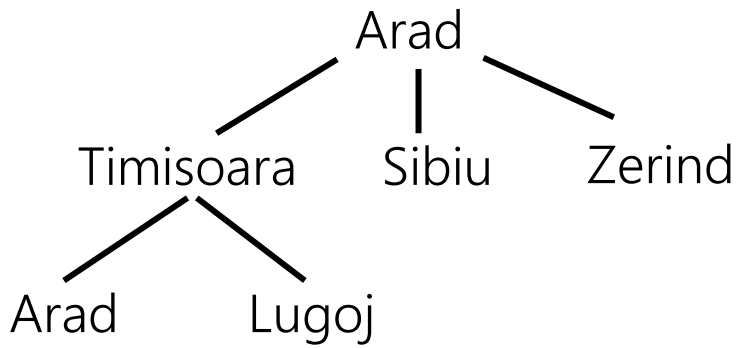
Per rappresentare la sequenza delle operazioni da effettuare si utilizza un albero.

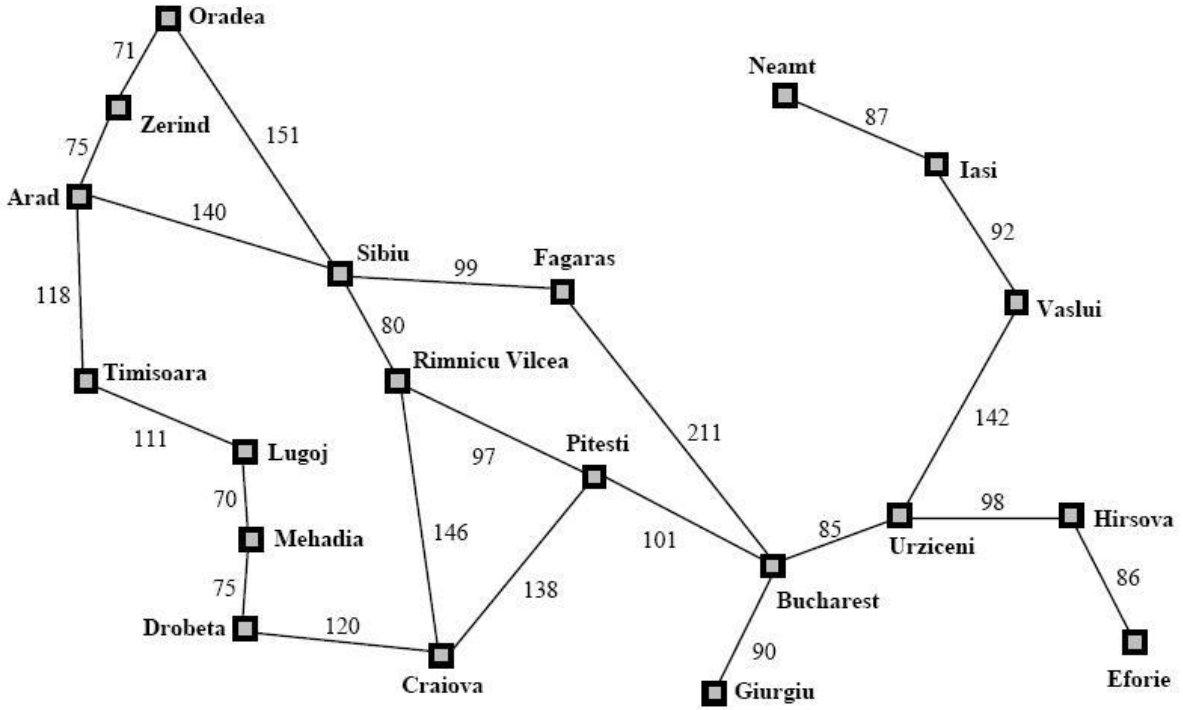
- Un **nodo** rappresenta uno stato *ammissibile* dello *spazio degli stati*
- Un **arco** rappresenta una delle *possibili azioni* che è possibile effettuare dallo stato corrente

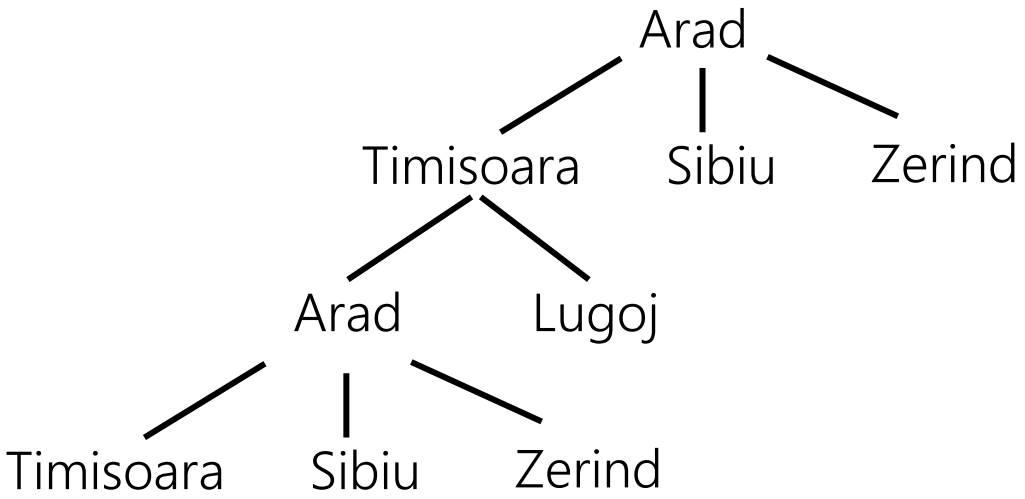

Oradea
71
Zerind
151
75
Arad
140
Sibiu
99
Fagaras
118
80
Rimnicu Vilcea
Timisoara
111
Lugoj
97
Pitesti
211
70
Mehadia
146
101
85
Urziceni
98
Hirsova
75
138
Bucharest
86
Drobeta
120
Craiova
90
Giurgiu
Eforie
Neamt
87
Iasi
92
Vaslui
142

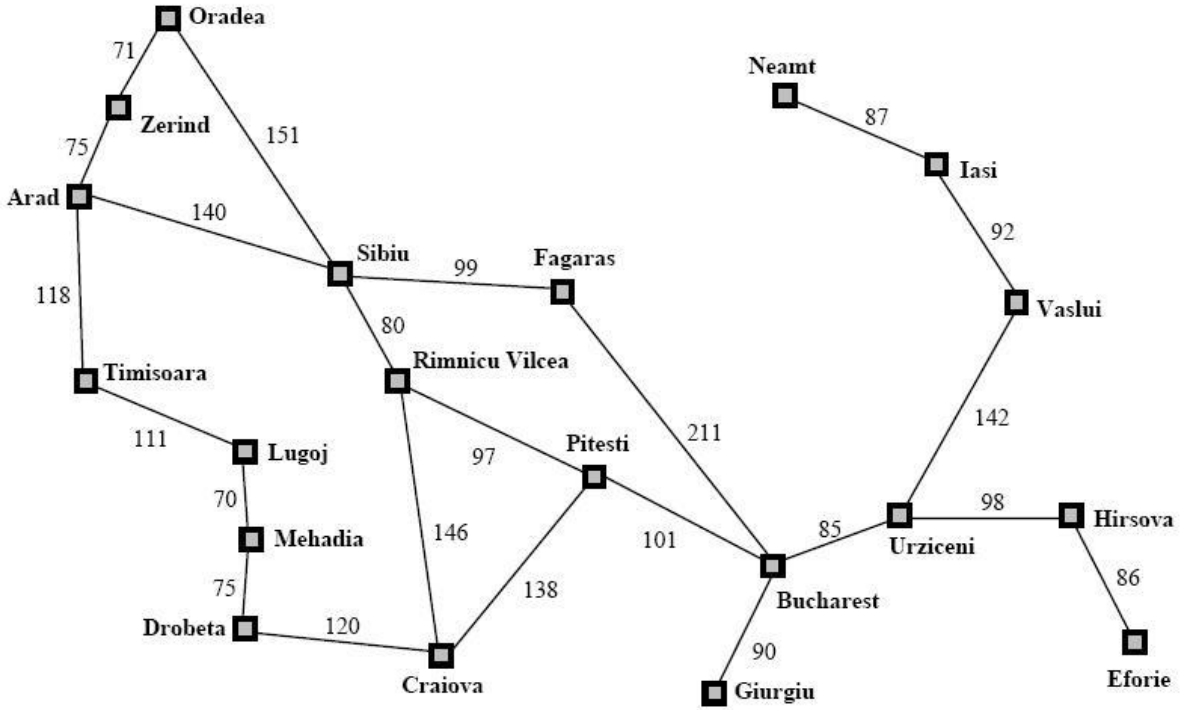

Arad
Timisoara
Sibiu
Zerind

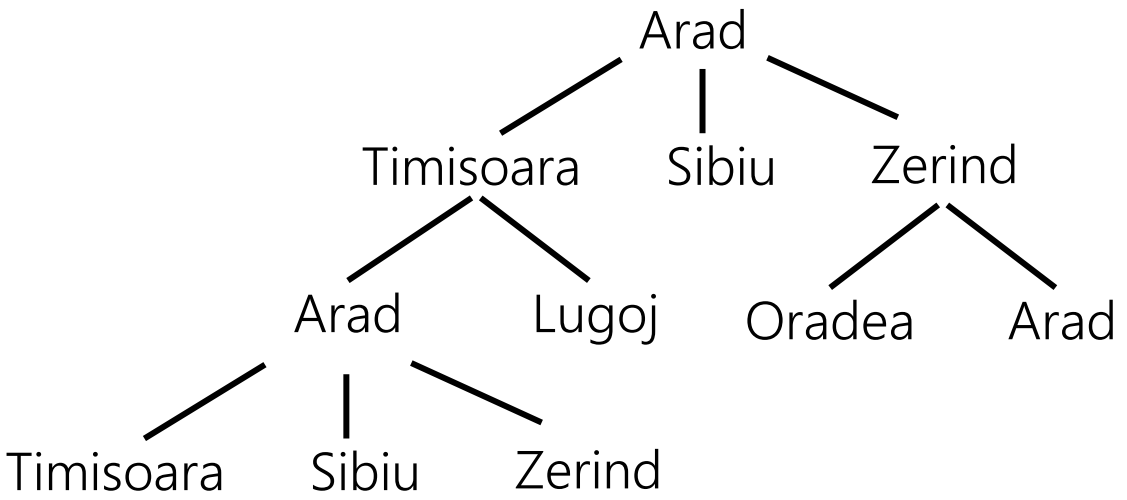

Oradea
71
Zerind
151
75
Arad
140
Sibiu
99
Fagaras
118
80
Rimnicu Vilcea
Timisoara
111
Lugoj
97
Pitesti
211
70
Mehadia
146
101
85
Urziceni
98
Hirsova
75
138
Bucharest
86
Drobeta
120
Craiova
90
Giurgiu
Eforie
Neamt
87
Iasi
92
Vaslui
142


Arad
Timisoara
Sibiu
Zerind
Arad
Lugoj


Oradea
71
Zerind
151
75
Arad
140
Sibiu
99
Fagaras
118
80
Rimnicu Vilcea
Timisoara
111
Lugoj
97
Pitesti
211
70
Mehadia
146
101
75
138
Drobeta
120
Craiova
Neamt
87
Iasi
92
Vaslui
142
85
Urziceni
98
Hirsova
Bucharest
86
90
Giurgiu
Eforie


Arad
Timisoara
Sibiu
Zerind
Arad
Lugoj
Timisoara
Sibiu
Zerind


Oradea
71
Zerind
151
75
Arad
140
Sibiu
99
Fagaras
118
80
Rimnicu Vilcea
Timisoara
111
Lugoj
97
Pitesti
211
70
Mehadia
146
101
75
138
Drobeta
120
Craiova
Neamt
87
Iasi
92
Vaslui
142
85
Urziceni
98
Hirsova
Bucharest
86
90
Giurgiu
Eforie


Arad
Timisoara
Sibiu
Zerind
Arad
Lugoj
Oradea
Arad
Timisoara
Sibiu
Zerind

Obiettivo

Trovare la strada ‘migliore’.

Obiettivo

Trovare la strada ‘migliore’.

Caratteristiche salvabili in un nodo

$$\langle stato, genitore, operatore, profondità, costo\ parziale, \dots \rangle$$

Algoritmo Tree-search

```
function tree-search (problema, strategia) returns soluzione o fallimento
```

Algoritmo Tree-search

```
function tree-search (problema, strategia) returns soluzione o fallimento
```

1. Verifica se la frontiera è vuota

Algoritmo Tree-search

```
function tree-search (problema, strategia) returns soluzione o fallimento
```

1. Verifica se la frontiera è vuota
   - se è vuota, solleva un fallimento.
     se c'è almeno un elemento, passa al punto 2

Algoritmo Tree-search

```
function tree-search (problema, strategia) returns soluzione o fallimento
```

1. Verifica se la frontiera è vuota
   - se è vuota, solleva un fallimento.
     se c'è almeno un elemento, passa al punto 2
2. Scegli uno dei nodi della frontiera

Algoritmo Tree-search

```
function tree-search (problema, strategia) returns soluzione o fallimento
```

1. Verifica se la frontiera è vuota
   - se è vuota, solleva un fallimento.
     se c'è almeno un elemento, passa al punto 2
2. Scegli uno dei nodi della frontiera
   - il modo in cui è scelto dipende dalla strategia
   - se il nodo scelto è il nodo obiettivo, ricostruisci il cammino. Altrimenti aggiungi i nodi foglia.

Algoritmo Tree-search

```
function tree-search (problema, strategia) returns soluzione o fallimento
```

1. Verifica se la frontiera è vuota
   - se è vuota, solleva un fallimento.
     se c'è almeno un elemento, passa al punto 2
2. Scegli uno dei nodi della frontiera
   - il modo in cui è scelto dipende dalla strategia
   - se il nodo scelto è il nodo obiettivo, ricostruisci il cammino. Altrimenti aggiungi i nodi foglia.
3. Ritorna al punto 1

Strategia nella Tree-search

La strategia deve essere valutata in termini di

Strategia nella Tree-search

La strategia deve essere valutata in termini di

- COMPLESSITÀ
  - spaziale e temporale

Strategia nella Tree-search

La strategia deve essere valutata in termini di

- COMPLESSITÀ
  - spaziale e temporale
- COMPLETEZZA

Strategia nella Tree-search

La strategia deve essere valutata in termini di

- COMPLESSITÀ
  - spaziale e temporale
- COMPLETEZZA
- OTTIMALITÀ

Ricerca in ampiezza

Espande i nodi a livelli.

Ricerca in ampiezza

Espande i nodi a livelli.

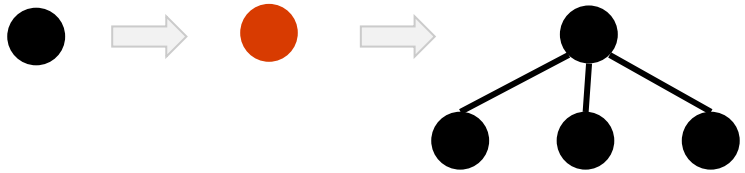
Prima tutti i figli della radice, poi tutti i figli del primo figlio.
Poi tutti i figli del secondo figlio della radice e così via

Esempio Ricerca in ampiezza

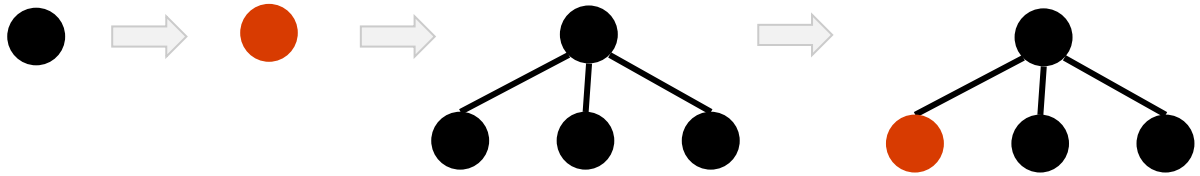
-

Esempio Ricerca in ampiezza

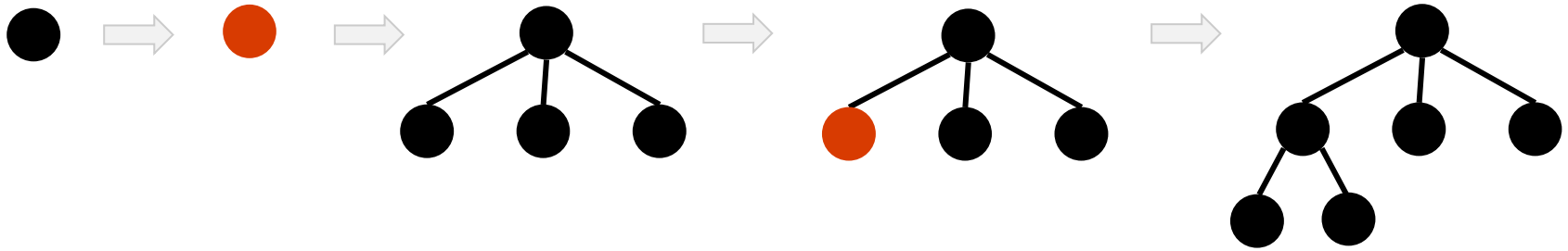
Esempio Ricerca in ampiezza



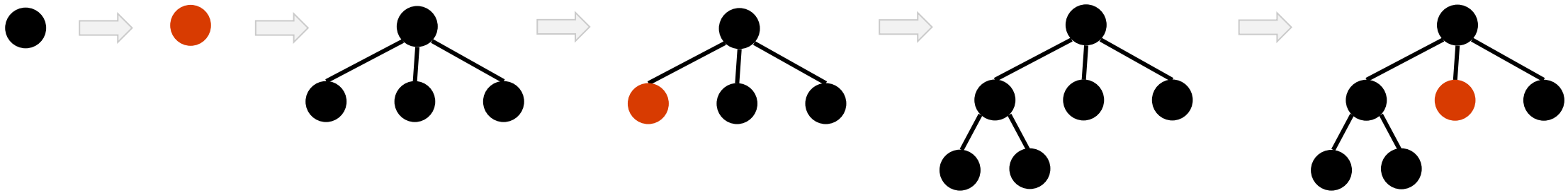
Esempio Ricerca in ampiezza



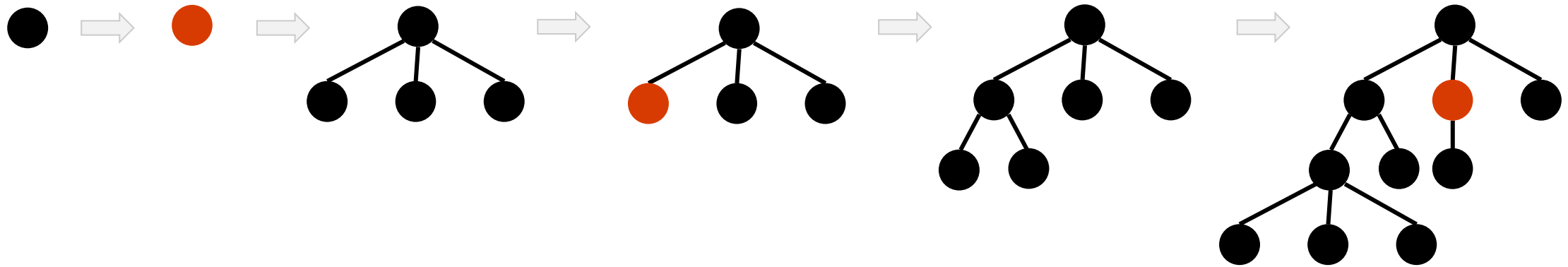
Esempio Ricerca in ampiezza

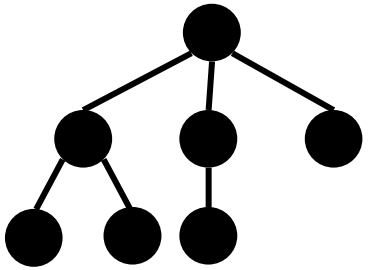


Esempio Ricerca in ampiezza

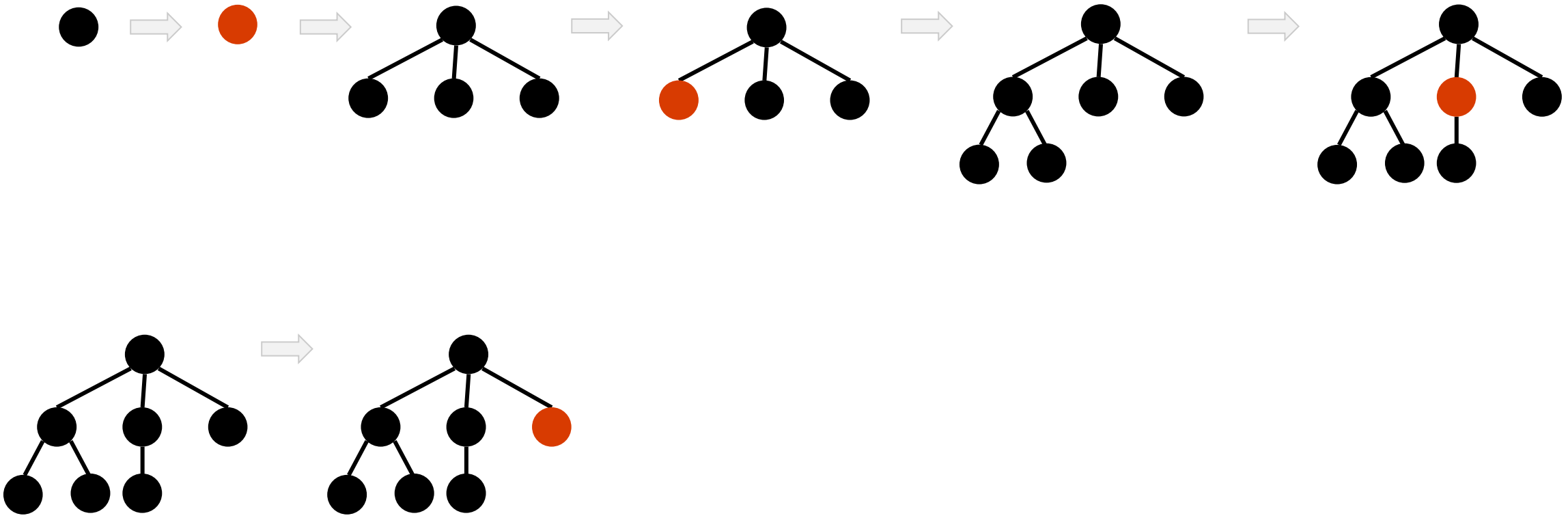


Esempio Ricerca in ampiezza

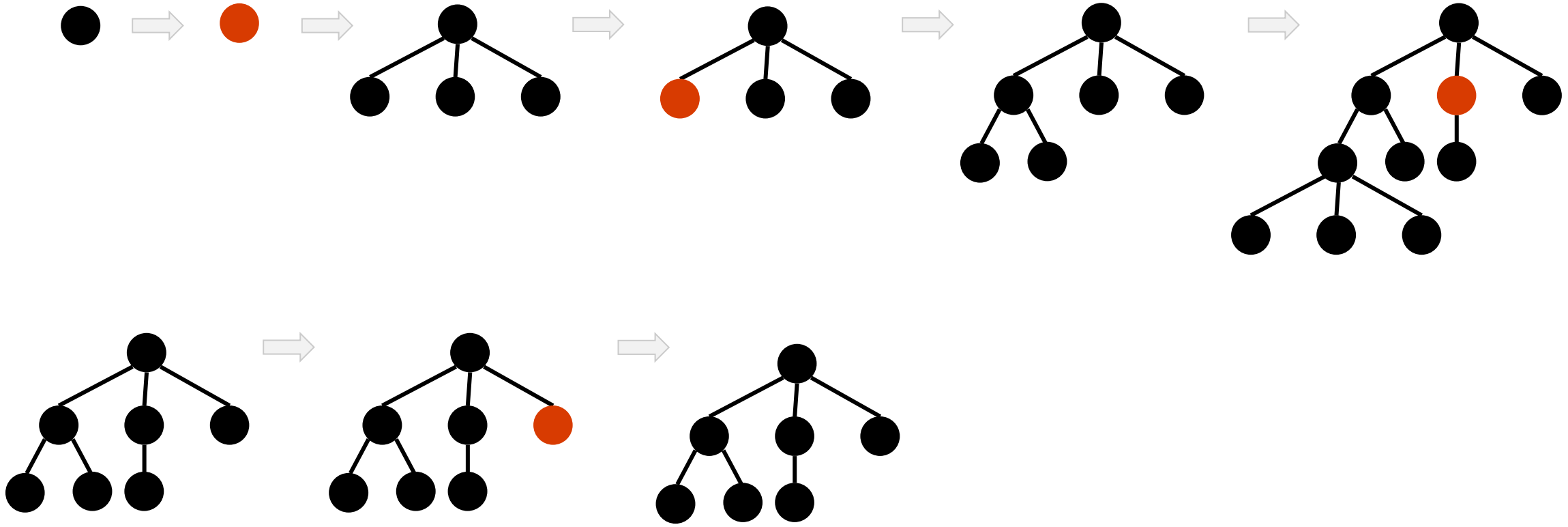




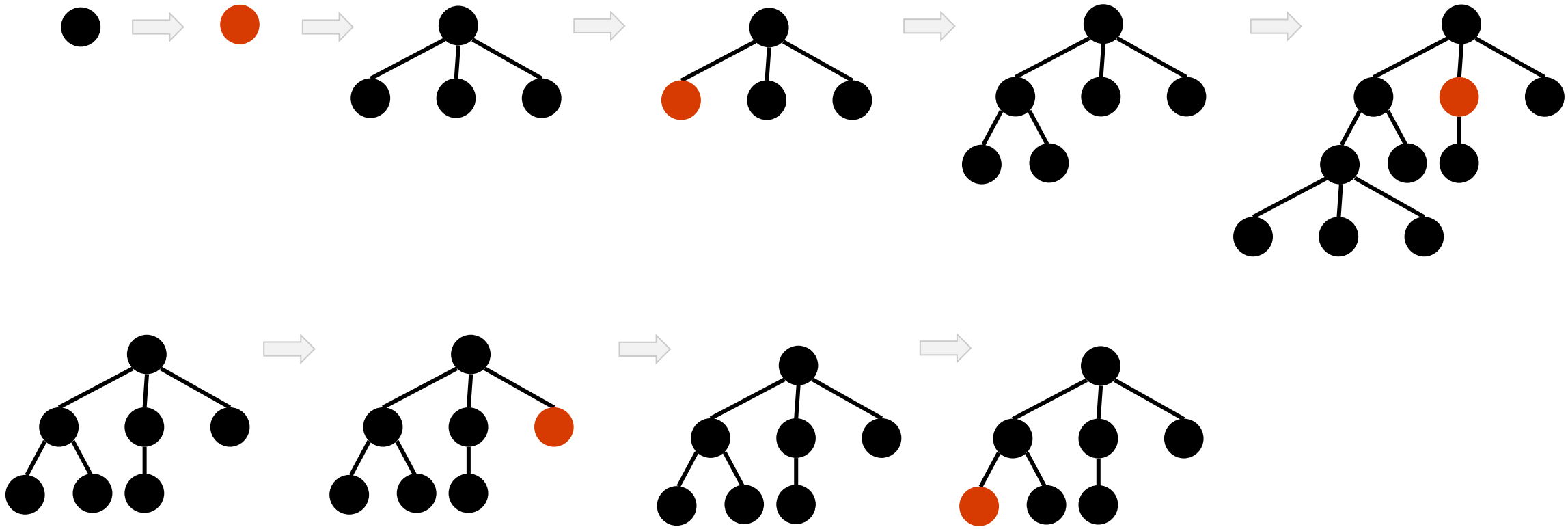
Esempio Ricerca in ampiezza



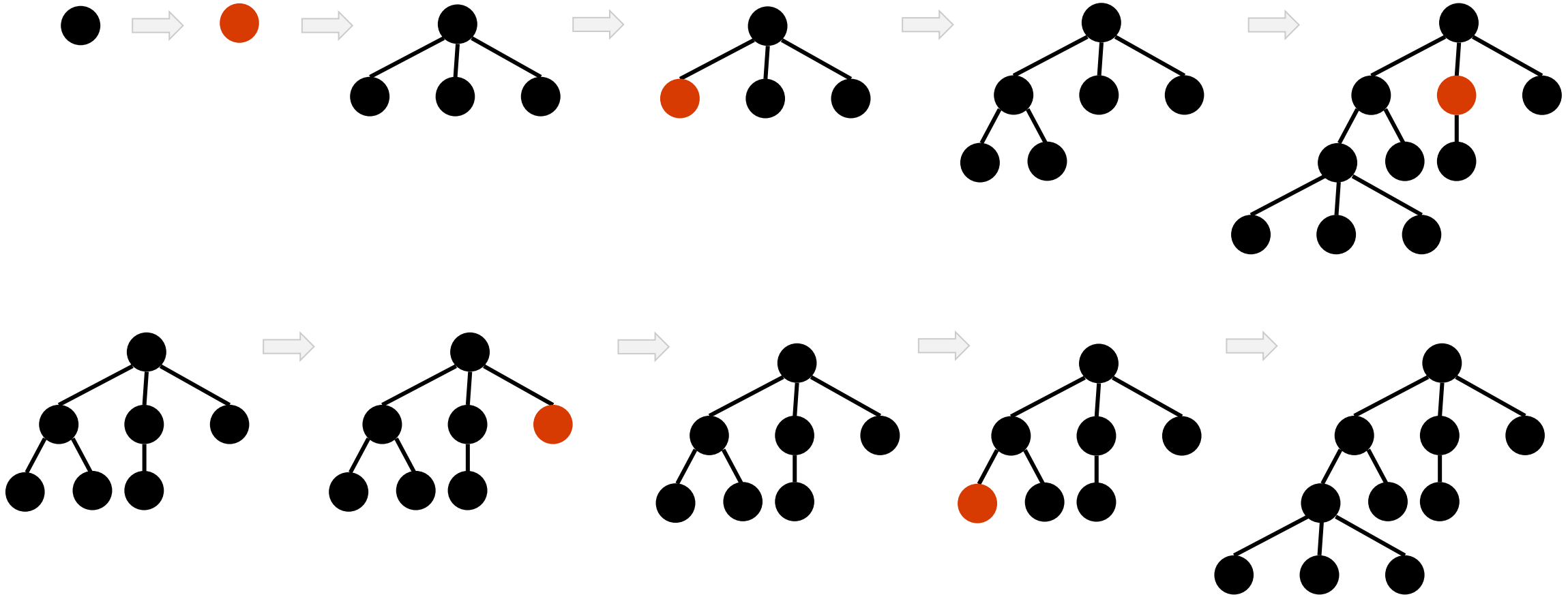
Esempio Ricerca in ampiezza



Esempio Ricerca in ampiezza



Esempio Ricerca in ampiezza



Caratteristiche Ricerca in ampiezza

COMPLETEZZA	Sì
OTTIMALITÀ	Non necessariamente. È ottimo se il costo del cammino è una funzione $monotona$ non decrescente della profondità nel nodo $$depth(n) < depth(m) \rightarrow path.cost(n) \leq path.cost(m)$$
COMPLESSITÀ TEMPORALE	$$O(b^d)$$ b è il fattore di ramificazione d è la lunghezza minima di un cammino dal nodo iniziale alla soluzione Ipotizziamo di trovarci a una profondità d e la soluzione è l'ultimo nodo a destra. Per arrivare a visitare questo nodo devo aver visitato tutti i nodi in precedenza che sono $1 + b + b^2 + \cdots + (b^{d+1} - b)$
COMPLESSITÀ IN SPAZIO	$$O(b^d)$$ Perché tutte le foglie sono salvate in memoria.


DEMO

Ricerca guidata dal costo

Tra i nodi della frontiera espande prima quello che ha costo parziale *g(n)* minore.

Ricerca guidata dal costo

Tra i nodi della frontiera espande prima quello che ha costo parziale *g(n)* minore.

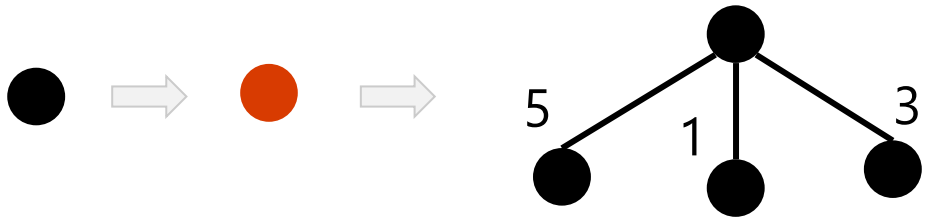
NOTA: se *g(n)=depth(n)* si ha la ricerca in ampiezza

Esempio Ricerca guidata dal costo

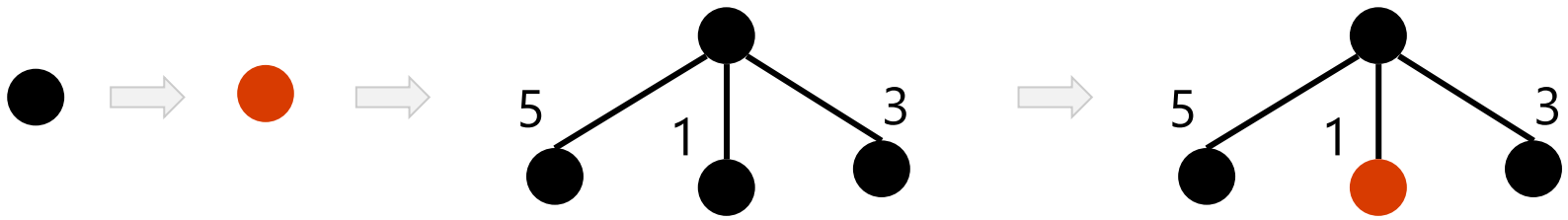
-

Esempio Ricerca guidata dal costo

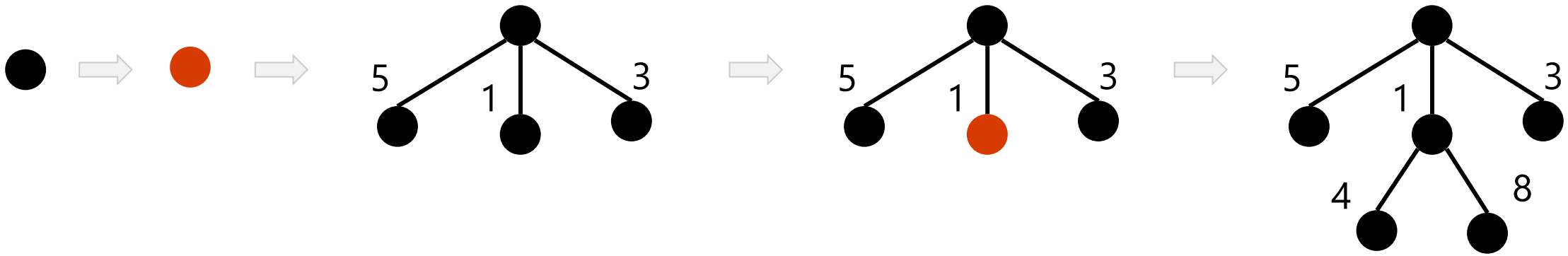
Esempio Ricerca guidata dal costo



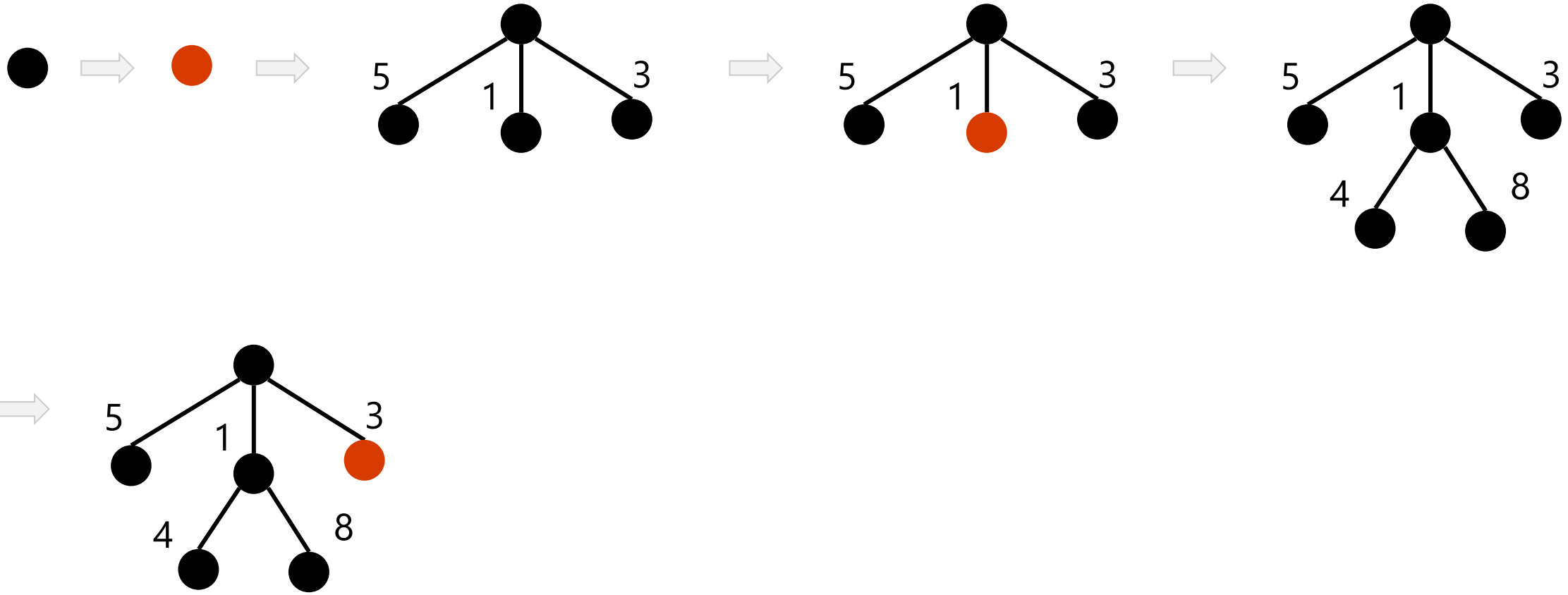
Esempio Ricerca guidata dal costo



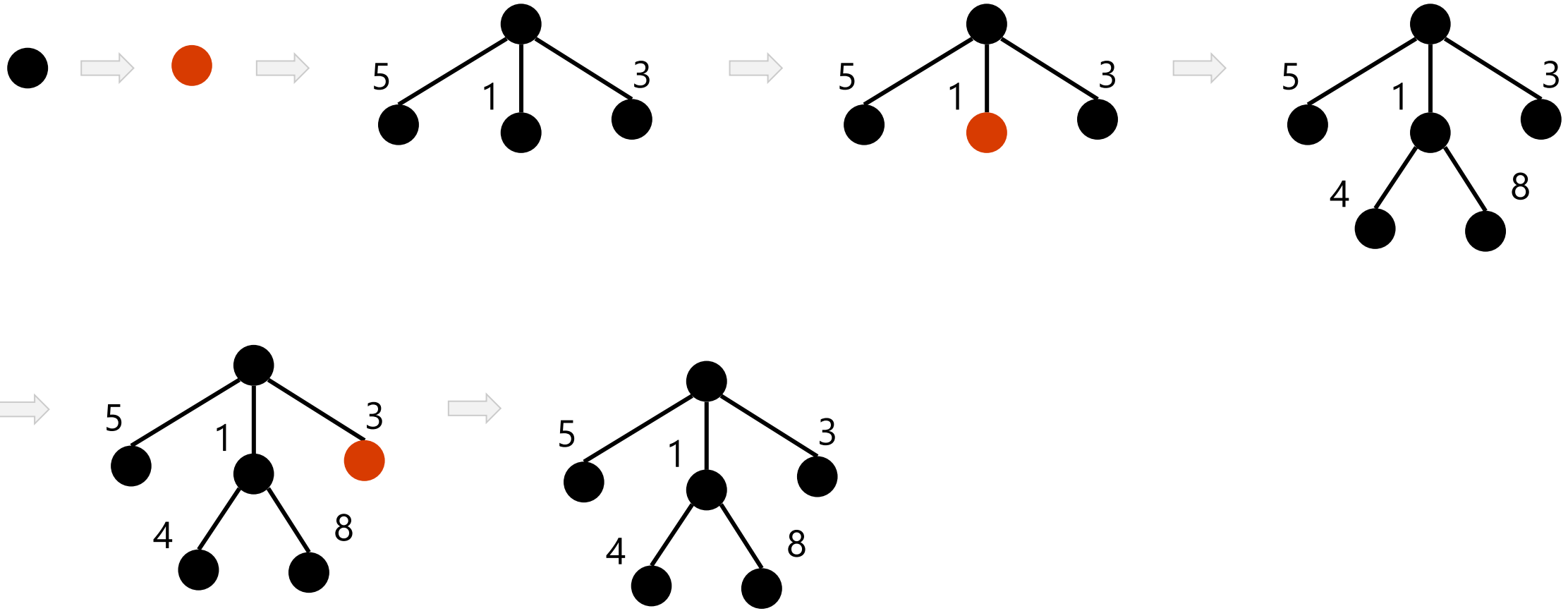
Esempio Ricerca guidata dal costo



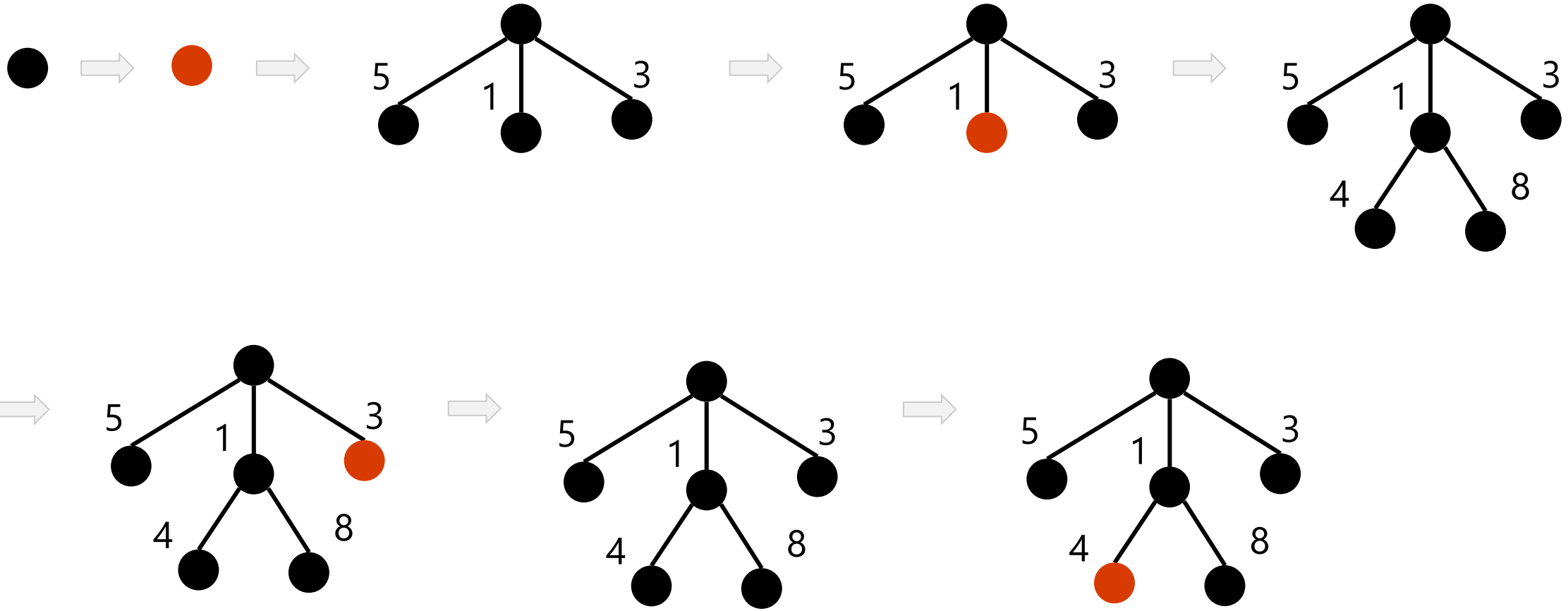
Esempio Ricerca guidata dal costo



Esempio Ricerca guidata dal costo



Esempio Ricerca guidata dal costo



Caratteristiche Ricerca guidata dal costo

COMPLETEZZA	Sì
OTTIMALITÀ	Non necessariamente. È ottimo se il costo di ogni step $g\ (SUCCESSOR(n)) - g(n)$ è sempre maggiore o uguale di una costante positiva piccola ε
COMPLESSITÀ TEMPORALE	$O(b^d)$ b è il fattore di ramificazione d è la lunghezza minima di un cammino dal nodo iniziale alla soluzione Ipotizziamo di trovarci a una profondità d e la soluzione è l'ultimo nodo a destra. Per arrivare a visitare questo nodo devo aver visitato tutti i nodi in precedenza che sono $1 + b + b^2 + \cdots + (b^{d+1} - b)$
COMPLESSITÀ IN SPAZIO	$O(b^d)$ Perché tutte le foglie sono salvate in memoria.

Ricerca in profondità

Una volta ‘scelta’ una strada la percorre fino a che è possibile.

Ricerca in profondità

Una volta ‘scelta’ una strada la percorre fino a che è possibile.

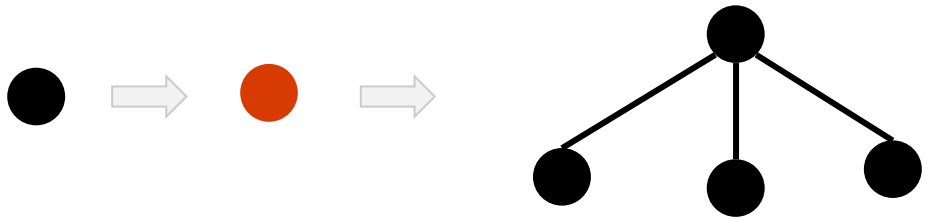
Solo dopo aver raggiunto un fallimento ‘torna indietro’ a provare una seconda strada.

Esempio Ricerca in profondità

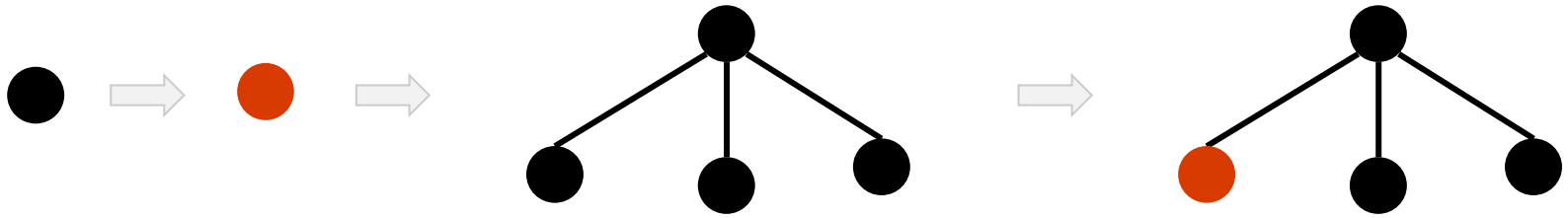
-

Esempio Ricerca in profondità

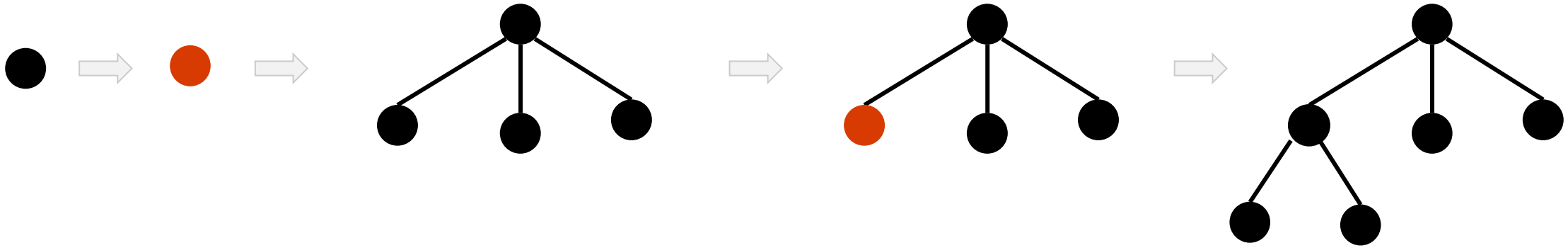
Esempio Ricerca in profondità



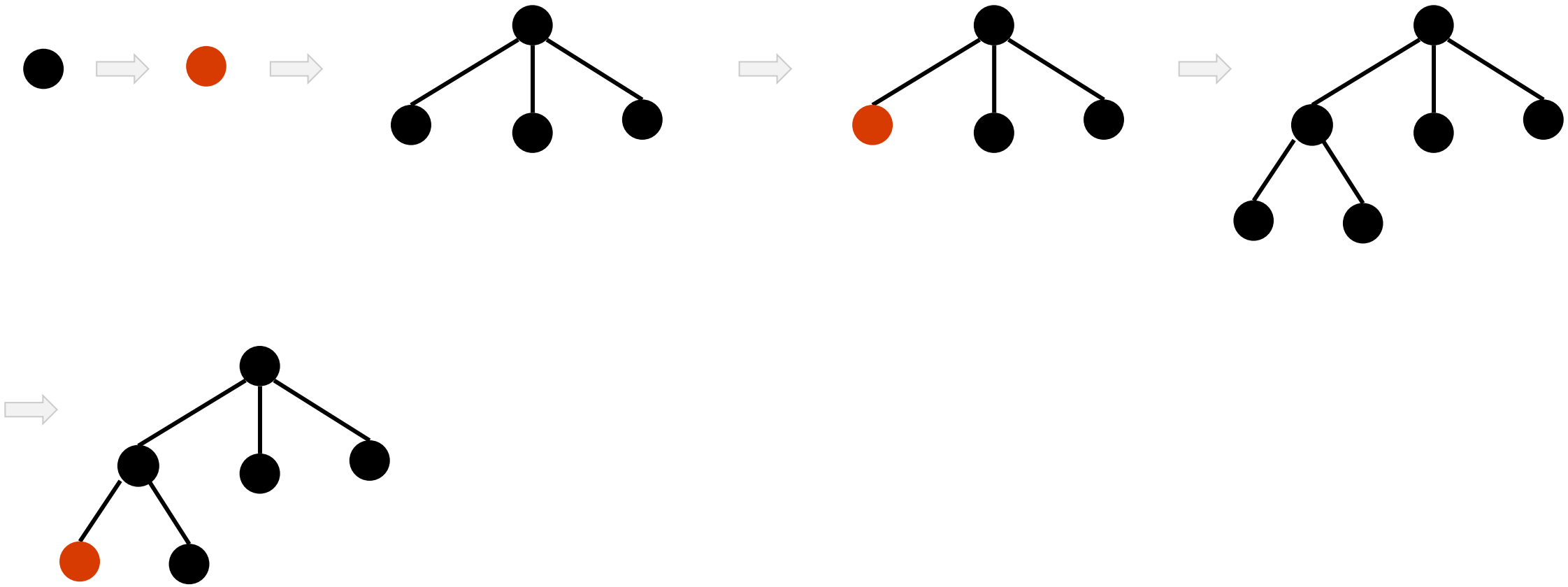
Esempio Ricerca in profondità



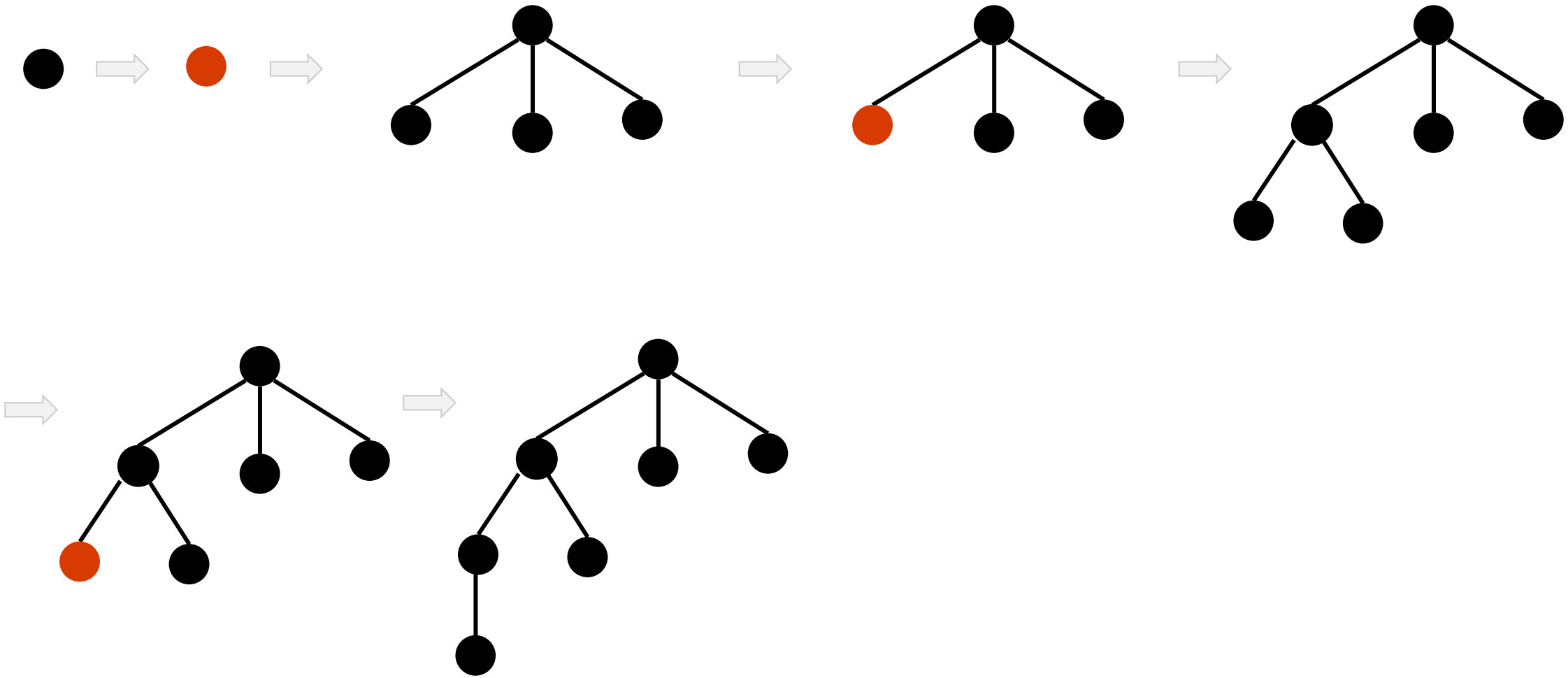
Esempio Ricerca in profondità



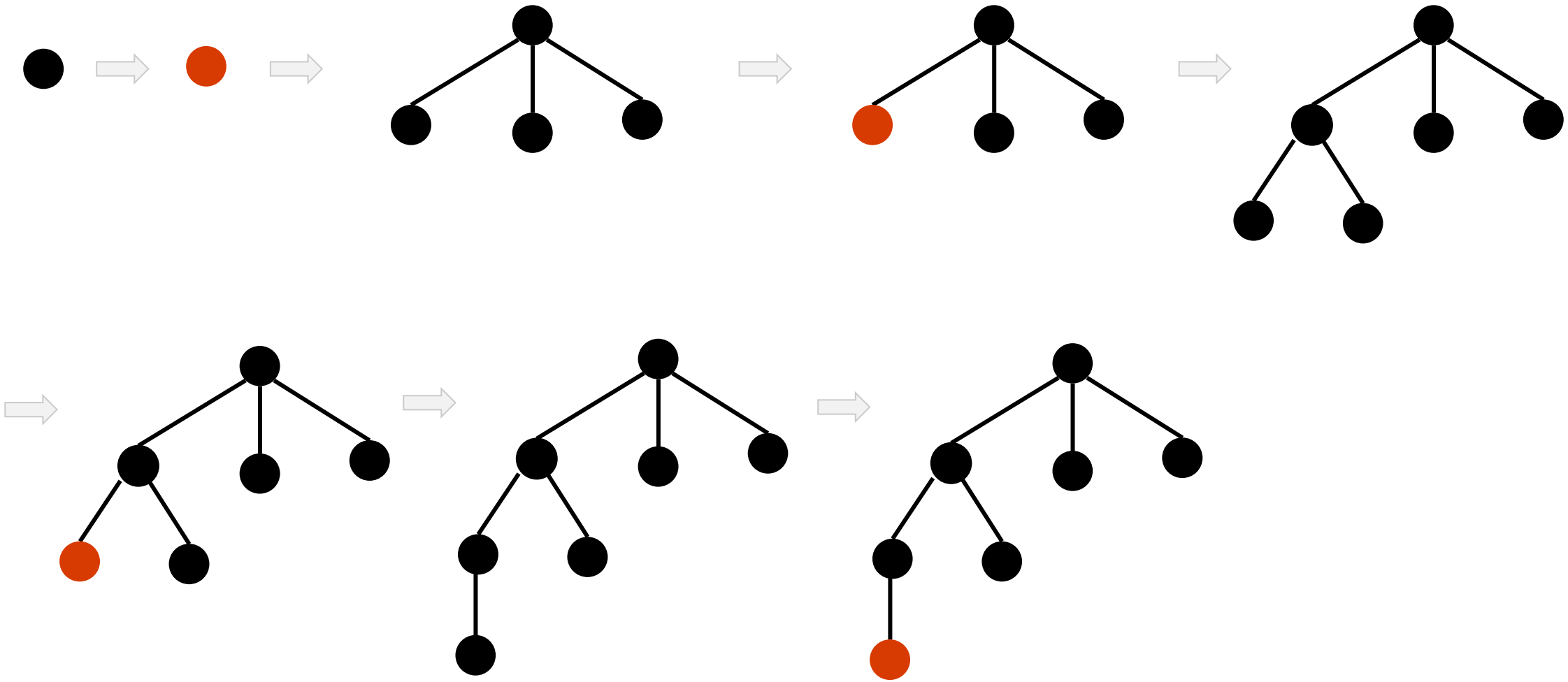
Esempio Ricerca in profondità



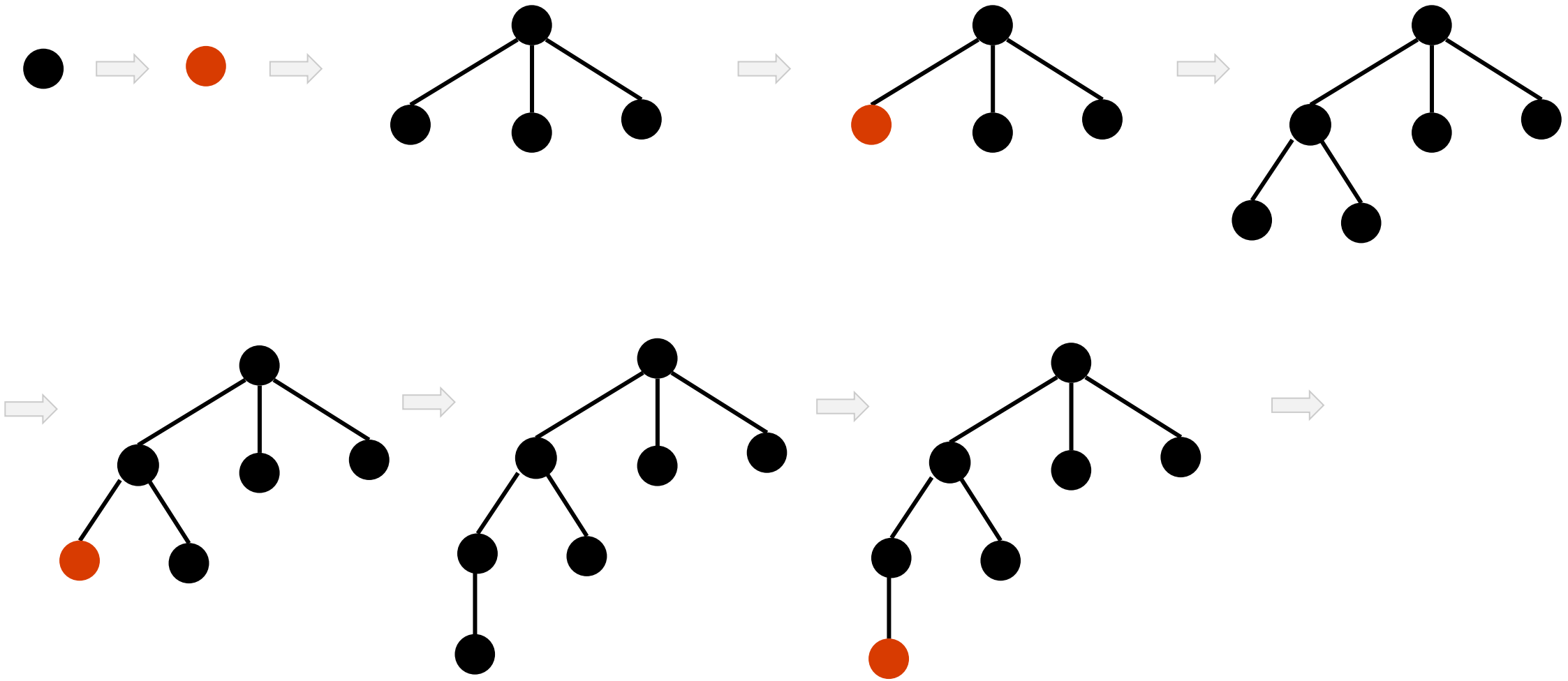
Esempio Ricerca in profondità



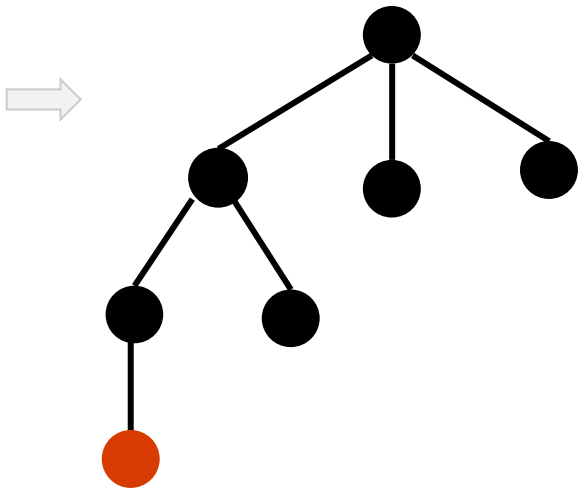
Esempio Ricerca in profondità



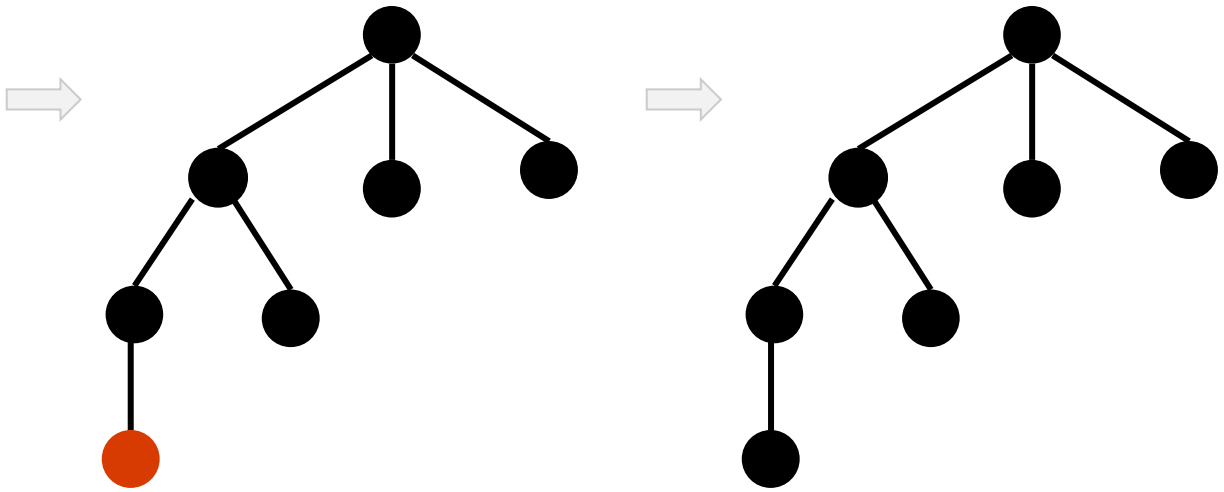
Esempio Ricerca in profondità



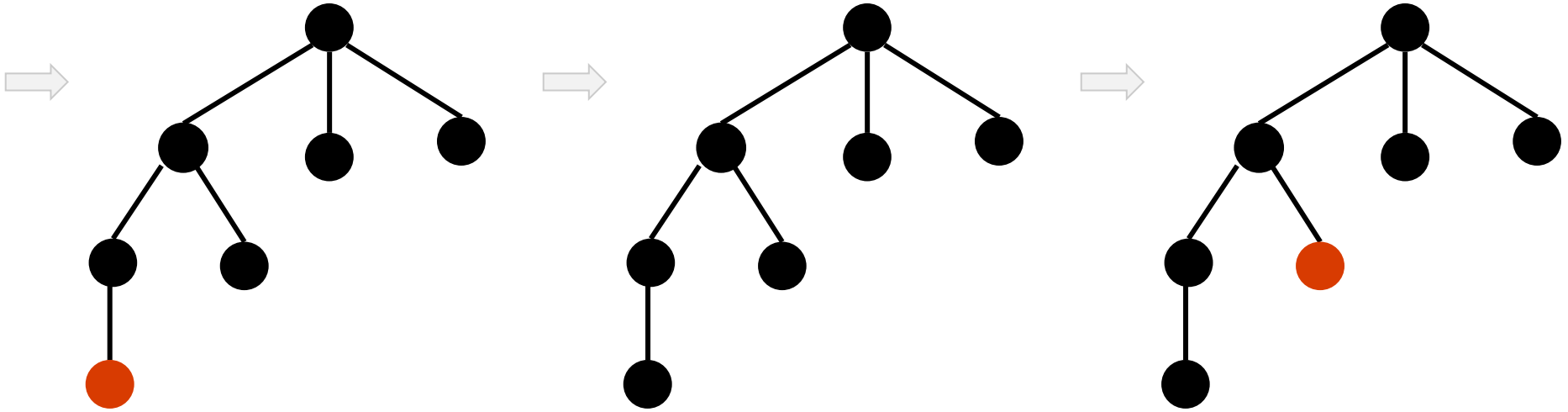
Esempio Ricerca in profondità



Esempio Ricerca in profondità



Esempio Ricerca in profondità



Caratteristiche Ricerca in profondità

COMPLETEZZA	No. Potrei avere un sottoalbero destro di dimensione infinita e la soluzione non si trova su questo
OTTIMALITÀ	Non necessariamente. Potrei avere due soluzioni: una nel primo sottoalbero e una nel secondo. Quest'ultima è ottima. L'algoritmo individua prima quella non ottima.
COMPLESSITÀ TEMPORALE	$O(b^m)$ b è il fattore di ramificazione m è la profondità massima dell'albero di ricerca La prima soluzione potrebbe trovarsi nel sottoalbero più profondo.
COMPLESSITÀ IN SPAZIO	$O(b \cdot m)$ b è il fattore di ramificazione m è la profondità massima dell'albero di ricerca La soluzione si trova a profondità m e di ogni livello ho al più b figli. In totale ho al più $b \cdot m$ nodi in memoria.


DEMO

Varianti della Ricerca in profondità

- PROFONDITÀ LIMITATA
- PROFONDITÀ LIMITATA ITERATIVA

Ricerca in profondità LIMITATA

Si decide di non espandere più i nodi a una profondità maggiore di *k*.

Ricerca in profondità LIMITATA

Si decide di non espandere più i nodi a una profondità maggiore di *k*.

GUADAGNO: l'algoritmo potrebbe essere completo se la soluzione ha una lunghezza minore o uguale a *k*

Ricerca in profondità LIMITATA ITERATIVA

Non conoscendo *k* a priori si effettuano più cicli.
A ogni ciclo si incrementa *k*.

Ricerca in profondità LIMITATA ITERATIVA

Non conoscendo *k* a priori si effettuano più cicli.
A ogni ciclo si incrementa *k*.

GUADAGNO: l'algoritmo è completo.

Ricerca in profondità LIMITATA ITERATIVA

Non conoscendo *k* a priori si effettuano più cicli.
A ogni ciclo si incrementa *k*.

GUADAGNO: l'algoritmo è completo.
GUADAGNO: è ottimo se il costo del cammino è funzione *monotona non decrescente* della profondità del nodo.

Tree search vs Graph search

L’approccio appena visto soffre di un grave problema: non effettua nessuno controllo sugli stati già visitati.

Tree search vs Graph search

L’approccio appena visto soffre di un grave problema: non effettua nessuno controllo sugli stati già visitati.

Potrei trasformare un problema *lineare* in uno *esponenziale*.ù

Tree search vs Graph search

L'approccio appena visto soffre di un grave problema: non effettua nessuno controllo sugli stati già visitati.

Potrei trasformare un problema *lineare* in uno *esponenziale*.

Devo tener conto degli stati già visitati e non rivisitarli.

Tree search vs Graph search

ATTENZIONE.
Ragionamento, in generale, errato.

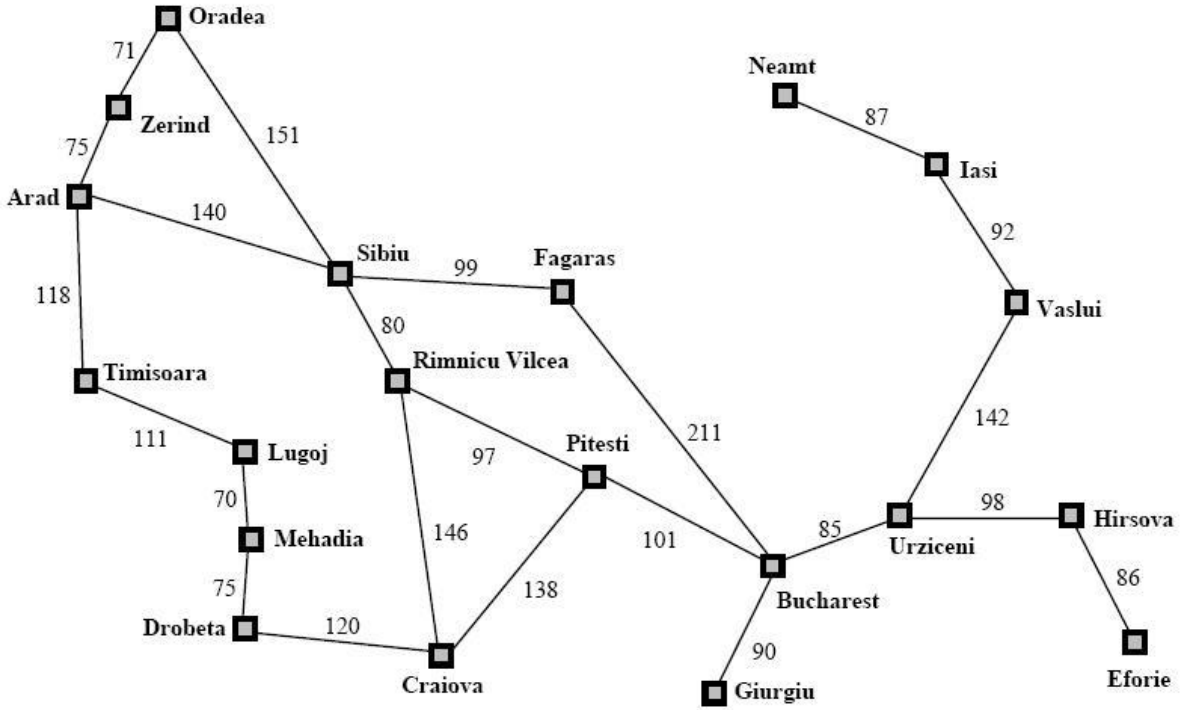
Tree search vs Graph search

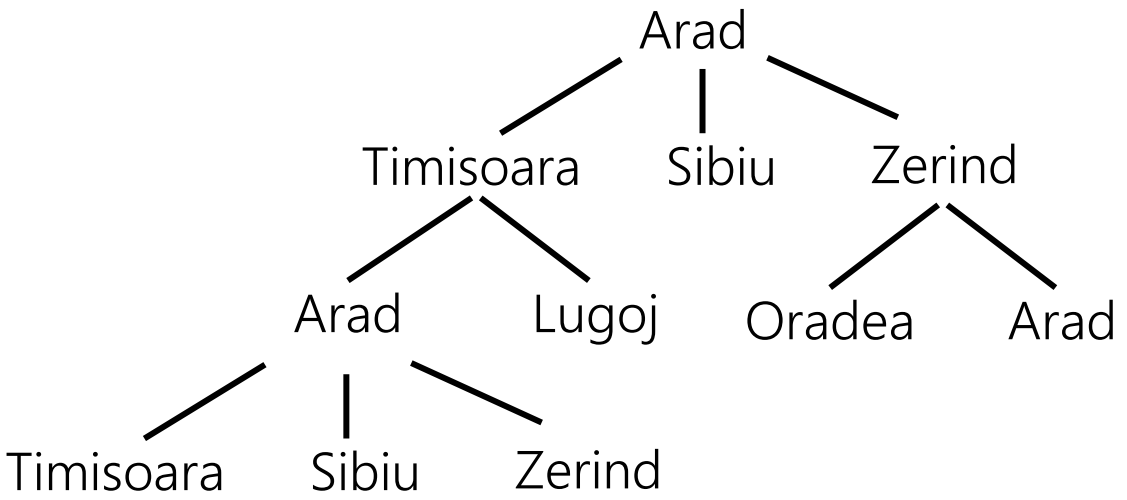
ATTENZIONE.
Ragionamento, in generale, errato.

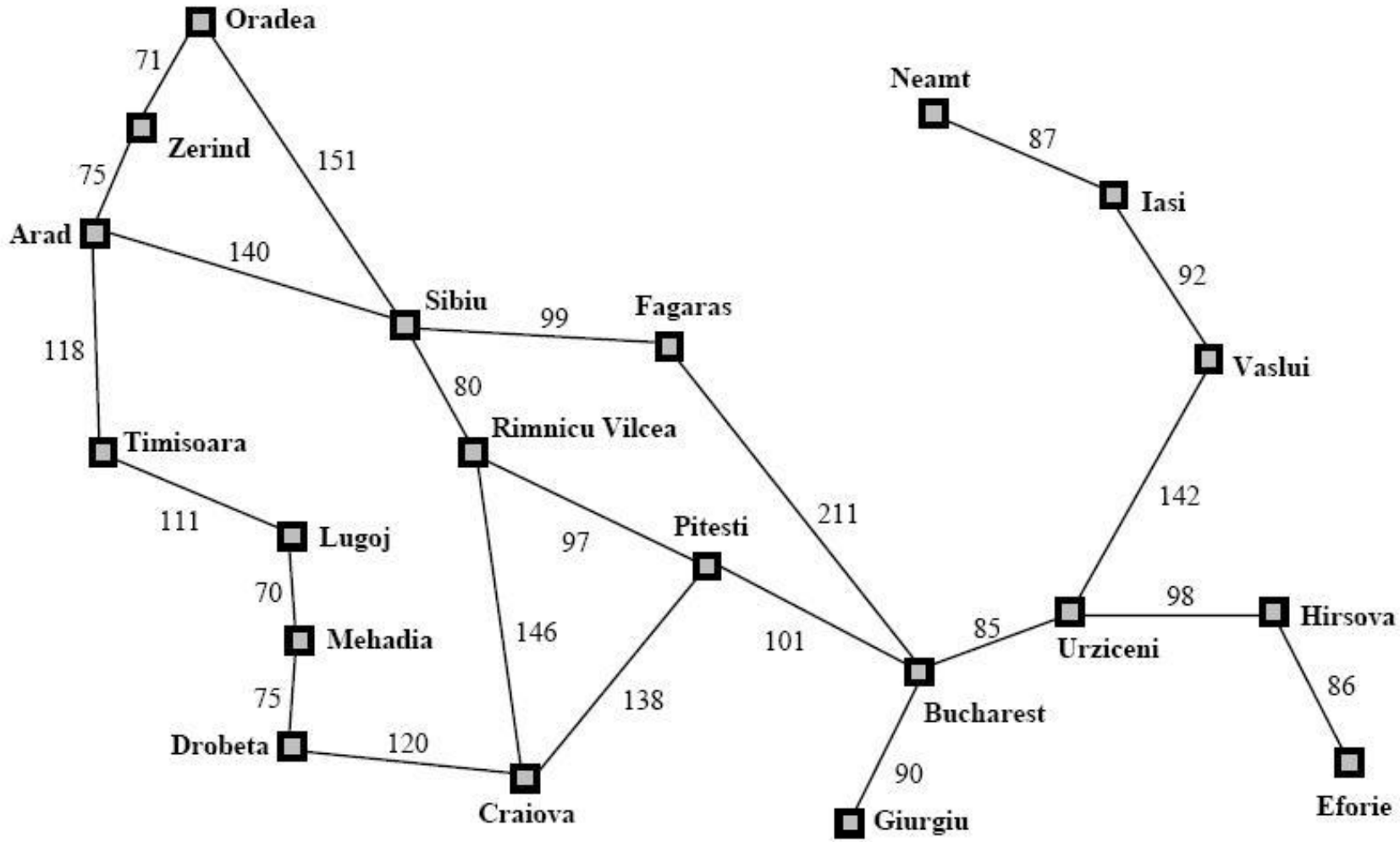
Visito per la seconda volta un nodo.
Scarto il cammino che mi ci ha portato.
NON ho informazioni per dire che il secondo cammino sia peggiore del primo.

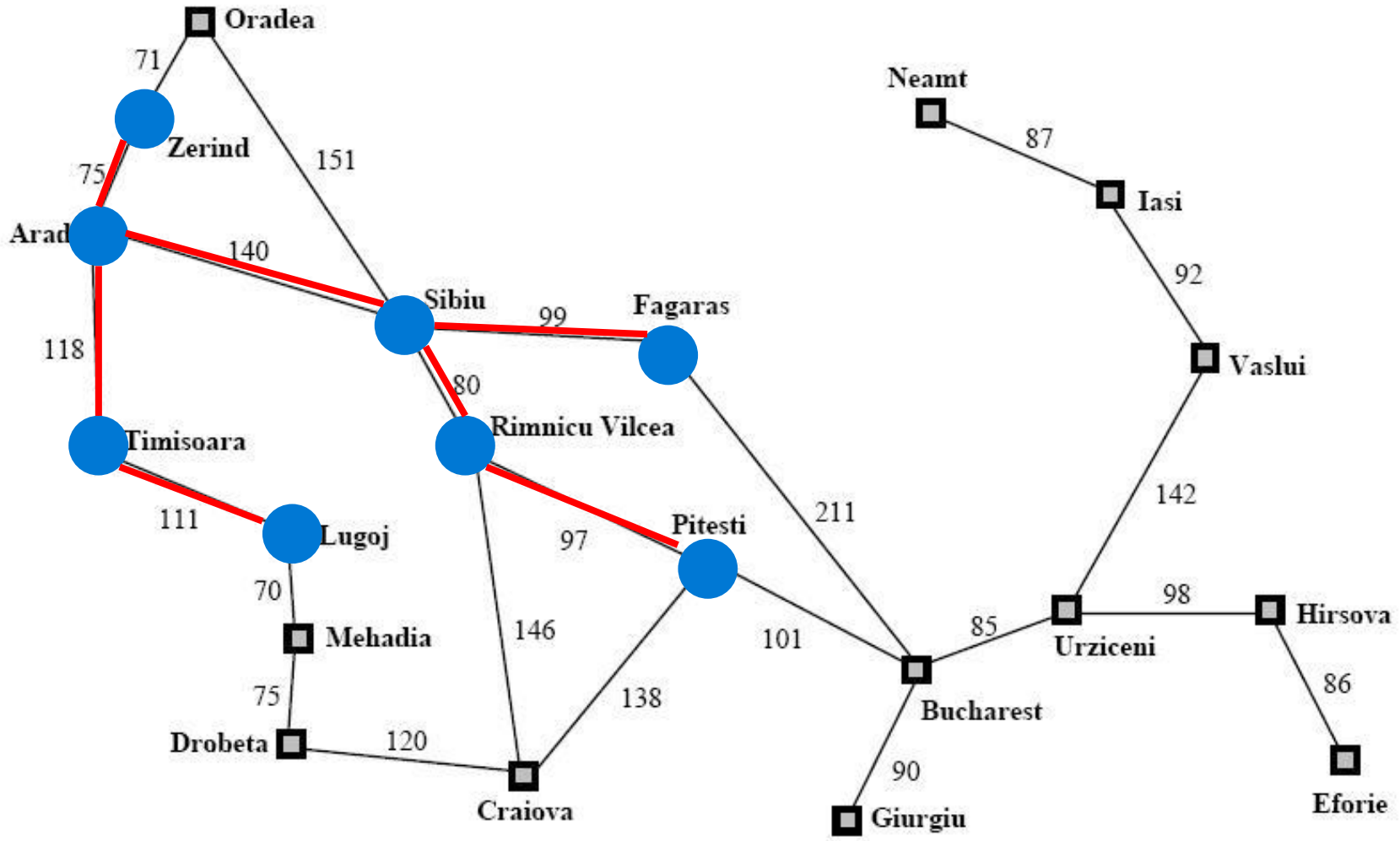
Graph Search

L’idea è quella di costruire un albero di ricerca direttamente sullo *spazio* degli stati


Oradea
71
Zerind
151
75
Arad
140
Sibiu
99
Fagaras
118
80
Rimnicu Vilcea
Timisoara
111
Lugoj
97
Pitesti
211
70
Mehadia
146
101
75
138
Drobeta
120
Craiova
Neamt
87
Iasi
92
Vaslui
142
85
Urziceni
98
Hirsova
Bucharest
86
90
Giurgiu
Eforie


Arad
Timisoara
Sibiu
Zerind
Arad
Lugoj
Oradea
Arad
Timisoara
Sibiu
Zerind


Oradea
71
Zerind
75
151
Arad
140
Sibiu
99
Fagaras
118
80
Rimnicu Vilcea
Timisoara
111
Lugoj
97
Pitesti
211
70
Mehadia
146
101
75
138
Drobeta
120
Craiova
Neamt
87
Iasi
92
Vaslui
142
98
Hirsova
85
Urziceni
86
Bucharest
90
Giurgiu
Eforie


Oradea
71
Zerind
75
151
Arad
140
Sibiu
99
Fagaras
118
80
Timisoara
Rimnicu Vilcea
111
97
Pitesti
211
Lugoj
70
Mehadia
146
101
75
138
Drobeta
120
Craiova
Neamt
87
Iasi
92
Vaslui
142
98
Hirsova
85
Urziceni
86
Bucharest
90
Giurgiu
Eforie

Prossimo talk

Ciò che abbiamo visto si definisce *ricerca non informata* (o ***cieca***)

Prossimo talk

Ciò che abbiamo visto si definisce *ricerca non informata* (o ***cieca***)

Nel prossimo talk vedremo la *ricerca informata* (o ***euristica***)

Microsoft Learn
Student Ambassadors

RISORSE

github.com/mariocuomo/talks

Grazie