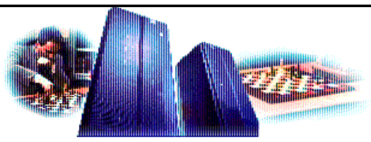
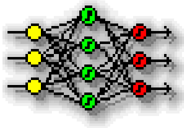


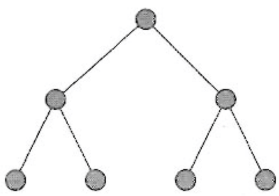
Corso di Laurea in
Scienze cognitive e
Processi decisionali



Intelligenza Artificiale e analisi dei dati
Alberi di decisione

Alberto Borghese
Università degli Studi di Milano
Laboratorio di Sistemi Intelligenti Applicati (AIS-Lab)
Dipartimento di Informatica
Alberto.borghese@unimi.it

Decision trees



Rappresentazione mediante grafo orientato (dall'alto al basso)

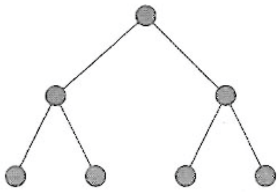
Comparare con lo STG della FSM

DECISION THEORY: Aim is to develop an **action plan** is to build an to achieve a **given goal (control problem)**.

COMPUTER SCIENCE (machine learning): the tree is built from the data, and each arch represents a probability or the learnt cost of the transition.

COMPUTER SCIENCE (data mining): the tree is used to **classify (classification problem).**

How do they work?



We need a sequence of actions (control or incremental classification) to move through the tree.

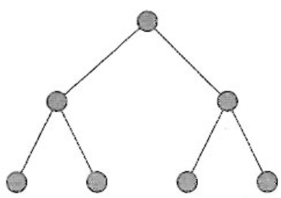
1) Definition of the tree
2) Computation of the path
3) Use of the sequence of action

Formulate-Search-Execute

Nodes and states

A state is a (representation of) a physical configuration, that corresponds to a situation

A node is a data structure constituting part of a search tree includes state, parent node, action, path cost $g(x)$, depth



Example: Romania

- On holiday in Romania; currently in Arad.
- Flight leaves tomorrow from Bucharest

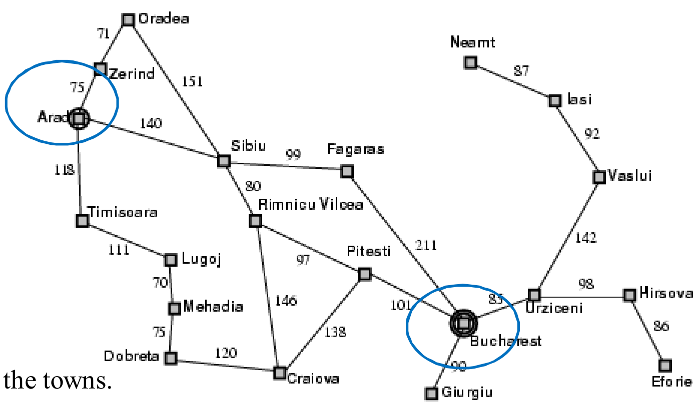
- Formulate goal:
  - be in Bucharest

- Formulate problem:
  - states: various cities
  - actions: drive between cities

- Find solution:
  - sequence of cities, e.g., Arad, Sibiu, Fagaras, Bucharest

Example: route finding - formalization

- From Arad to Bucarest



State: ensemble of the towns.

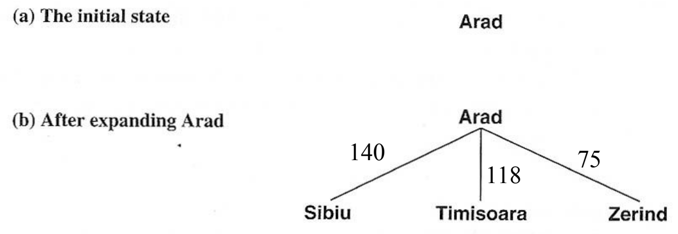
Goal: reach Bucarest.

Action: choice of a possible direction from the actual town.

Cost: length of the path.

Initial state: Arad

Example: route finding – Costruzione dell’albero di ricerca

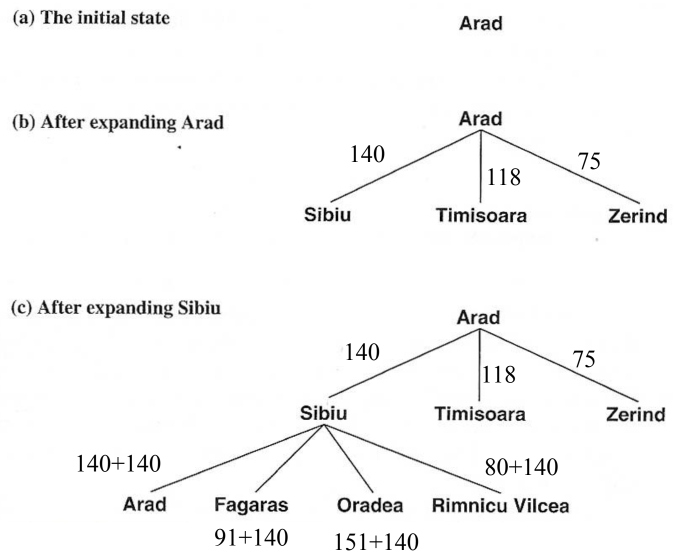


Which action is best for reaching the goal, in each state?

In Arad we have different choices, we have to decide which next town to choose. **How?**

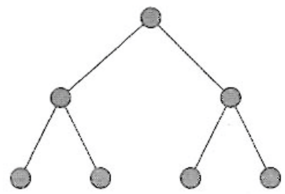
We suppose that we have in hands a route map (known environment)

Example: route finding – Costruzione dell’albero di ricerca



Tecniche di search

Visito l'albero e certo la soluzione migliore.



Ricerca partendo da un nodo e analizzando i nodi figli fino a quando non si arriva ai nodi foglia (frontiera).

Tree-search.

- Valutazione in termini di:
  - - Completezza: la ricerca trova una soluzione, se esiste?
  - - Tempo: quanto richiede il calcolo di una soluzione?
  - - Complessità: quanta memoria viene richiesta per la ricerca?
  - - Ottimalità: la ricerca trova la soluzione ottima quango esistono più soluzioni possibili?

Techiche prive di informazioni a-priori o con informazioni a-priori.

Search algorithm

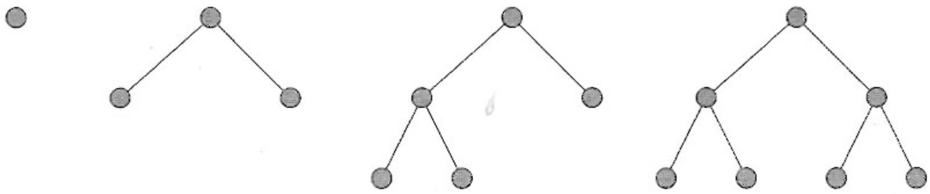
function TREE-SEARCH(*problem*) **returns** a solution, or failure
  initialize the frontier using the initial state of *problem*
  loop do
    if the frontier is empty **then return** failure
    choose a leaf node and remove it from the frontier
    if the node contains a goal state **then return** the corresponding solution
    expand the chosen node, adding the resulting nodes to the frontier

function GRAPH-SEARCH(*problem*) **returns** a solution, or failure
  initialize the frontier using the initial state of *problem*
  initialize the explored set to be empty
  loop do
    if the frontier is empty **then return** failure
    choose a leaf node and remove it from the frontier
    if the node contains a goal state **then return** the corresponding solution
    add the node to the explored set
    expand the chosen node, adding the resulting nodes to the frontier
      only if not in the frontier or explored set

Figure 3.7 An informal description of the general tree-search and graph-search algorithms. The parts of GRAPH-SEARCH marked in bold italic are the additions needed to handle repeated states.

Breadth-first search (larghezza prima)

b = branching factor



Definisco il costo di ciascun nodo figlio

Ricerca completa.
Ricerca ottima..

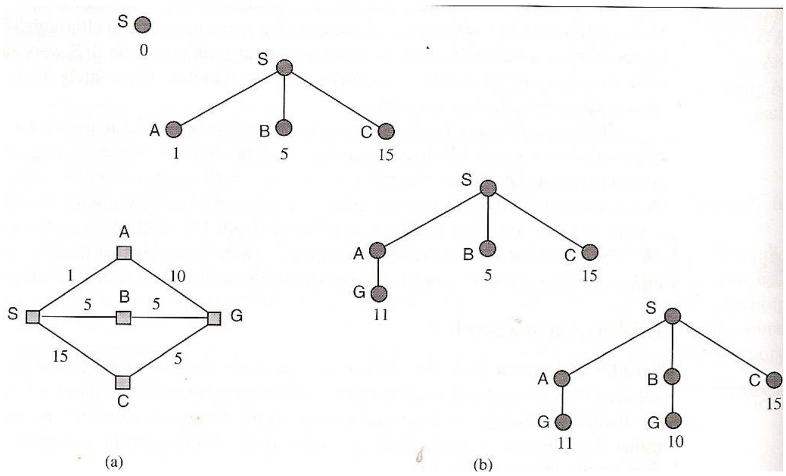
$N = 1 + b^1 + b^2 + b^3 + \dots + b^{d-1} = b^d - 1$
Crescita più che polinomiale (esponenziale) in tempo e spazio

Alcuni risultati (b=10)

Profondità	Nodi	Tempo	Memoria
0	1	1 ms	100 byte
2	111	0.1 s	11 Kbyte
4	11,111	11 s	1 Mbyte
6	10^6	18 min	111 Mbyte
8	10^8	31 h	11 Gbyte
10	10^{10}	128 gg	1 Tbyte
12	10^{12}	35 anni	111 Tbyte
14	10^{14}	3500 anni	11,111 Tbyte

Memory requirements and computational time, make optimal search unfeasible as can be expected by a NP problem.

Uniform cost search

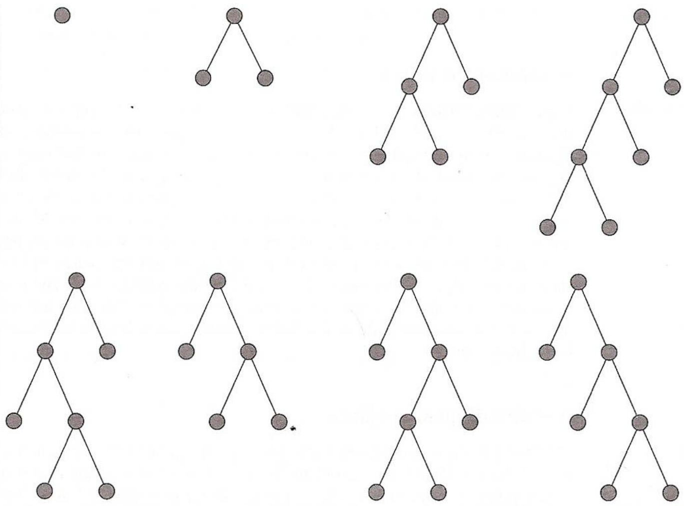


Ricerca completa.
Ricerca ottima..

g(SUCCESSOR(n)) ≥ g(n).

Depth-first search

Come diminuire l'occupazione di memoria?



La memoria richiesta è pari a: M = b d (breadth-first: $M = b^d$)
d = 12, b = 10; 12 Kbyte versus 111 Tbyte!!

Depth-limited search

Problemi della Depth-first search:

- Problemi per alberi profondi.
- Se prendo il ramo sbagliato, devo continuare ad esplorarlo, anche se, dopo pochi livelli mi potrei accorgere che è l'albero sbagliato.

Come evitare ciò?

Imporre una limitazione alla profondità dell'albero visitato. Questo si può fare in modo informato quando si hanno informazioni a-priori sui dati.

Questo limite richiede informazioni e non è detto che sia un buon limite.