كلية العلوم

القسم : **الرياضيات**

السنة : **الرابعة**

المادة : ذكاء صنعي

المحاضرة : **الثالثة/نظري/**

{{ مكتبة A to Z }}

Facebook Group : مكتبة A to Z

SOLVING PROBLEMS BY SEARCHING

Outline

Problem-solving agents
Problem types
Problem formulation
Example problems
Basic search algorithms

تقنيات البحث

The process of looking for a sequence of actions that reaches the goal is called **search**.

A search algorithm takes a problem as input and returns a **solution** in the form of an action sequence.

يقوم مبدأ عمل خوارزميات البحث في أنها تأخذ المشكلة كمدخلات ثم تقدم الحل في صورة سلسلة من العمليات أو الإجراءات التي تنتهي عند الوصول إلى الهدف والحصول على الحل النهائي.

الذكاء الصنعي يقوم بحل المشاكل عن طريق البحث عن حلول في فضاء الحالة **state space** وهذا البحث يتم بعدة طرائق وتقنيات(**خوارزميات بحث**).
فما هو فضاء الحالة؟ وما هي خوارزميات البحث وكيف تعمل؟

يصف هذا الفصل نوعًا واحدًا من الوكلاء القائمين على الأهداف يسمى وكيل حل المشكلات.

وكلاء حل المشكلات

من المفترض أن يقوم الوكلاء الأذكياء بتعظيم قياس أدائهم. وكما ذكرنا في الفصل الثاني، فإن تحقيق ذلك يكون أحيانًا إذا كان الفاعل قادرًا على تبني هدف والسعي إلى تحقيقه. دعونا نلقي نظرة أولاً على سبب وكيفية قيام الوكيل بذلك.

تخيل وكيلًا في مدينة Aradبرومانيا يستمتع بعطلة سياحية. يحتوي مقياس أداء الوكيل على العديد من العوامل:

فهو يريد تحسين سُمرة البشرة،
وتحسين اللغة الرومانية،
والاستمتاع بالمناظر الطبيعية،
والاستمتاع بالحياة الليلية (كما هي)،
وتجنب الكحوليات، وما إلى ذلك.

لنفترض أن الوكيل لديه تذكرة غير قابلة للاسترداد للسفر من بوخارست في اليوم التالي. في هذه الحالة، من المنطقي أن يتبنى الوكيل هدف الوصول إلى بوخارست. يمكن رفض مسارات العمل التي لا تصل إلى بوخارست في الوقت المحدد دون المزيد من الدراسة و تبسيط مشكلة قرار الوكيل إلى حد كبير.

تساعد **الأهداف في تنظيم السلوك** عن طريق الحد من الأهداف التي يحاول الوكيل تحقيقها

إن **صياغة الأهداف** ، بناءً على **الوضع الحالي ومقياس أداء الوكيل**، هي الخطوة الأولى في حل المشكلات.

صياغة المشكلة هي عملية تحديد الإجراءات والحالات التي يجب مراعاتها، في ضوء الهدف.

Problem Solving Agent

يصف هذا الفصل نوعًا واحدًا من الوكلاء القائمين على الأهداف يسمى وكيل حل المشكلات.

- **Problem-solving agent**: a type of **goal-based agent**

وكيل حل المشكلات: نوع من الوكلاء القائمين على الأهداف

- The process of looking for such a sequence of actions is called **search**

عملية البحث عن مثل هذه التسلسل من الإجراءات تسمى البحث

صياغة الهدف: بناءً على الموقف الحالي ومقياس أداء الوكيل

- **Goal formulation**: based on current situation and agent's performance measure

- **Problem formulation**: deciding what actions and states to consider, given a goal

صياغة المشكلة: تحديد الإجراءات والحالات التي يجب مراعاتها، مع الأخذ في الاعتبار الهدف

الوكلاء الذين يساعدون بحل المشاكل: هو نوع من أنواع الوكلاء القائمين على الهدف فهو يختص فقط بحل المشاكل.
التابع الخاص بهذا الوكيل: دخله مستقبلات (**percept**) من المحيط وخرجه الفعل المحدد(**action**)

```
function SIMPLE-PROBLEM-SOLVING-AGENT(percept) returns an action
  persistent: seq, an action sequence, initially empty
              state, some description of the current world state
              goal, a goal, initially null
              problem, a problem formulation

  state ← UPDATE-STATE(state, percept)
  if seq is empty then
      goal ← FORMULATE-GOAL(state)
      problem ← FORMULATE-PROBLEM(state, goal)
      seq ← SEARCH(problem)
      if seq = failure then return a null action
  action ← FIRST(seq)
  seq ← REST(seq)
  return action
```

المتحولات داخل التابع:
Seq: سلسلة الأفعال التي سيقوم بها الوسيط، بداية تكون فارغة
State:الحالة الحالية للوكيل(هذا ليس بمعنى ذاكرة)
Gool: الهدف الذي يحاول الوكيل الوصول إليه في البداية يكون null
Problem: الصيغة التي تتعامل معها المسألة

نقوم بتهيئة الحالة بناء على المستقبلات

اذا كانت سلسلة الأفعال المتاحة فارغة

يتم تحديد صيغة المسألة اعتمادا على الحالة الحالية والهدف
تحديد ما هي سلسلة الأفعال المتاحة لهذه المسألة
بعد كل هذا نقوم بتحديد ما هو الفعل المتاح للحالة الحالية

seq نقوم بوضع الأفعال المتبقية بالمتحول

وكيل بسيط لحل المشاكل: يقوم أولا **بصياغة هدف ومشكلة**، **وبحث** عن سلسلة من الإجراءات التي من شأنها حل المشكلة، ثم ينفذ الإجراءات واحدة تلو الأخرى. عندما يكتمل هذا، فإنه يصوغ هدفًا آخر ويبدأ من جديد.

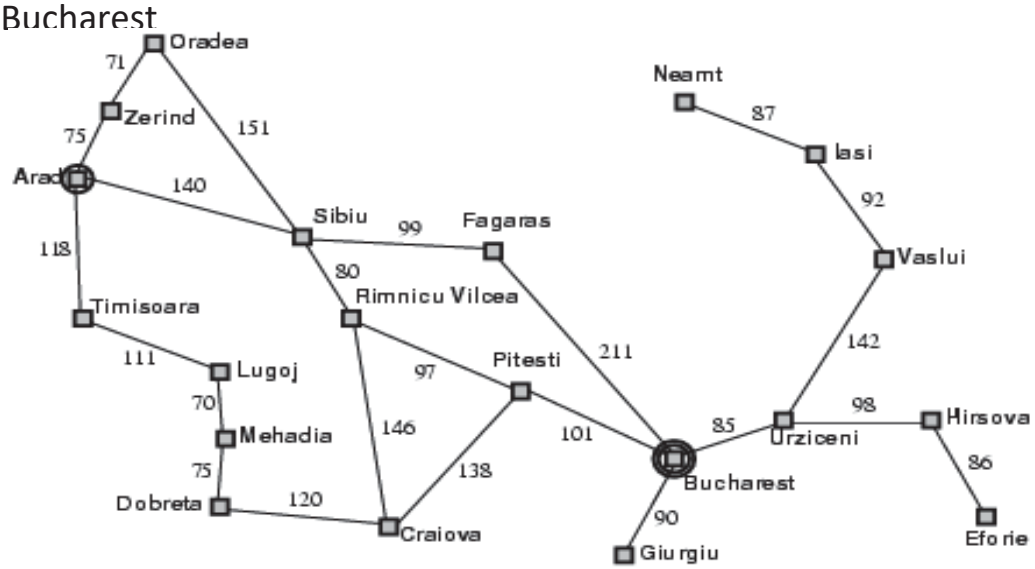
Example: Romania

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

Problem types

Deterministic, fully observable → single-state problem
Agent knows exactly which state it will be in; solution is a sequence

Non-observable → sensorless problem (conformant problem)
multiple-state problem ليس لديه أي معلومة عن مكانه وبالتالي سيكون هناك مجموعة احتمالات
Agent may have no idea where it is; solution is a sequence

Nondeterministic and/or **partially observable** → contingency problem مشكلة طارئة
percepts provide new information about current state often interleave
{ search, execution} تزود المدركات معلومات جديدة حول الحالة الحالية غالبًا ما تتداخل عملية {البحث، التنفيذ}

Unknown state space → exploration problem

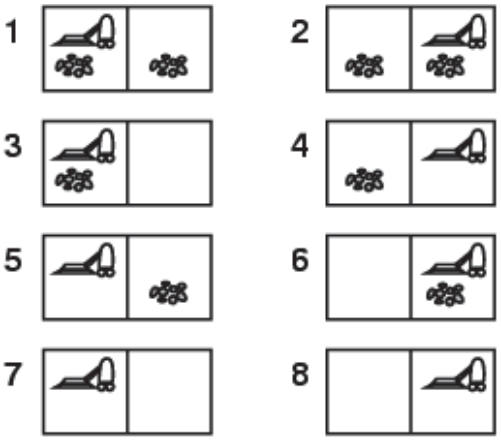
Problem types

Deterministic, fully observable → single-state problem

Agent knows exactly which state it will be in;

Vacuum world → everything observed

Romania → The full map is observed



Single-state: Start in #5.

Solution??

[Right, Suck]

Problem types

Non-observable → sensorless problem (multiple-state problem)

Agent may have no idea where it is; solution is a sequence

Vacuum world → No sensors

Romania → No map just know operators(cities you can move to)

multiple-state problem : Start in {1, 2, 3, 4, 5, 6, 7, 8}

ليس لديه أي معلومات لكن عنده كل الاحتمالات

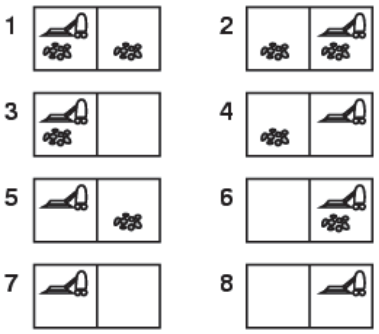
e.g., Right goes to {2, 4, 6, 8}.

تم توليد أربع احتمالات مختلفة

Solution??

[Right, Suck,Left, Suck]

كل action يتولد اكثر من احتمال أو اكثر من حالة



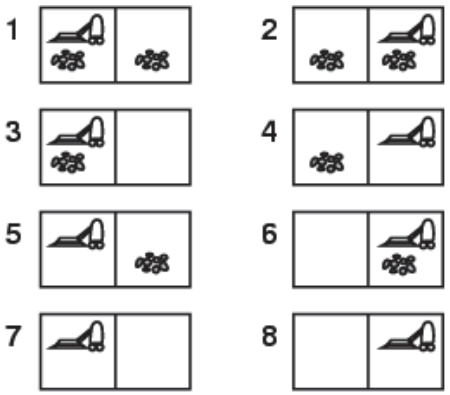
- Initial state:
  start with one of the set {1, 2, 3, 4, 5, 6, 7, 8}.
- Goal: {7,8}.
- Solution? **[*right, suck, left, suck*]**
  - ***Right*** → {2; 4; 6; 8}
  - ***Suck*** → {4; 8}
  - ***Left*** → {3; 7}
  - ***Suck*** → {7}

Problem types

Nondeterministic and/or partially observable → contingency problem
percepts provide new information about current state

Contingency: [L,clean]
Start in #5 or #7
Solution??
[Right, if dirt then Suck]
قبل كل action يحتاج معلومة ويسأل

حالة طارئة يمكن ان تحصل
الوكيل محتاج يسأل دائما قبل كل قرار
كل خطوة يحتاج معلومة ليأخذ القرار



Unknown state space → exploration problem
Vacuum world → know state of current location
Romania → know current location and neighbor cities

Single-state problem formulation

A problem can be defined formally by five Components

Initial state

Actions

Transition model: description of what each action does ***(successor)***

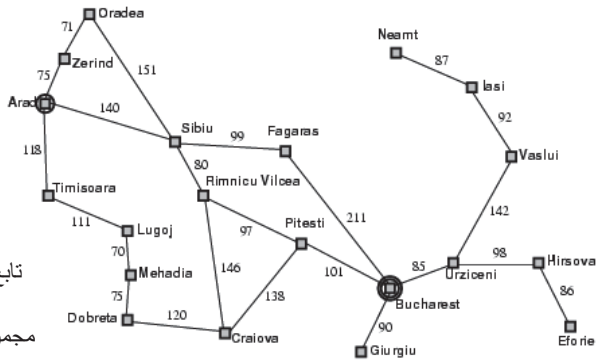
Goal test

Path cost

Problem Formulation (The Romania Example)

State: We regard a problem as **state space** here a state is a City

نعتبر المشكلة بمثابة فضاء الحالة



Initial State: the state to start from In(Arad)

تهيئة الحالة الابتدائية

Successor Function:. *S(x)* = set of action–state pairs

تابع تحديد الخطوة التالية

e.g., *S(Arad)* = {*<Arad → Zerind, Zerind>, ...* }

مجموعة من الأزواج مؤلفة من حالات وافعال

Goal Test: determine a given state is a goal state.

اختبار لهدف: تحديد حالة معينة هي حالة الهدف

explicit, e.g., *x* = "at Bucharest"

واضح: الوصول لمدينة بوخارست

implicit, e.g., NoDirt(x)

ضمني: المكنسة هل المربع متسخ أو لا

Path Cost: Additive.

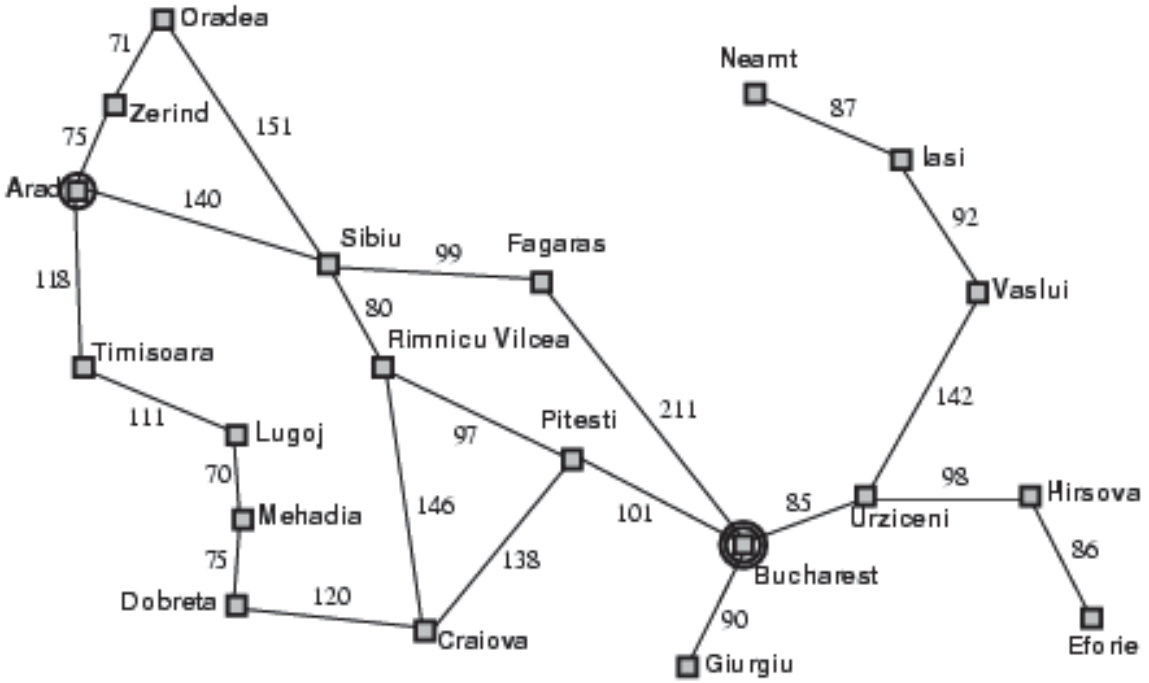
كلفة المسار (تراكمية)

على سبيل المثال، مجموع المسافات، عدد الإجراءات التي تم تنفيذها، إلخ.

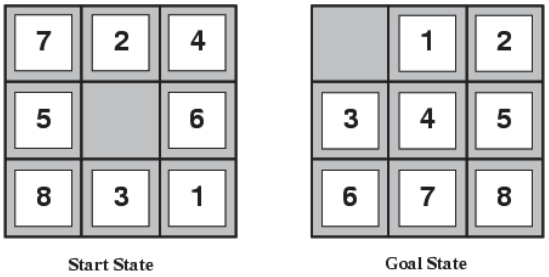
– e.g., sum of distances, number of actions executed, etc.
– *c(x,a,y)* is the step cost, assumed to be ≥ 0

Solution: a sequence of actions leading from the initial state to a goal state

Example: Romania (fig 3.2)



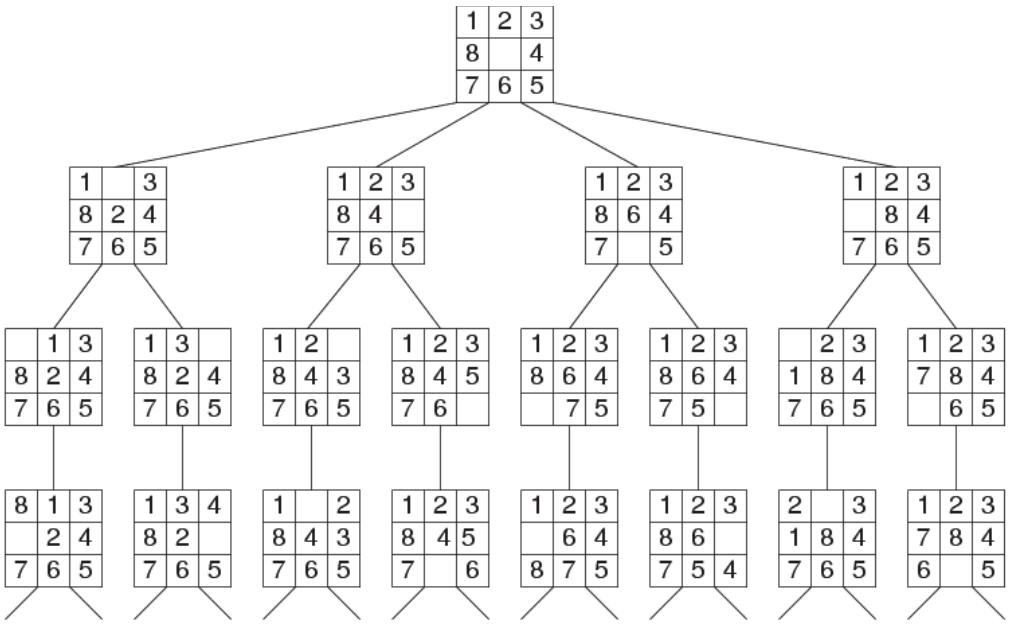
Example: The 8-puzzle (fig 3.4)



states? locations of tiles
actions? move blank left, right, up, down
goal test? = goal state (given)
path cost? 1 per move

[Note: optimal solution of *n*-Puzzle family is NP-hard]

Fragment of 8-Puzzle Problem Space



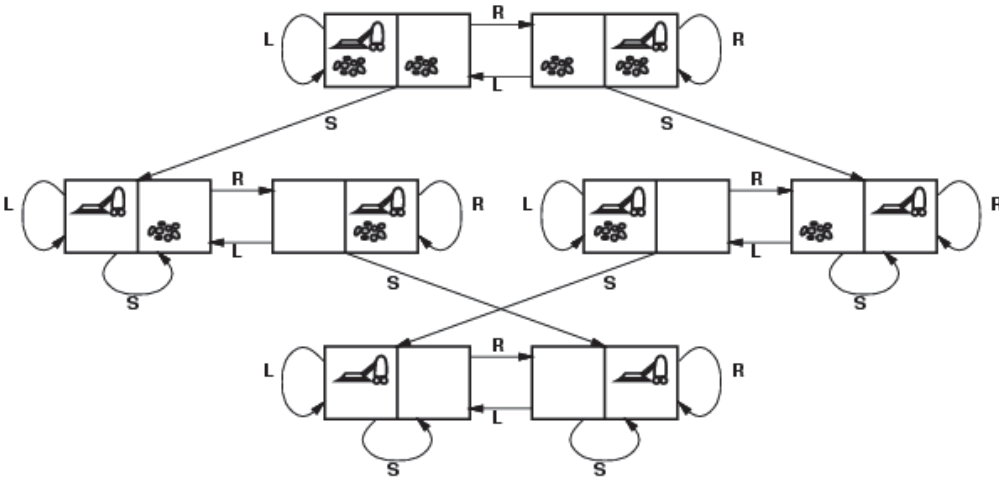
Vacuum world state space graph (fig3.3)

states? integer dirt and robot location

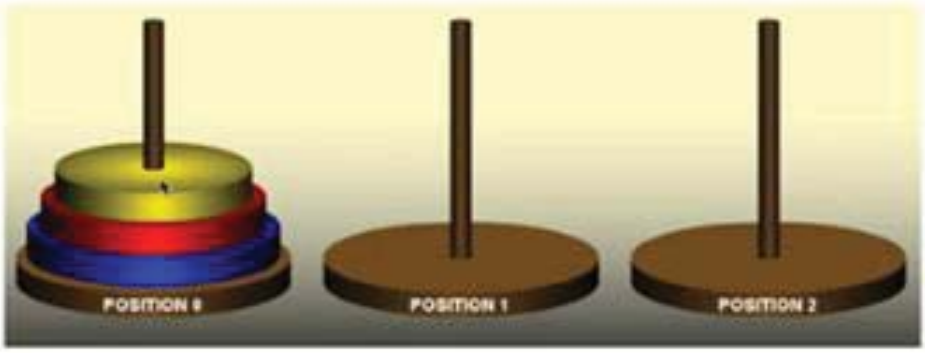
actions? *Left*, *Right*, *Suck*

goal test? no dirt at all locations

path cost? 1 per action



Tower of Hanoi



- **States:** disks location in the three possible positions
- **Initial State:** All disks in position 0
- **Successor function:** move disk between positions (with constraints)
- **Goal test:** All disks in position 2
- **Path cost:** 1 per move

Tree search algorithms

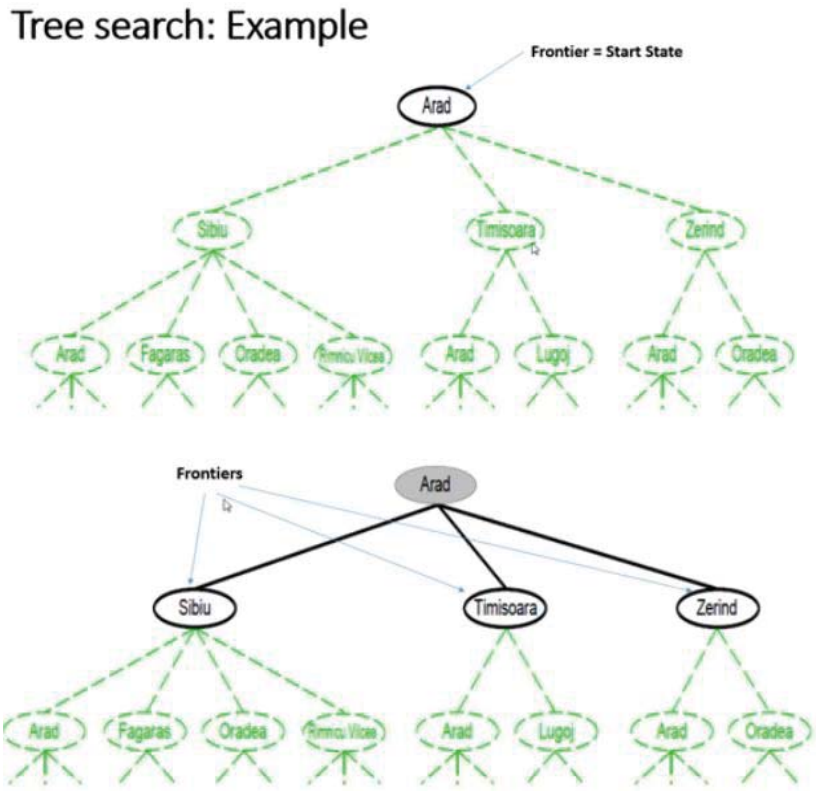
خوارزميات شجرة البحث:

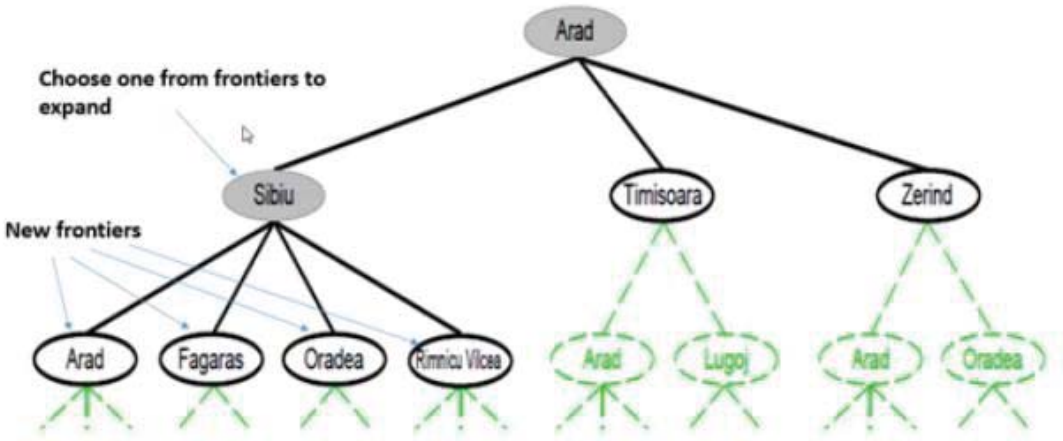
الهدف من هذه الخوارزميات هو العثور على حل لمسألة ما، ونفحص كامل فضاء الحالات المتاح.

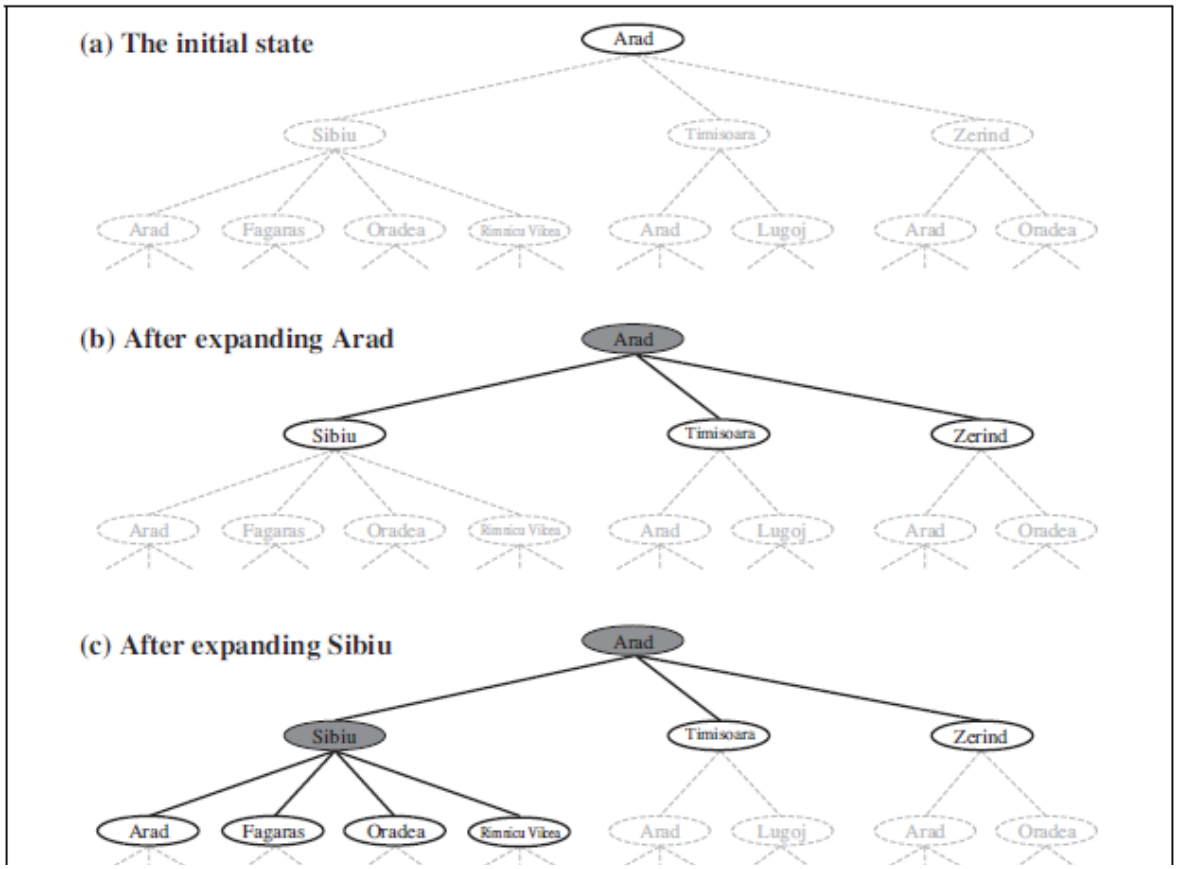
الحل هو تسلسل الأفعال، تشكل **تسلسلات الأفعال** المحتملة التي تبدأ بالحالة الأولية **شجرة بحث search tree**
حيث تمثل الحالة الأولية (**initial state**) جذرالشجرة **root**؛ الفروع (**branches**)هي إجراءات او **actions**
يوضح الشكل ٣,٦ الخطوات الأولى في بناء شجرة البحث للعثور على طريق من أراد إلى بوخارست.

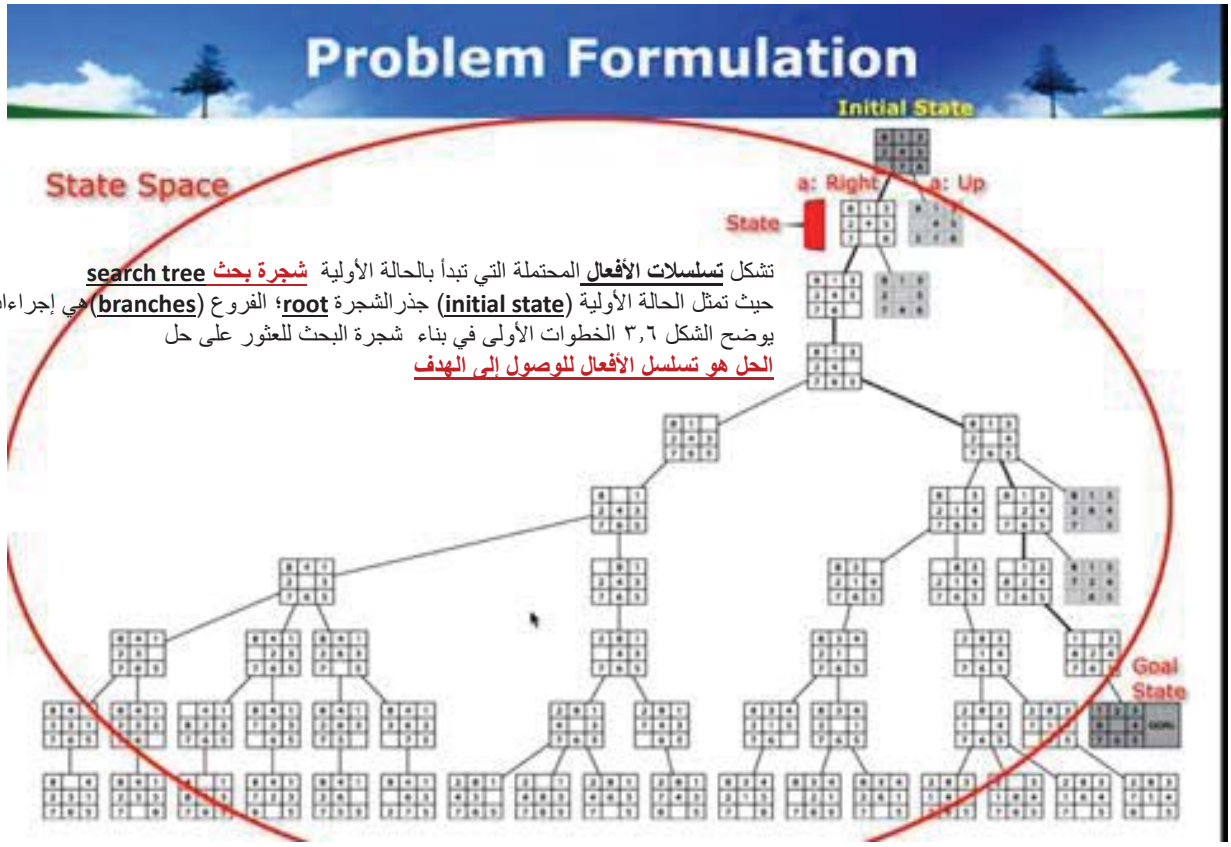
function TREE-SEARCH(*problem*) **returns** a solution, or failure
  initialize the frontier using the initial state of *problem* بداية يتم تهيئة شجرة البحث اعتمادا على الحالات الابتدائية للمسألة ثم الدخول بحلقة
  loop do
    if the frontier is empty **then return** failure اذا تم فحص جميع الحالات ولم يتم العثور على هذف سيعيد التابع فشل
    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

نتابع خلاف ذلك: نقوم باختيار غقدة ورقية من الشجرة حسب الاستراتيجيات المعتمدة ونقوم بفحصها
اذا كانت هي الحالة هدف نعيد الحل المقابل لهذه الحالة وإلا تقوم بتوسعة هذه العقدة ونضيف العقد الجديدة الموسعة الى شجرة البحث

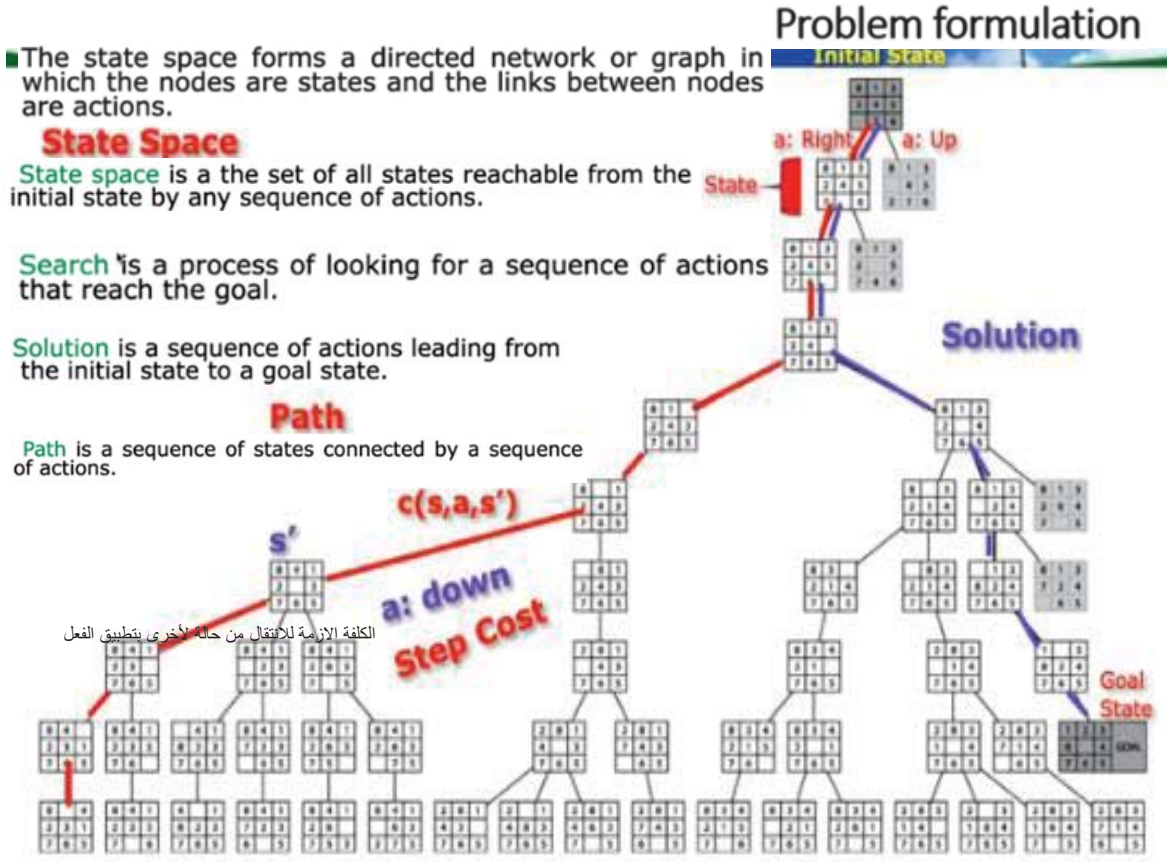



Arad
Choose one from frontiers to expand
Sibiu
Timisoara
Zerind
New frontiers
Arad
Fagaras
Oradea
Rimnicu Vilcea
Arad
Lugoj
Arad
Oradea


(a) The initial state
Arad
Sibiu
Timisoara
Zerind
Arad
Fagaras
Oradea
Rimnicu Vilcea
Arad
Lugoj
Arad
Oradea
(b) After expanding Arad
Arad
Sibiu
Timisoara
Zerind
Arad
Fagaras
Oradea
Rimnicu Vilcea
Arad
Lugoj
Arad
Oradea
(c) After expanding Sibiu
Arad
Sibiu
Timisoara
Zerind
Arad
Fagaras
Oradea
Rimnicu Vilcea
Arad
Lugoj
Arad
Oradea



تشكل **تسلسلات الأفعال** المحتملة التي تبدأ بالحالة الأولية **شجرة بحث search tree**
حيث تمثل الحالة الأولية (**initial state**) جذرالشجرة **root**؛ الفروع (**branches**)هي إجراءات او **actions**
يوضح الشكل ٣,٦ الخطوات الأولى في بناء شجرة البحث للعثور على حل
الحل هو تسلسل الأفعال للوصول إلى الهدف



Implementation: states vs. nodes

A **state** is a (representation of) a physical conguration

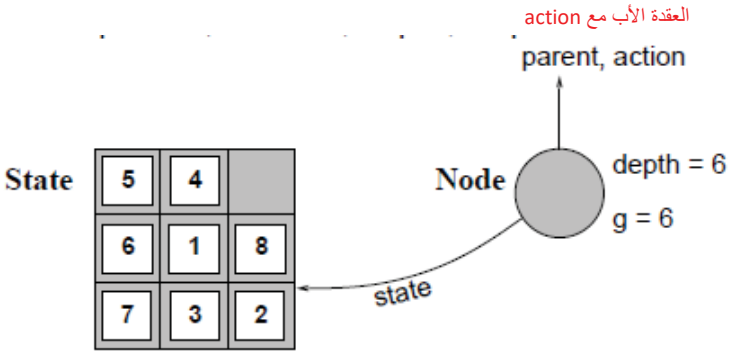
A **node** is a data structure constituting part of a search tree
Includes **state, parent**, **children**, **depth**, path **cost g(x)**

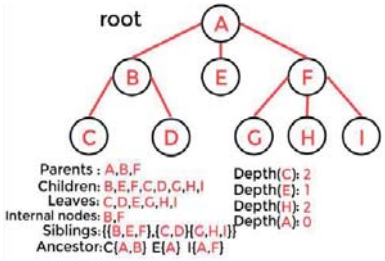
States do not have parents, children, depth, or path cost!

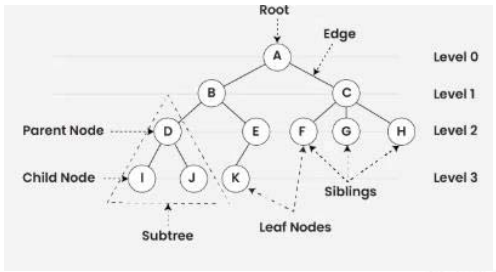
ما الفرق بين **العقدة والحالة:**

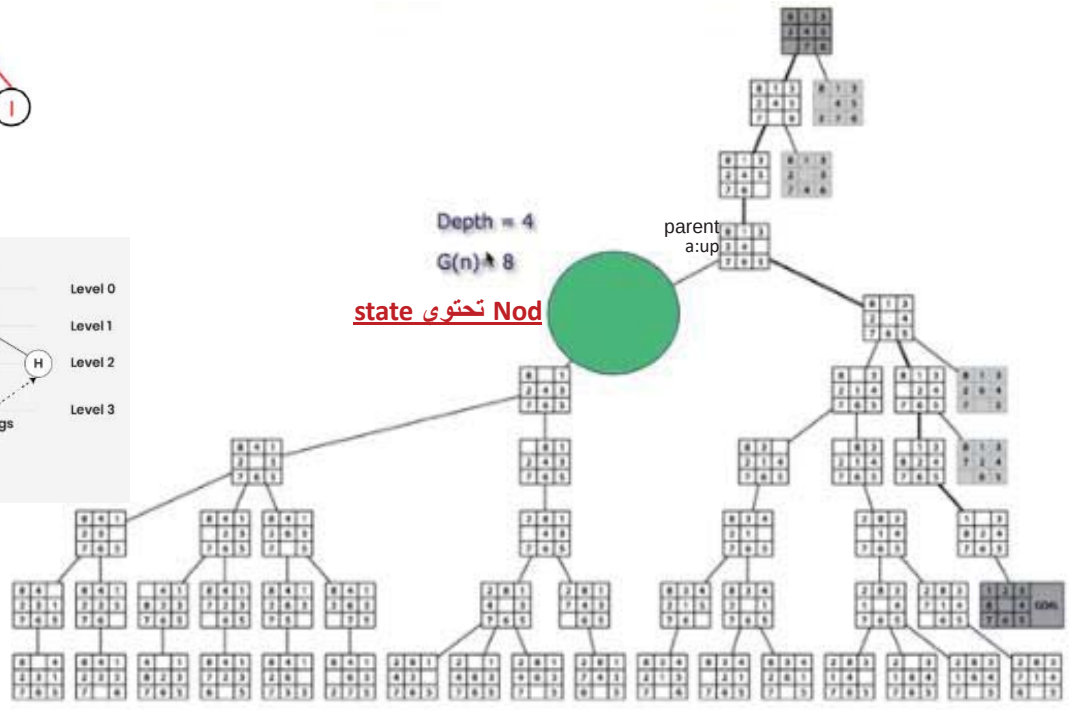
الحالة هي توضع فيزيائي بالمسألة مثلا مدينة، مربع متسخ...

العقدة: هي بنية معطيات ضمن الشجرة تتكون من عدة حقول أحدها هي الحالة وأيضا العقدة الأب، العمق، التكلفة، الفعل الذي أوصلنا للعقدة الحالية.



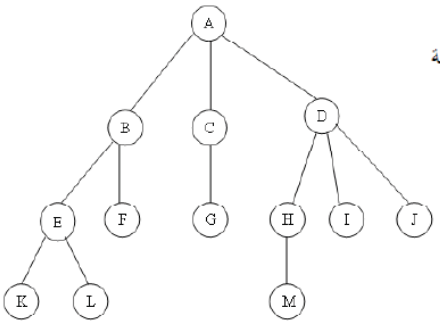






تعريف 1 :

- درجة عقدة (degree of node) هي عدد الأشجار الفرعية لهذه العقدة . فمثلا درجة العقدة A تساوي 3 و درجة العقدة C تساوي 1 و درجة العقدة F صفر .
- درجة الشجرة degree(T) هو العدد الأعظمي للأشجار الفرعية لعقدة أي أن : $degree(T) = \max\{degree(x) \mid x \text{ is a node of } T\}$. درجة الشجرة T تساوي 3



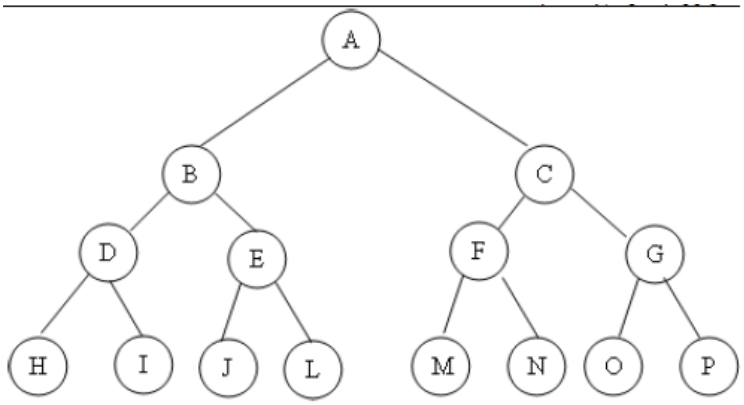
الشكل(7- 1) : شجرة

- تسمى كل عقدة لها شجرة فرعية واحدة على الأقل عقدة داخلية (internal node) مثل العقد : B, C, D, E, H
- تسمى كل عقدة ليس لها أي شجرة فرعية عقدة ورقية (leaf node) أو عقدة خارجية (external node) مثل العقد K, L, F, G, M, I, J

- يمكن أن توجد بين العقد العلاقات التالية : أب (parent) – ابن(child) – أخ (sibling) – سلف (ancestor) – حفيد (grandchild). نقول عن عقدتين إنهما أختان (siblings) إذا كان لهما الأب نفسه . مثلا : العقدة B أب للعقد E, F ، وبالعكس العقدتان E, F أولاد للعقدة B . كما أن العقد H, I, J أخوات .
- نقول عن العقدة x إنها سلف لعقدة y ، إذا وفقط إذا كانت x أباً لـــ y ، أو كانت x سلفاً لأب y . مثلا : كل من العقد H, D, A يكون سلفاً للعقدة M
- نقول عن العقدة x إنها حفيد لعقدة y ، إذا وفقط إذا كانت x ابناً لـــ y ، أو كانت x حفيداً لابن y. العقد K, E, F أحفاد للعقدة B .

نسمي مساراً (path) ضمن شجرة كل متتالية من العقد ، و نسمي فرعاً (branch) في الشجرة كل مسار يصل بين عقدة الجذر و إحدى العقد الورقية . مثلا :A, D, H, M فرع في الشجرة T

الشجرة الثنائية الممتلئة Full Binary Tree: هي شجرة تحوي عقدة واحدة في المستوي 0 و عقدتين في المستوي 1 ، و أربع عقد في المستوي 2 ،.....، 2^k في المستوي k ،



الشكل(7-5) : شجرة ثنائية ممتلئة

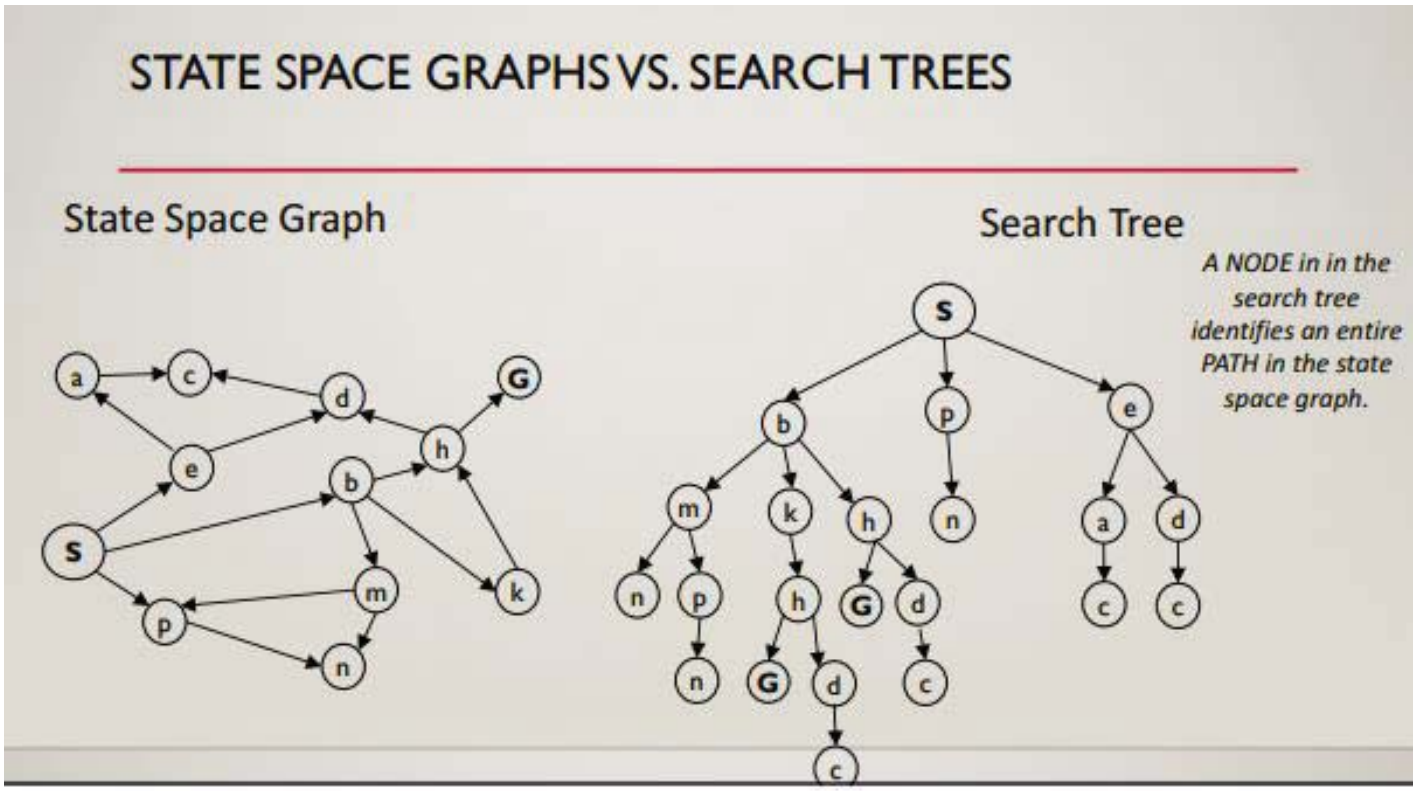
6-2 تعريف المكدس

هو قائمة مرتبة (خطية) من العناصر حيث يتم فيها الإدخال و الحذف من نهاية واحدة تدعى قمة المكدس top . فمثلا من أجل المكدس $S = (a_1, a_2, \ldots, a_n)$ ، يكون a_1 العنصر الأسفل في المكدس و a_n العنصر الأعلى في المكدس ، ويكون العنصر a_i فوق العنصر a_{i-1} . $(1 < i < n)$

القيود على المكدس تستلزم إذا أدخلنا العناصر A, B, C, D, E إلى مكدس ، على الترتيب، عندئذ يكون E العنصر الأول الذي نستطيع حذفه من المكدس . الشكل (6-1) يوضح متتالية المؤثرات الإضافة و الحذف . و بما أن كون آخر عنصر أضيف إلى المكدس يكون أول عنصر حذف منه . لذا ، فإن آلية تخزين العناصر في المكدس تكون بأسلوب Last-In-First –Out (LIFO) .

6-6 مفهوم الرتل Queue

يعتبر الرتل بنية من بنى المعطيات الأساسية شديد الشبه ببنية المكدس حيث لا يختلف عنه إلا في أن العنصر الأول في الإدخال هو نفسه العنصر الأول في الحذف . و ذلك ما يعرف First-In-First-Out (FIFO) .



Search strategies

استراتيجيات البحث

ما هي العقدة التي سنقوم بتوسعتها؟

الاستراتيجية المستخدمة هي التي تحدد العقدة

يوجد عدة معايير تميز الاستراتيجيات عن بعضها وهي

A strategy is defined by picking the *order of node expansion*
Strategies are evaluated along the following dimensions:

completeness—does it always find a solution if one exists?

كاملة أو شاملة:

دائما تقوم بالعثور على الحل في حال وجوده ضمن الشجرة

time complexity—number of nodes generated/expanded

تعقيد الزمن:

تحدد عدد العقد المولدة أو الموسعة

space complexity—maximum number of nodes in memory

تعقيد الفضاء أو الذاكرة(المكان)

العدد الأعظمي من العقد الذي ستقوم بتخزينه في الذاكرة

optimality—does it always find a least-cost solution?

الأمثلية:

وهي التي تعبر عن الحل الأفضل بكلفة اصغرية

Time and space complexity are measured in terms of
b—maximum branching factor of the search tree
d—depth of the least-cost solution
m—maximum depth of the state space (may be *1*)

لتحديد تعقيد الوقت و الذاكرة نستخدم المقاييس التالية:

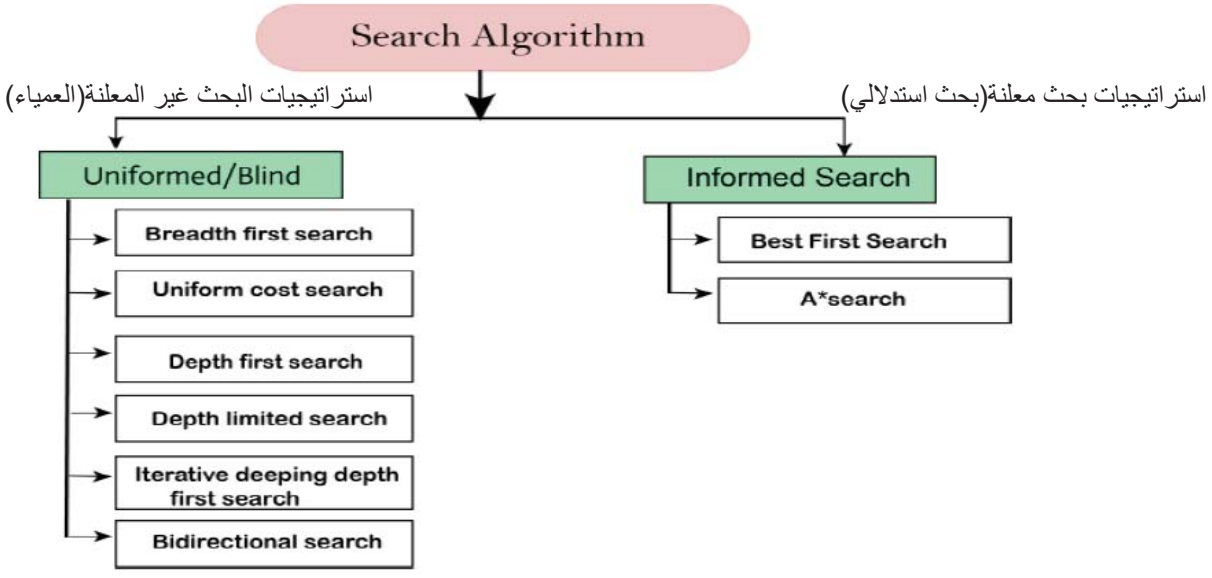
b: العدد الأعظمي لفروع الشجرة ضمن المسألة

d: عمق الحل ذو الكلفة الأصغري(الأمثل)

m: العمق الأعظمي للفضاء.يمكن في بعض الحالات أن يكون ضخم جدا

Types of search algorithms

Based on the search problems we can classify the search algorithms into **uninformed (Blind search) search** and **informed search (Heuristic search)** algorithms.



Uninformed search strategies

استراتيجيات البحث غير المعلنة

Uninformed strategies use only the information available in the problem definition

تستخدم استراتيجيات البحث غير الموجهة(العمياء) المعلومات المتاحة فقط في تعريف المشكلة

States, actions, goal test, path cost

- Breadth-first search (BFS)
- Uniform-cost search (UCS)
- Depth-first search (DFS)
- Depth-limited search (DLS)
- Iterative deepening search (IDS)

هذه الاستراتيجيات لا تحتوي على معلومات إضافية حول الحالات وكل ما يمكنها فعله هو إنشاء خلفاء وتمييز حالة الهدف عن حالة غير الهدف.
وتتميز جميع استراتيجيات البحث بالترتيب الذي يتم به توسيع العقد.