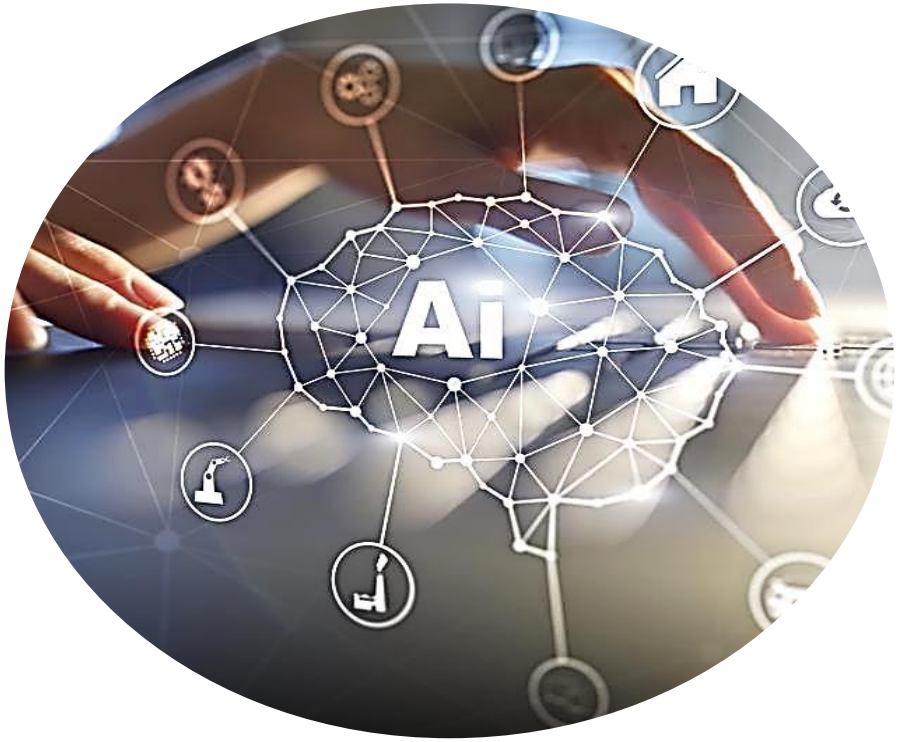
مقدمة في الذكاء الصنعي

الهندسة المعلوماتية



مدرس المقرر

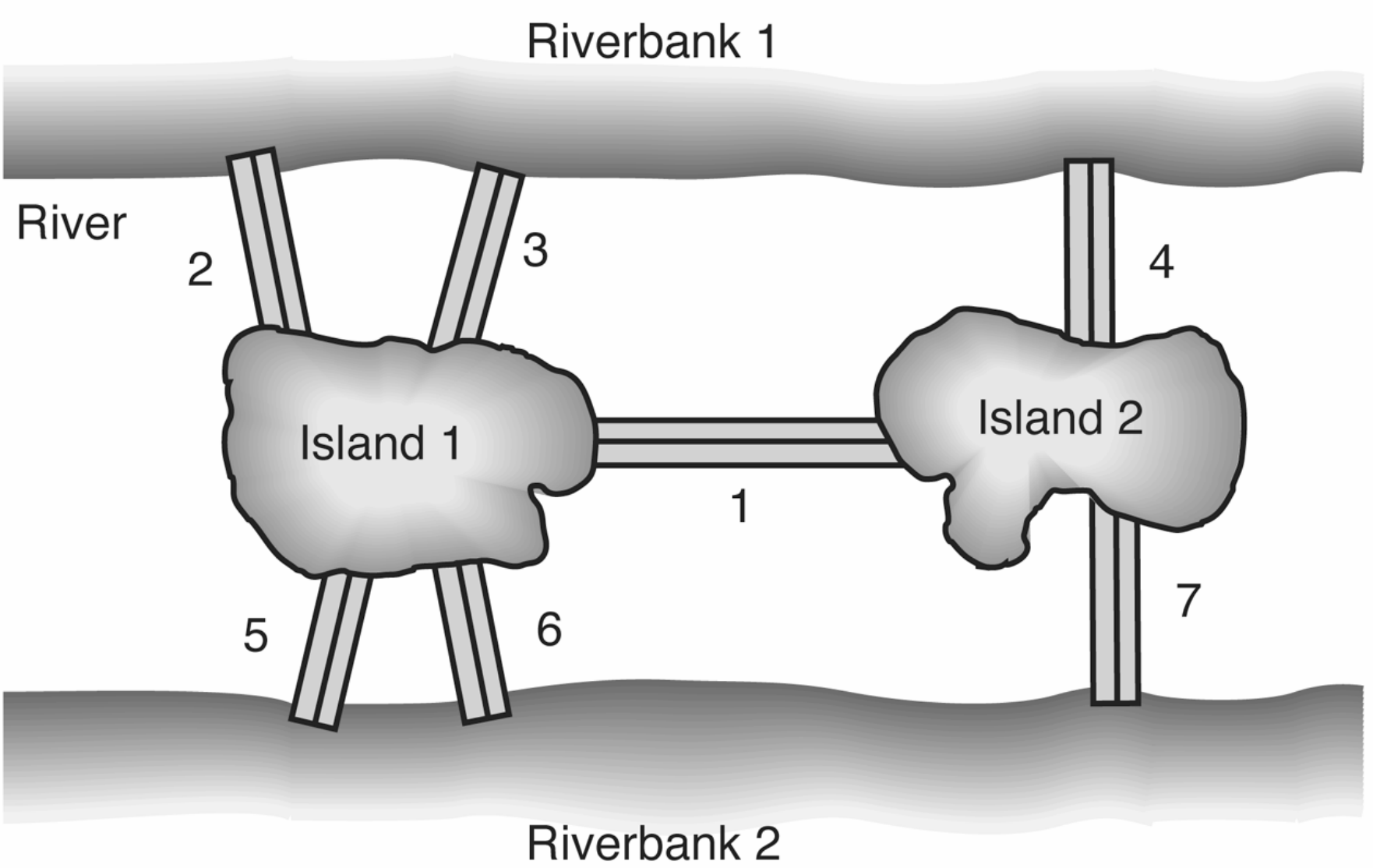
د.بلال شيحا

- **بعد تعريف فضاء المسألة (تمثيل الحالات , الأفعال, الحالة الابتدائية, الحالة النهائية) يبدأ البحث!**
- **انطلاقا من الحالة الابتدائية نعاود تطبيق الأفعال الممكنة حتى الوصول الى الحالة النهائية, الا ان فضاء البحث عادة ضخم جدا و بالتالي نحتاج الى منهجيات تقود البحث .**

Bridges of Königsberg Problem

- The city of Konigsberg occupied both banks and two islands of a river
- احتلت مدينة كونيغسبيرغ كلا الضفتين وجزيرتين من النهر
- The islands and the riverbanks were connected by seven bridges
- تم ربط الجزر وضفتي النهر بسبعة جسور
- The bridges of Konigsberg problem asks if there is a walk around the city that crosses each bridge exactly once
- مسألة جسور كونيغسبيرج هي السؤال عن إمكانية القيام بنزهة حول المدينة تعبر كل جسر من الجسور مرة واحدة فقط.

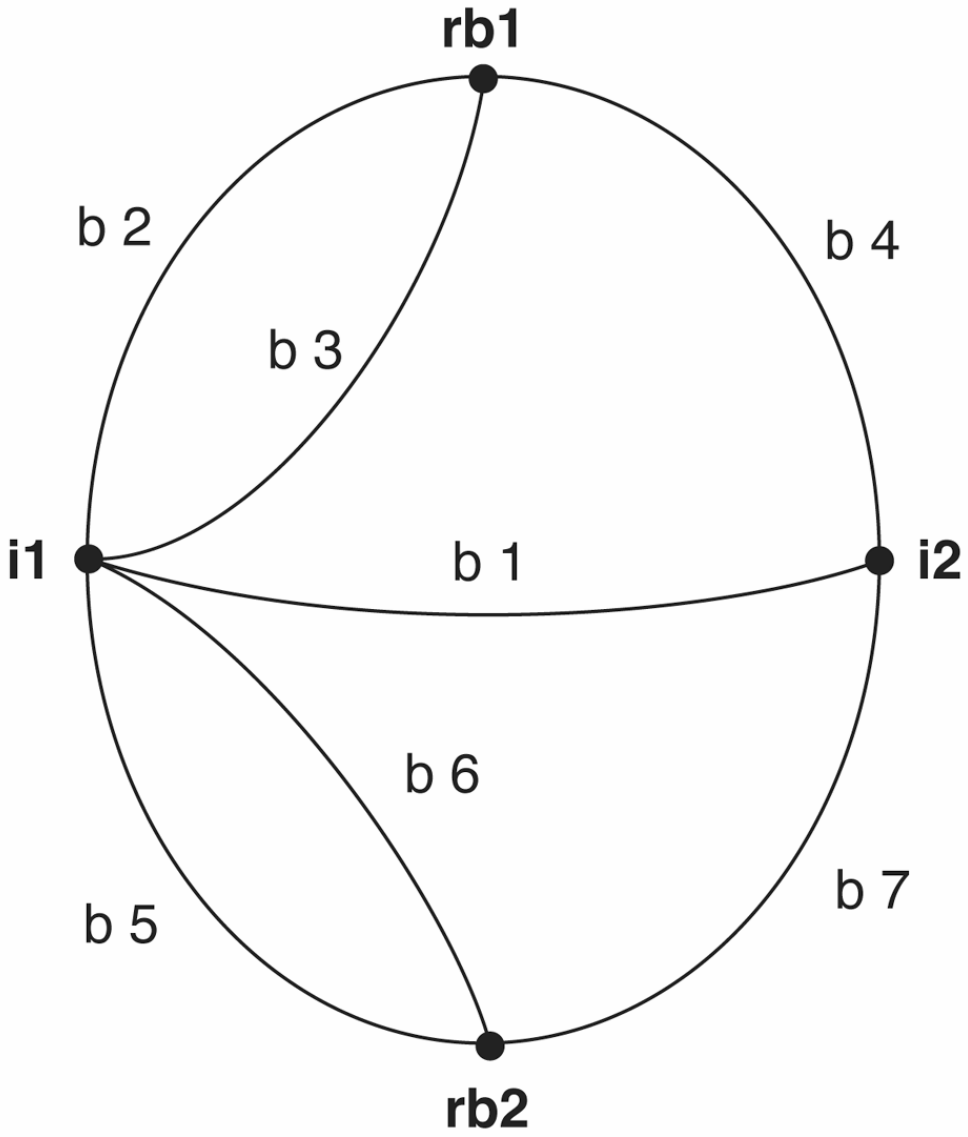
Bridges of Königsberg Problem



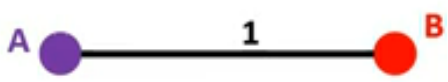
Bridges of Königsberg Problem

- Leonhard Euler invented graph theory to solve the bridges of Konigsberg problem
- اخترع ليونارد أويلر نظرية البيان لحل مشكلة جسور كونيغسبيرغ
- Graph Theory is our best tool for reasoning about the structure of objects and relations In the problem,
- نظرية البيان هي أفضل أداة لدينا للتفكير حول بنية الأشياء والعلاقات في المشكلة ،
- rb1 and rb2 are riverbanks 1 and 2
- i1 and i2 are two islands
- b1,b2,b3,b4,b5,b6,b7 are bridges

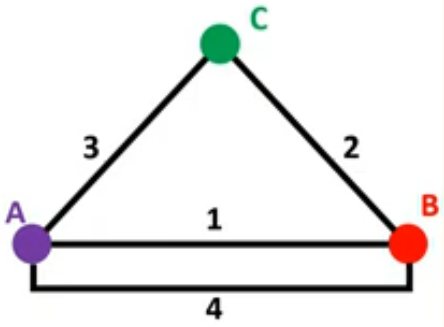
Graph of the Königsberg bridge system



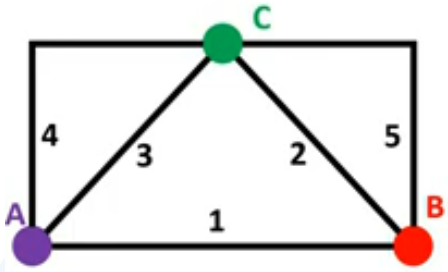
Graph of the Königsberg bridge system



Node	Degree
A	1
B	1

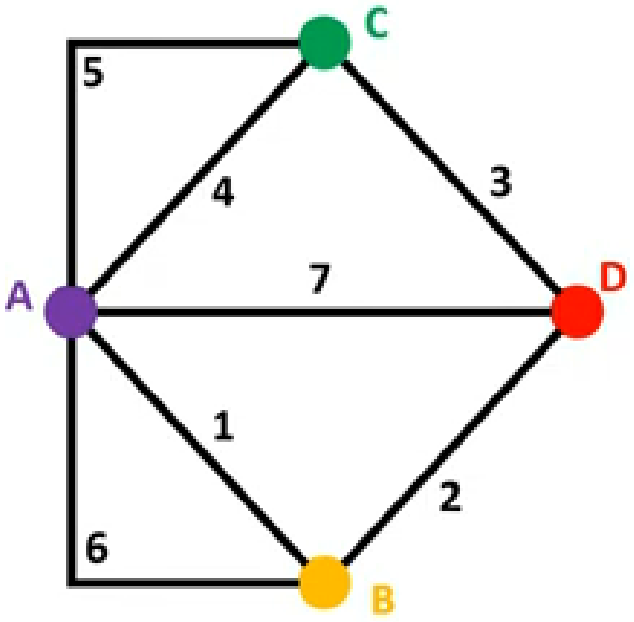


Node	Degree
A	3
B	3
C	2



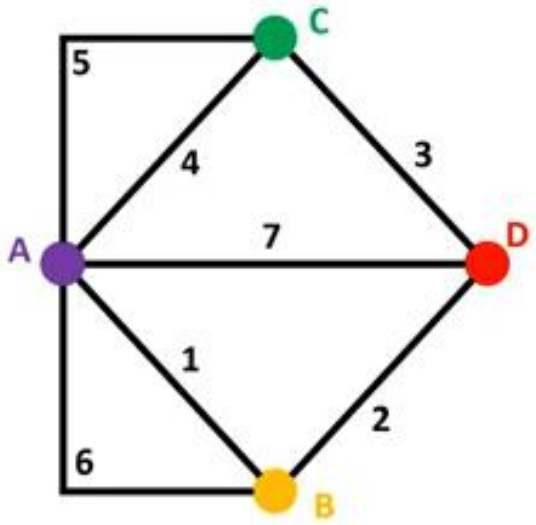
Node	Degree
A	3
B	3
C	4

Graph of the Königsberg bridge system



Node	Degree
A	5
B	3
C	3
D	3

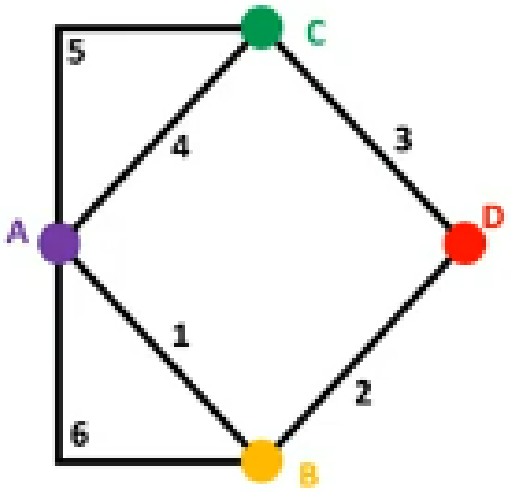
Graph of the Königsberg bridge system



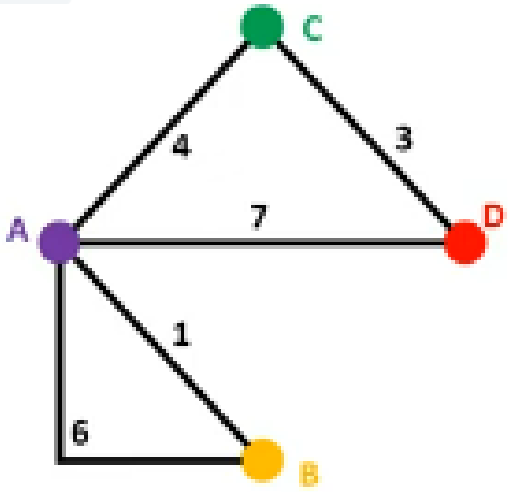
Node	Degree
A	5
B	3
C	3
D	3

IMPOSSIBLE ...!!!

Graph of the Königsberg bridge system



Node	Degree
A	4
B	3
C	3
D	2

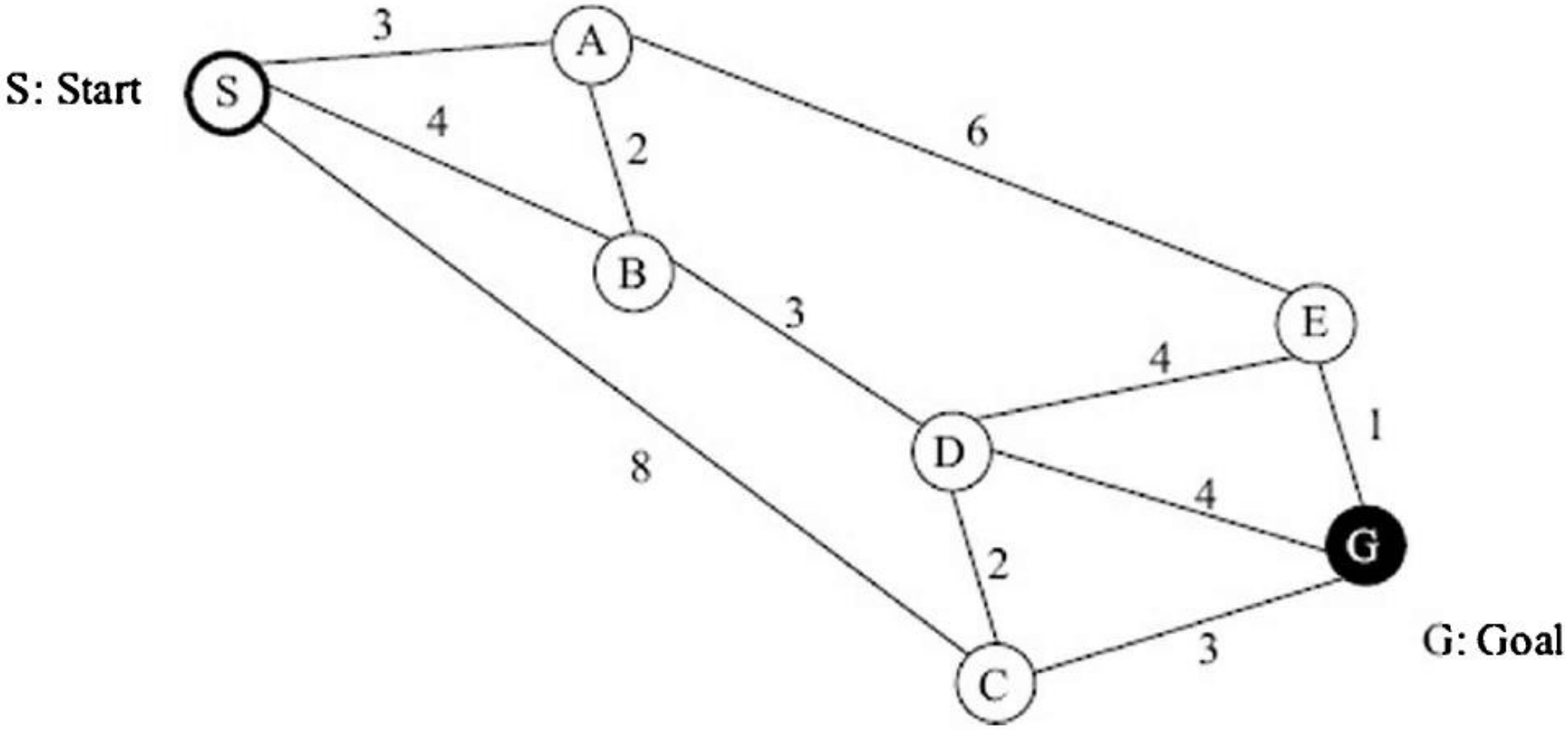


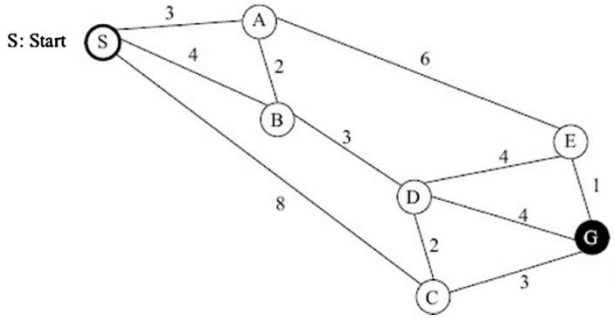
Graph of the Königsberg bridge system

- كان التفكير بحل مسألة جسور كوينغسبرغ السبعة بداية علم نظرية البيانات.
- ليس لها حل بصيغتها المطروحة.
- و يمكن ان يكون هناك حل في حال كان عدد الحواف الملتقية عند العقد (درجة العقدة) عدد زوجي (الانطلاق من عقدة و العودة إليها).
- و أيضا إذا كان هناك عقدتين بدرجة فردية و الباقي زوجي (الانطلاق من أحدى العقدتين و الوصول إلى العقدة الثانية).

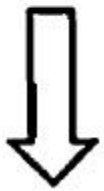
البحث الشجري
التجوال في شجرة البحث

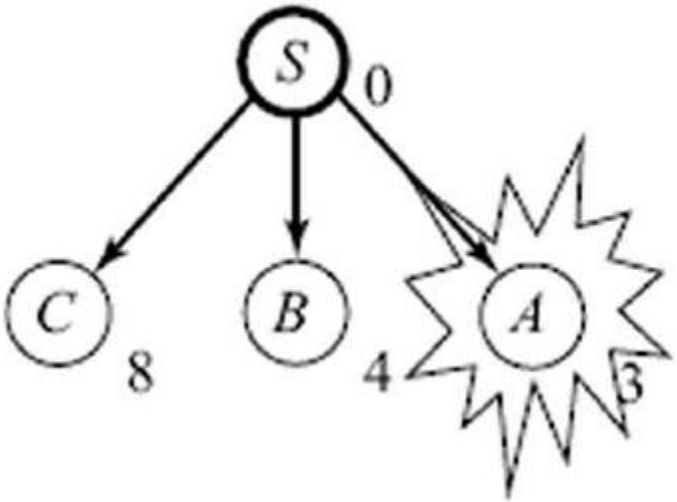
- سنعتمد فيما يلي البحث من خلال التجوال في الشجرة وفق القواعد المعرفية و سنبدأ بالتعريف بقواعد البحث الشجري:
- البحث الشجري: بحث منقطع (غير متصل) offline تجري خلاله محاكاة استكشاف حالات فضاء الحالة من خلال توليد الحالات التالية لحالة يجري اختبارها (ما يطلق عليه توسيع أو نشر الحالات).
- تعرف استراتيجية البحث من خلال اختيار تسلسل توسيع أو نشر العقد.

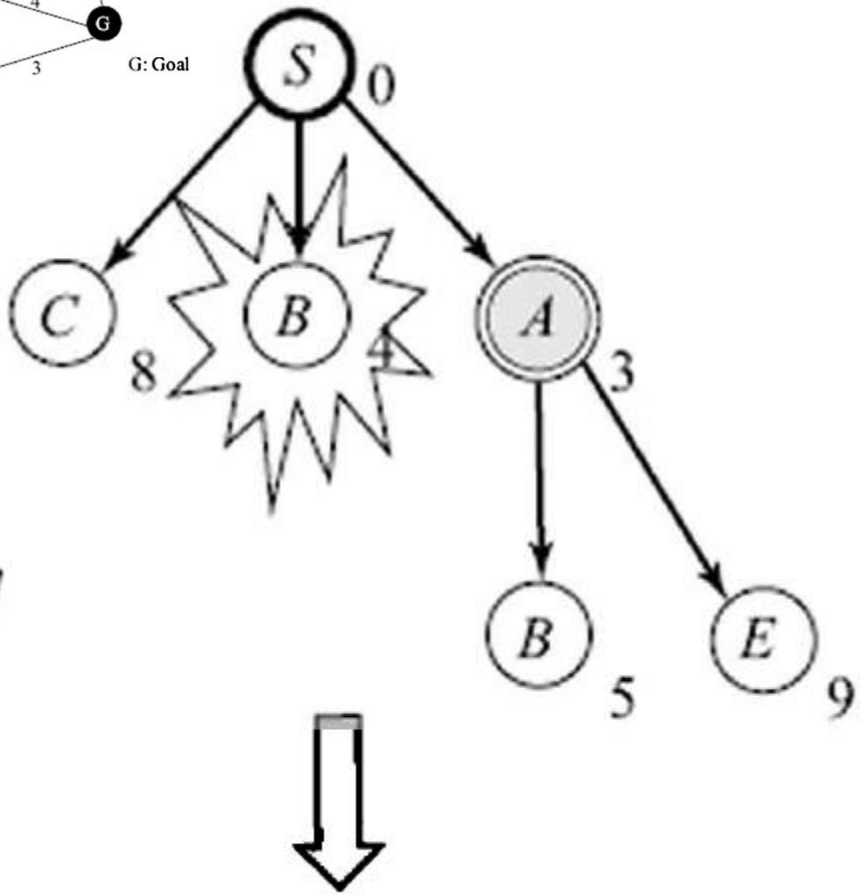

S: Start
S
A
B
C
D
E
G
G: Goal
3
4
2
6
3
8
4
1
4
2
3

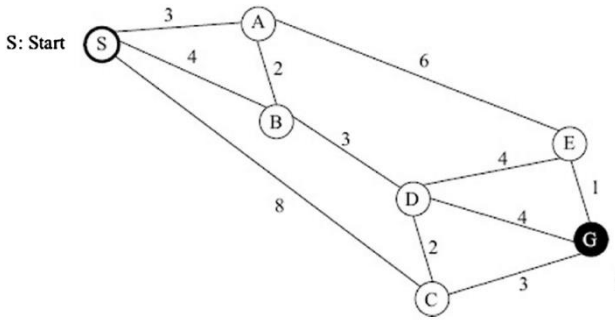

S: Start
S
A
B
C
D
E
G
G: Goal
3
4
2
6
3
4
1
8
4
2
3

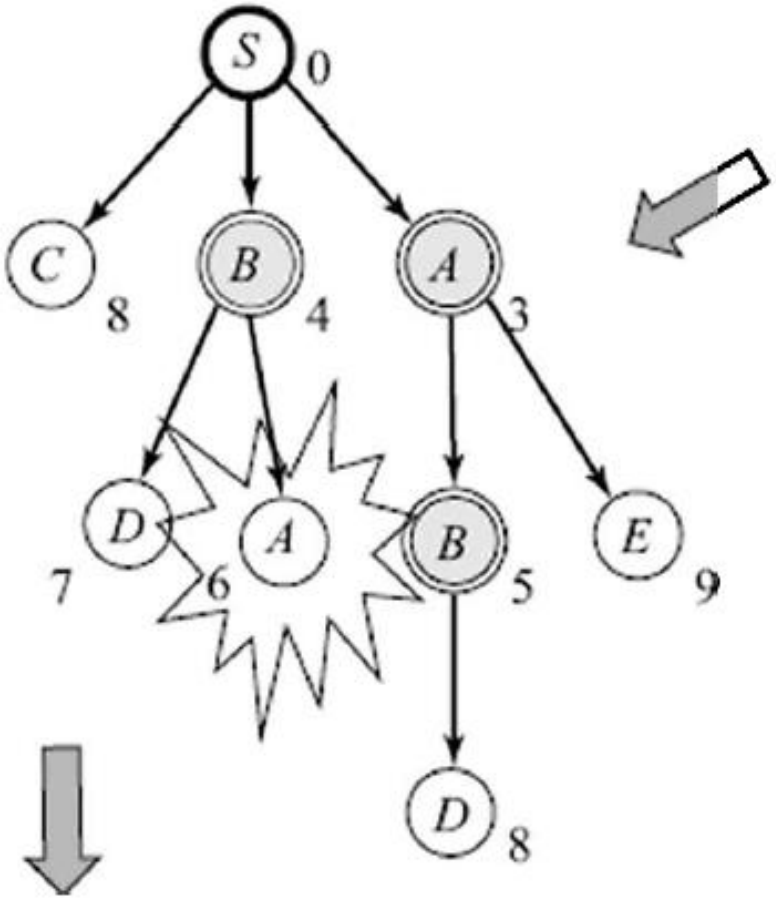

S
0

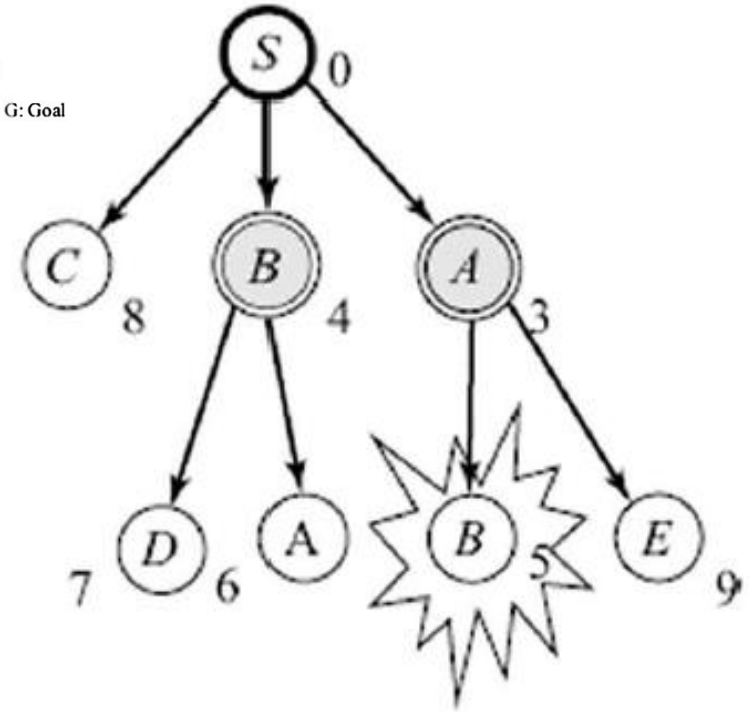


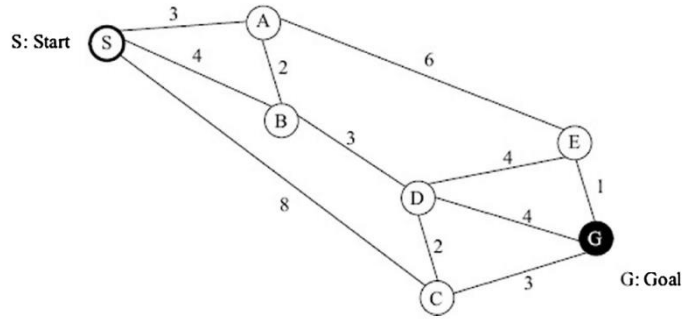

S
0
C
8
B
4
A
3

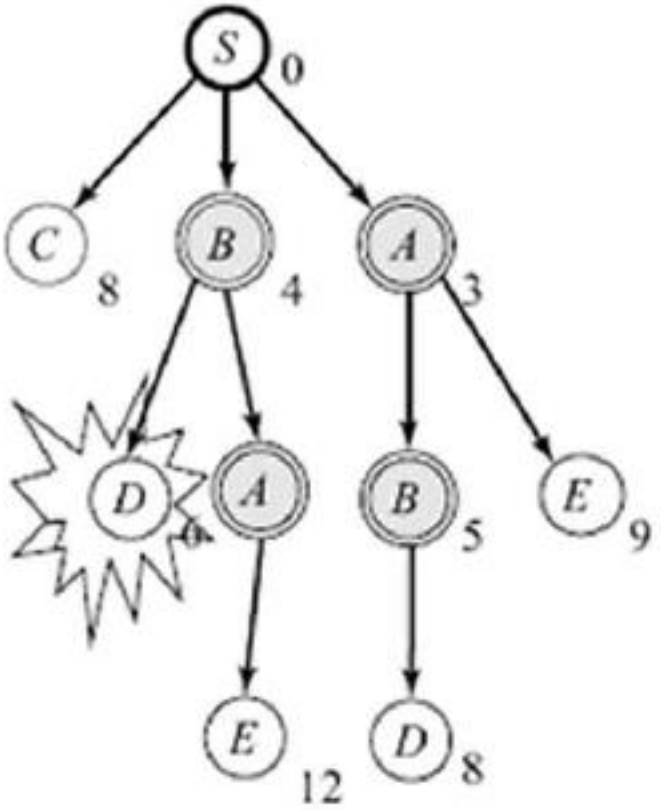

S
0
C
8
B
4
A
3
B
5
E
9

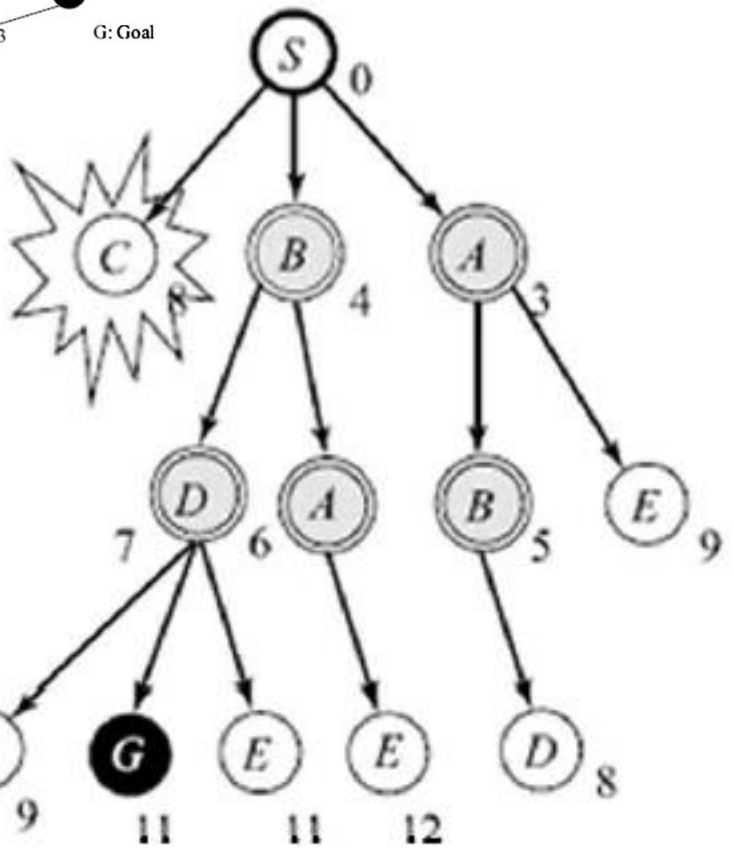

S: Start
S
A
B
C
D
E
G
G: Goal
3
4
2
6
3
4
1
8
4
2
3

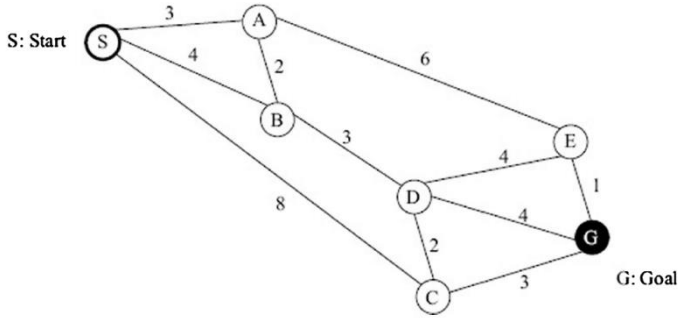

S 0
C 8
B 4
A 3
D 7
A 6
B 5
E 9
D 8

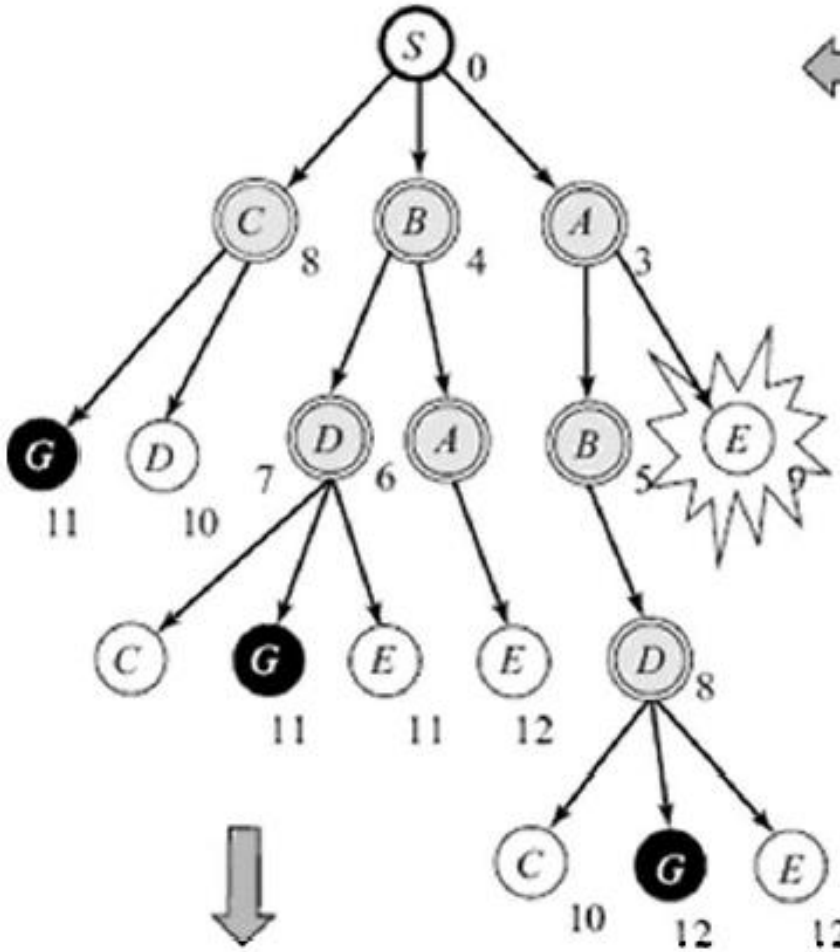

S 0
C 8
B 4
A 3
D 7
A 6
B 5
E 9

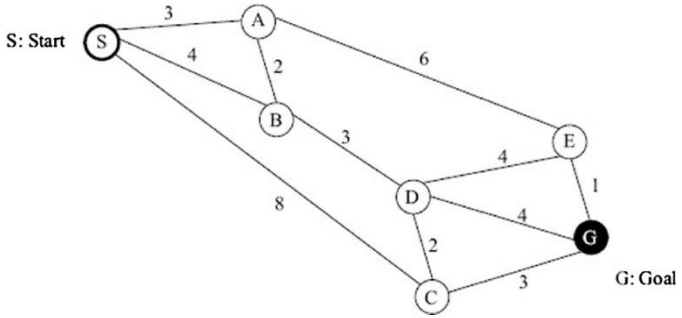

S: Start
S
A
B
C
D
E
G
G: Goal
3
4
2
6
3
4
1
8
4
2
3

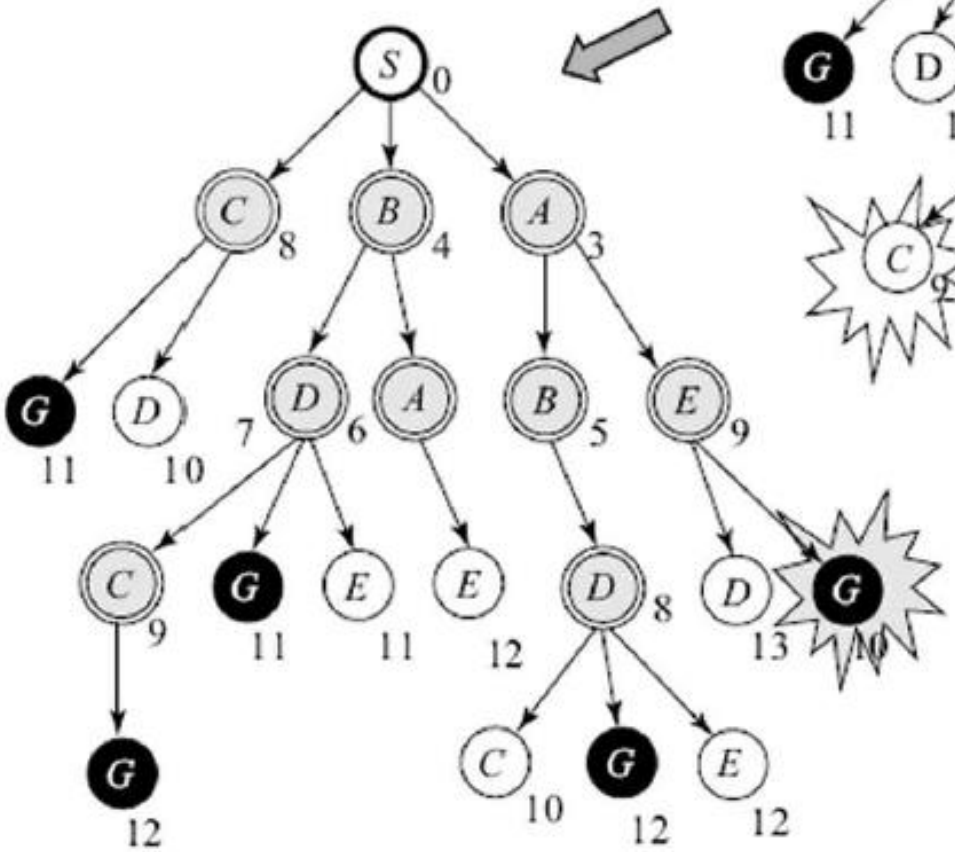

S 0
C 8
B 4
A 3
D 6
A 6
B 5
E 9
E 12
D 8

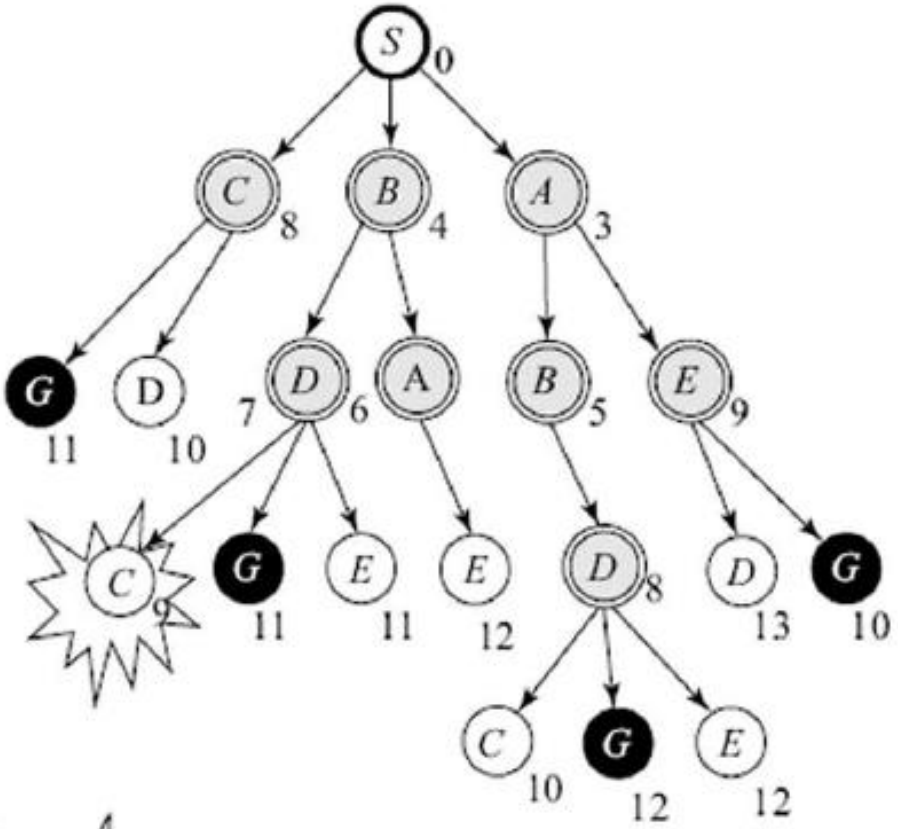

S 0
C 8
B 4
A 3
D 7
A 6
B 5
E 9
C 9
G 11
E 11
E 12
D 8

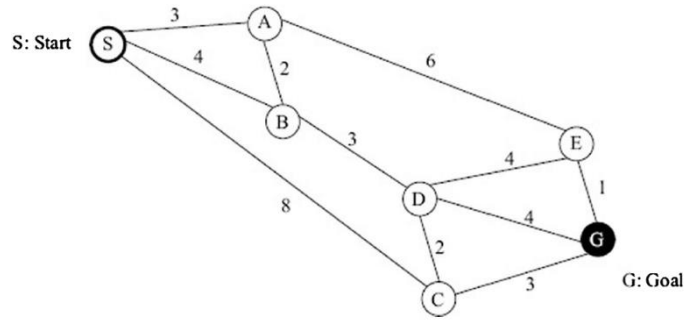

S: Start
S
A
B
C
D
E
G
G: Goal
3
4
2
6
3
4
1
8
4
2
3

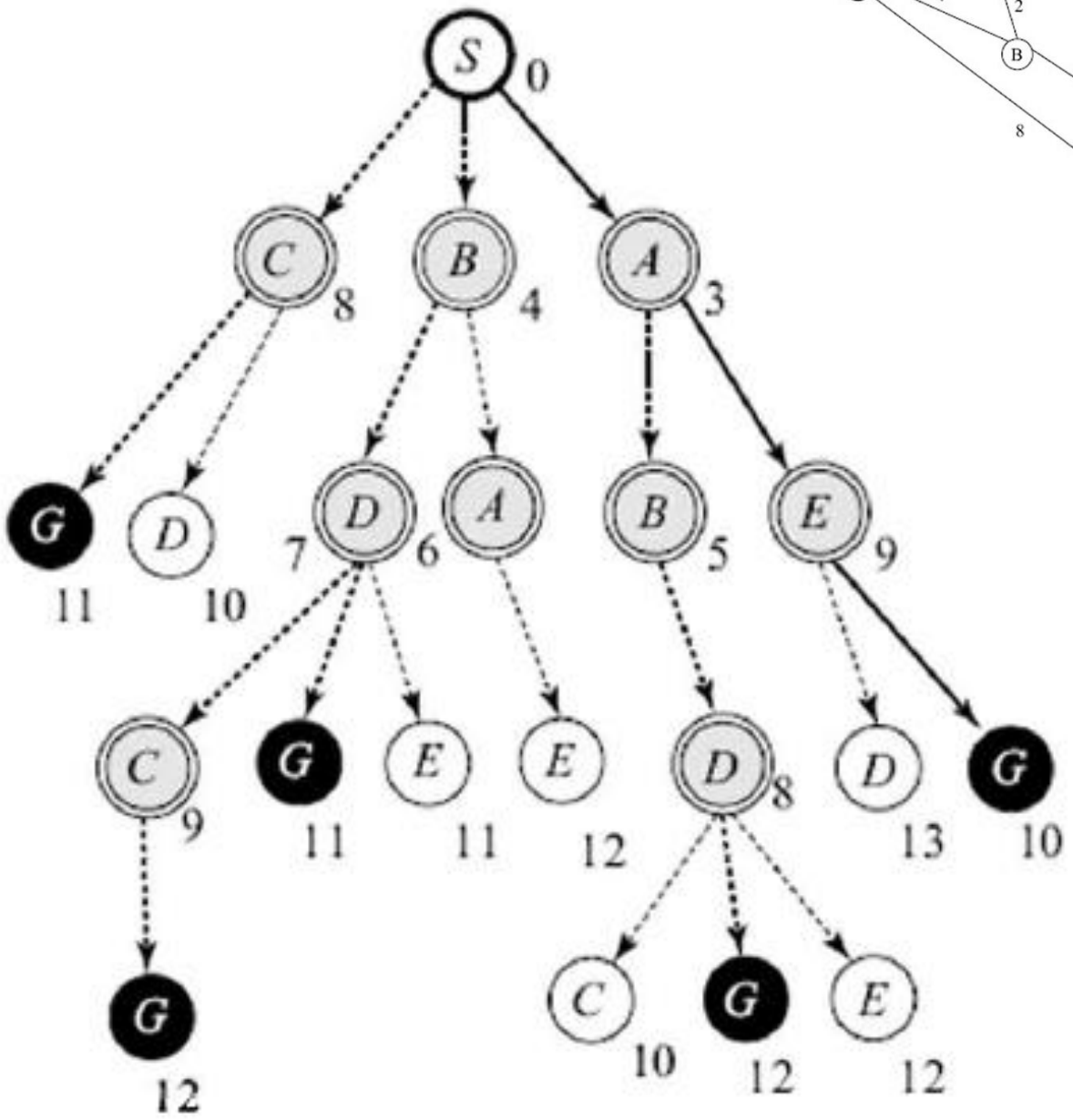

S 0
C 8
B 4
A 3
G 11
D 10
D 7
A 6
B 5
E 9
C
G 11
E 11
E 12
D 8
C 10
G 12
E 12


S: Start
S
A
B
C
D
E
G
G: Goal
3
4
2
6
3
4
1
8
4
2
3


S 0
C 8
B 4
A 3
G 11
D 10
D 7
A 6
B 5
E 9
C 9
G 11
E 11
E 12
D 8
D 13
G 10
C 10
G 12
E 12


S 0
C 8
B 4
A 3
G 11
D 10
D 7
A 6
B 5
E 9
C 9
G 11
E 11
E 12
D 8
D 13
G 10
C 10
G 12
E 12


S: Start
S
A
B
E
D
G
C
3
4
2
6
3
4
1
8
4
2
3
G: Goal


S 0
C 8
B 4
A 3
G 11
D 10
D 7
A 6
B 5
E 9
C 9
G 11
E 11
E 12
D 8
D 13
G 10
G 12
C 10
G 12
E 12

معايير تقويم عملية البحث

- الشمولية completeness والتي تعطينا فيما إذا كانت الاستراتيجية المستخدمة تضمن الوصول إلى حل، إذا كان هنالك من حل. وننطلق من عمق بحث ومعامل تفريع محددين وكلفة أصغرية للوحدة في حال كان عمق البحث لانهائياً.
- التعقيد الزمني time complexity كم نستغرق للوصول إلى حل في الحالة الأسوأ ويرتبط ذلك بعدد العقد الموسعة أو المنشورة.
- التعقيد المكاني space complexity يحدد بحجم الذاكرة التي نستغلها في الحالة الأسوأ، ويرتبط بالعدد الأعظمي للعقد في الذاكرة.
- الأمثلية optimality: هل نصل بشكل دائم إلى الحل ذي الكلفة الأقل.(العمق والممر الأفضل)

التعقيد الزمني والمكاني

يقاس التعقيد الزمني والمكاني من خلال:

- b: معامل التفريع الأعظمي لشجرة البحث (عدد العقد الأبناء الأعظمي انطلاقاً من العقدة التي تم توليدها).
- d: عمق الحل الأقل كلفة.
- m: العمق الأعظمي لفضاء الحالة (قد يكون لانهائي).

Strategies for State Space Search
استراتيجيات البحث في فضاء الحالة

Data-Driven and Goal-Driven Search A state space may be search in two directions:

from the given data of a problem instance toward a goal or from a goal back to the data.

- البحث المقاد بالبيانات و البحث المقاد بالهدف و يمكن البحث في فضاء الحالة في كلا اتجاهين : ينطلق من البيانات المعطاة لمشكلة ما والتقدم نحو الهدف أو ينطلق من الهدف عائدا إلى البيانات.

Strategies for State Space Search
استراتيجيات للبحث في فضاء الحالة

- البحث المستند إلى البيانات يأخذ حقائق المسألة ويطبق القواعد أو التحركات القانونية لإنتاج حقائق جديدة تؤدي إلى الهدف ؛
- يركز البحث المقاد بالهدف على الهدف، ويجد القواعد التي يمكن أن تنتج هذا الهدف، ويعود إلى الوراء بشكل متسلسل من خلال القواعد المتتالية والأهداف الفرعية للحقائق المعطاة للمسألة.

Strategies for State Space Search
استراتيجيات للبحث في فضاء الحالة

- بالإضافة إلى تحديد اتجاه البحث (المستند إلى البيانات أو الهدف) ، يجب أن تحدد خوارزمية البحث: الترتيب الذي يتم به فحص الحالات في الشجرة أو البيان . يأخذ هذا القسم في الاعتبار احتمالين للترتيب الذي يتم فيه أخذ عقد البيان في الاعتبار: بالعمق أولاً والبحث بالعرض أولاً .

Implementing Graph Search

- Backtracking search
- نبدأ بخوارزمية التراجع Backtracking لأنها واحدة من أولى خوارزميات البحث التي يدرسها علماء الكمبيوتر، ولها تطبيق طبيعي في بيئة تكرارية موجهة نحو المكدس . سنقدم نسخة أبسط من خوارزمية التراجع مع بحث بالعمق أولاً .

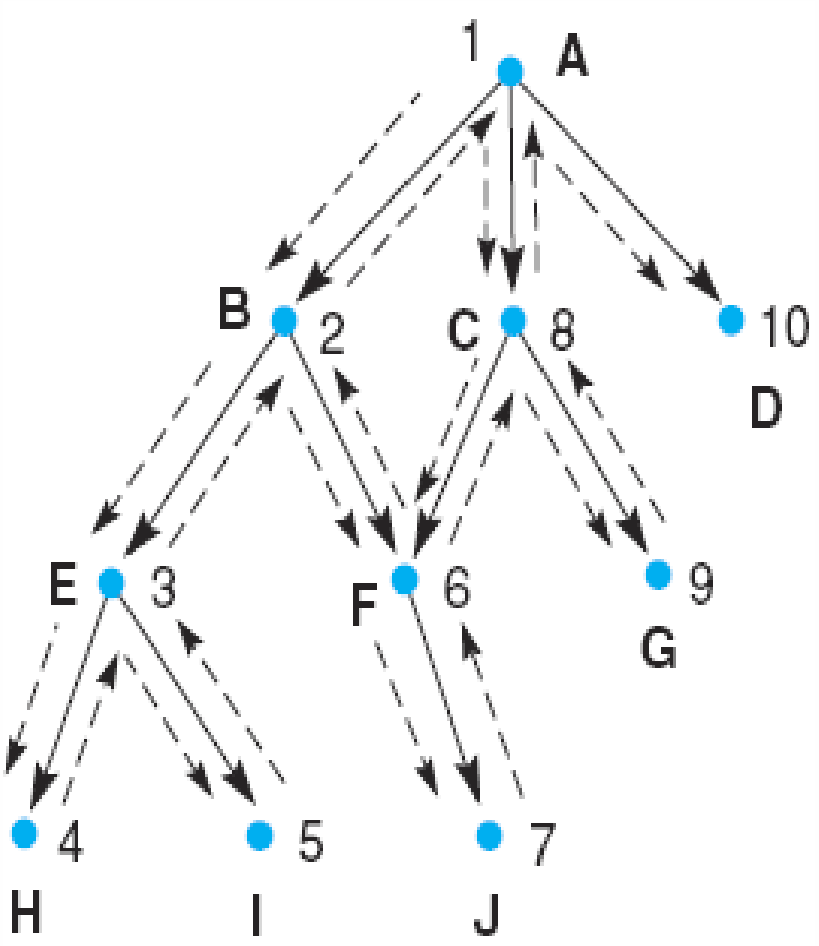
Backtracking search

يبدأ من حالة البداية ويسلك طريقًا حتى يصل إلى الهدف أو "طريق مسدود " إذا عثر على الهدف ، فإنه يتوقف ويعيد مسار الحل.

إذا وصلت إلى طريق مسدود، فإنها "تتراجع" إلى العقدة الأحدث على المسار التي بها أشقاء غير مختبرين وتستمر إلى الأسفل في أحد هذه الفروع ، على النحو التالي:

إذا كانت الحالة الحالية S لا تفي بمتطلبات وصف الهدف ، فقم بإنشاء سليلها (خلفها) الأول Schild1 وتطبيق إجراء التراجع بشكل متكرر على هذه العقدة .

إذا لم يعثر التراجع على عقدة هدف في البيان الفرعي في Schild1 ، كرر الإجراء لأخيه Schild2 , يستمر هذا حتى يصبح أحد سليل الابن عقدة هدف أو يتم البحث عن جميع الأبناء .إذا لم يؤدِ أي من أبناء S إلى الهدف، فإن التراجع يتم إلى والد S ، حيث يتم تطبيقه على شقيق S .....


1
A
B
2
C
8
10
D
E
3
F
6
9
G
4
5
7
H
I
J

Backtracking search

- تستمر الخوارزمية حتى تجد هدفًا أو تستنفد كامل فضاء الحالة. الشكل السابق يعرض خوارزمية التراجع المطبقة على فضاء حالة افتراضية . يشير اتجاه الأسهم المتقطعة على الشجرة إلى تقدم البحث لأعلى ولأسفل في الفضاء . يشير الرقم الموجود بجانب كل عقدة إلى الترتيب الذي تمت زيارته به . نحدد الآن خوارزمية التراجع، باستخدام ثلاث قوائم لتتبع العقد في فضاء الحالة:

Backtracking search

- SL، قائمة الحالة ، تعرض الحالات في المسار الحالي قيد التجربة. إذا تم العثور على هدف، فإن SL يحتوي على قائمة مرتبة من الحالات تمثل مسار الحل .
- NSL، قائمة الحالة الجديدة، تحتوي على عقد تنتظر التقييم ، أي العقد التي لم يتم إنشاء أحفادها والبحث فيها بعد.
- DE قائمة الطرق المسدودة، تعرض الحالات التي فشل أحفادها في احتواء هدف. إذا تمت مصادفة هذه الحالات مرة أخرى، فسيتم اكتشافها كعناصر من قائمة DE و إزالتها من الاعتبار على الفور.

Backtracking search

عند تعريف خوارزمية التراجع في الحالة العامة (بيان بدلاً من شجرة)، من الضروري اكتشاف تكرارات متعددة لأي حالة بحيث لا يتم إعادة إدخالها في المسار و تتسبب في حدوث حلقات (لانهائية). يتم تحقيق ذلك عن طريق اختبار كل حالة مولدة حديثًا فيما إذا كانت عضوا في أي من هذه القوائم الثلاث . إذا كانت حالة جديدة تنتمي إلى أي من هذه القوائم، فهي حالة قد تمت زيارتها بالفعل ويمكن تجاهلها .

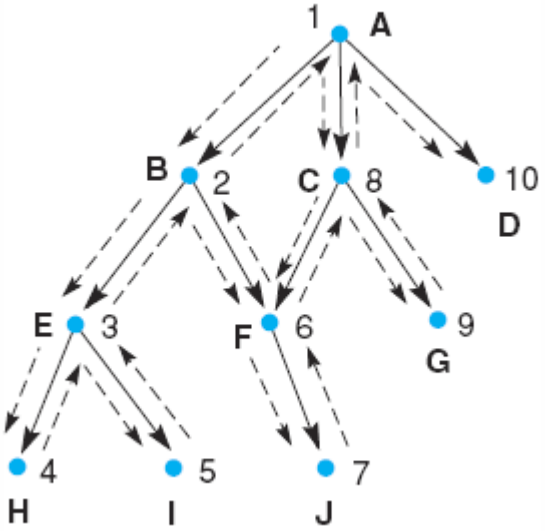
Function backtrack algorithm

```
function backtrack;

begin
   SL := [Start];  NSL := [Start];  DE := [ ];  CS := Start;                    % initialize:
   while NSL ≠ [ ] do                                  % while there are states to be tried
      begin
         if CS = goal (or meets goal description)
            then return SL;                    % on success, return list of states in path.
         if CS has no children (excluding nodes already on DE, SL, and NSL)
            then begin
               while SL is not empty and CS = the first element of SL do
                  begin
                     add CS to DE;                              % record state as dead end
                     remove first element from SL;                                %backtrack
                     remove first element from NSL;
                     CS := first element of NSL;
                  end
               add CS to SL;
            end
            else begin
               place children of CS (except nodes already on DE, SL, or NSL) on NSL;
               CS := first element of NSL;
               add CS to SL
            end
      end;
      return FAIL;
end.
```

بحث التقصي العكسي

Backtracking search



Initialize: SL = [A]; NSL = [A]; DE = [ ]; CS = A;

AFTER ITERATION	CS	SL	NSL	DE
0	A	[A]	[A]	[ ]
1	B	[B A]	[B C D A]	[ ]
2	E	[E B A]	[E F B C D A]	[ ]
3	H	[H E B A]	[H I E F B C D A]	[ ]
4	I	[I E B A]	[I E F B C D A]	[H]
5	F	[F B A]	[F B C D A]	[E I H]
6	J	[J F B A]	[J F B C D A]	[E I H]
7	C	[C A]	[C D A]	[B F J E I H]
8	G	[G C A]	[G C D A]	[B F J E I H]

Backtracking search

- في حالة التراجع, تمثل CS(الحالة الحالية) الحالة التي تمت إضافتها مؤخرًا إلى SL وتمثل "حدود" مسار الحل الذي يتم استكشافه حاليًا. يتم ترتيب قواعد الاستدلال والتحركات وتطبيقها على CS. و النتيجة هي مجموعة منظمة من الحالات الجديدة، أبناء CS. يتم وضع أول هؤلاء الأبناء في الوضع الحالي الجديد ويتم وضع البقية بالترتيب في NSL للفحص المستقبلي. تتم إضافة الحالة الحالية الجديدة إلى SL و يستمر البحث .إذا لم يكن لـ CS أي أبناء، فسيتم إزالته من SL وأي أبناء متبقية من سابقتها على SL يتم فحصها.

Backtracking search

- استخدام قائمة بالحالات غير المعالجة (NSL) للسماح للخوارزمية بالعودة إلى أي من هذه الحالات.
- قائمة بالحالات "السيئة (DE) "لمنع الخوارزمية من إعادة محاولة المسارات غير المجدية.
- قائمة بالعقد (SL) على مسار الحل الحالي التي يتم إرجاعها إذا تم العثور على هدف
- عمليات فحص صريحة لعضوية الحالات الجديدة في هذه القوائم لمنع التكرار.

تصنيفات البحث

- لا معرفي (غير موجه) (أعمى) Uninformed Search (blind)
- معرفي (موجه) Informed Search

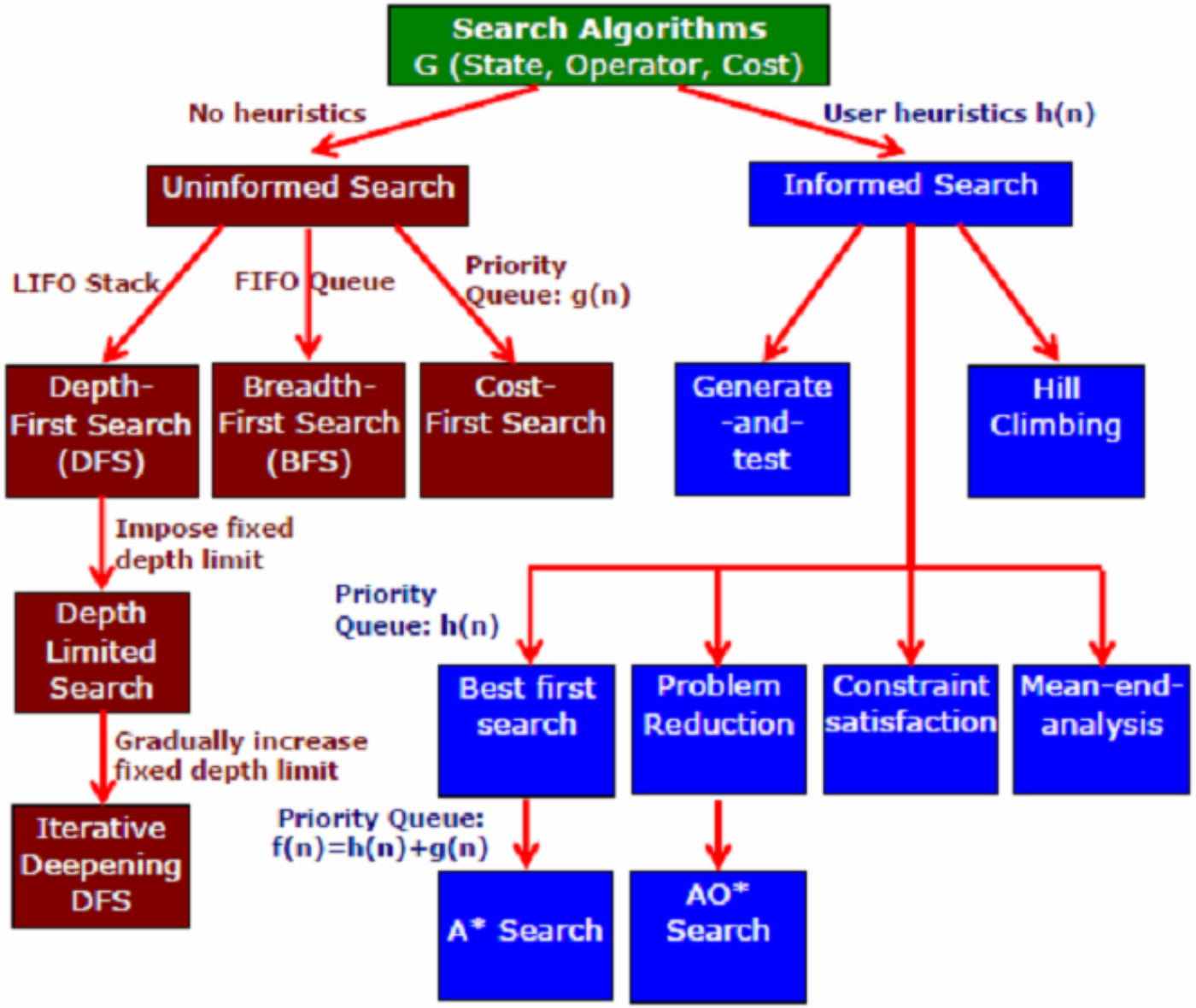


Fig. Different Search Algorithms

البحث اللامعرفي (الأعمى)
Uninformed Search

- نعرف البحث اللامعرفي (الأعمى) على أنه البحث الذي يستخدم استراتيجيات البحث المعتمدة حصراً على معلومات متوفرة في التعريف بالمسألة.

استراتيجيات البحث اللامعرفي (الأعمى)

- نطلق تسمية الأعمى Blind لسببين:
- عدم وجود معلومات عن عدد الخطوات أو كلفة ***الممر من الحالة المدروسة إلى الهدف.***
- عدم وجود تابع تقويم الممر (عند وجود هذا التابع نطلق على البحث تسمية البحث الاجتهادي heuristic).

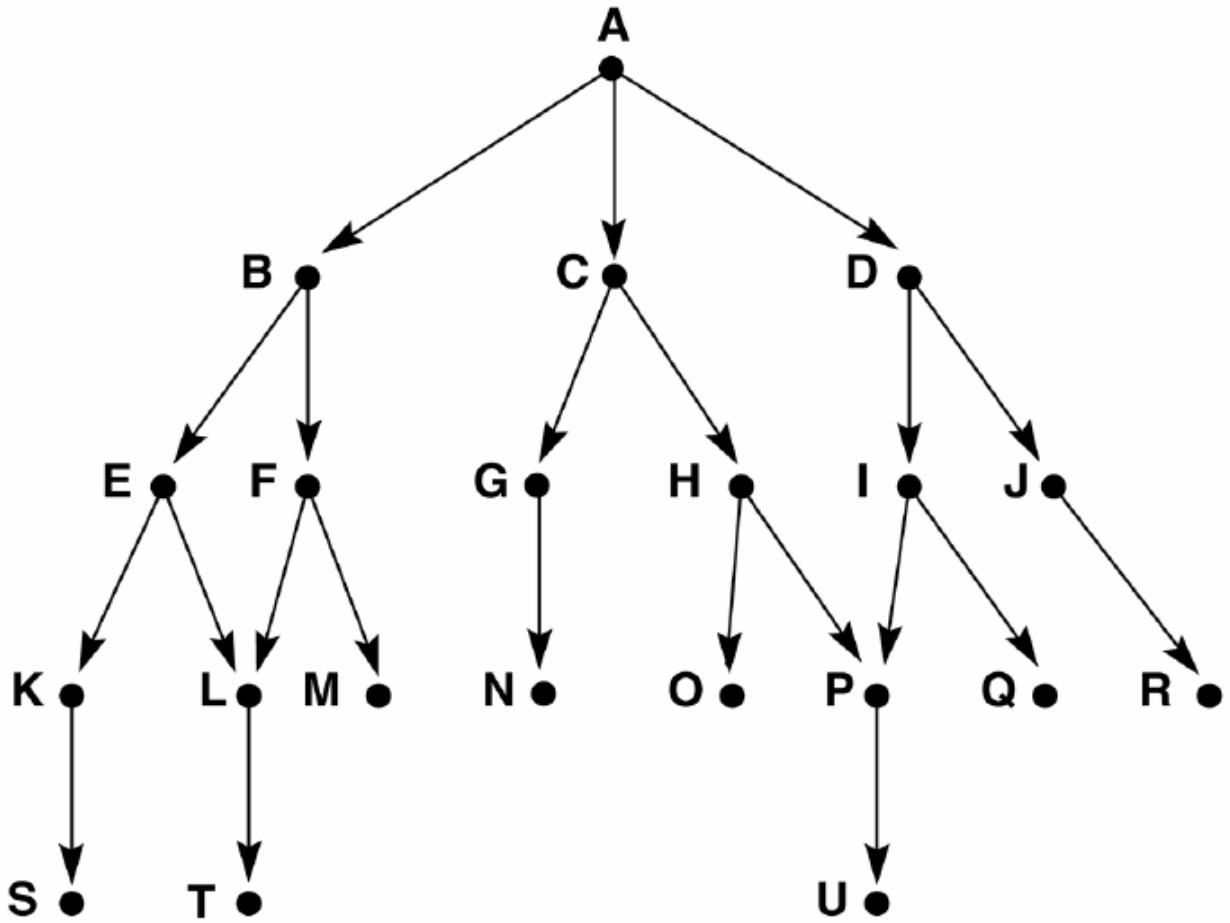
استراتيجيات البحث اللامعرفي (الأعمى)

Depth-first search or backtracking	البحث في العمق أولاً أو تقصي عكسي
Depth-limited search	البحث محدد العمق
Iterative Depth-limited search	البحث محدد العمق التكراري
Breadth-first search	البحث بالعرض أولاً
Uniformed Cost-first Search	البحث الموحد بالكلفة أولاً

البحث بالعمق أولا
Depth First Search
(DFS)

- when a state is examined, all the children and their descendants are examined before any of its siblings
- depth-first search goes deeper into the search space whenever this is possible
- only when no further descendants of a state can be found are its siblings considered

- عند اختبار الحالة، يتم اختبار جميع الأولاد وذريتهم قبل أي من إخوتها.
- يتعمق البحث في العمق أولا في فضاء البحث كلما كان ذلك ممكنًا.
- فقط عندما لا يمكن العثور على أحفاد حالة أخرى، يتم النظر إلى إخوتها



- **in the order A, B, E, K, S, L, T, F, M, C, G, N, H, O, P, U, D, I, Q, J, R. The backtrack algorithm implemented depth-first search**

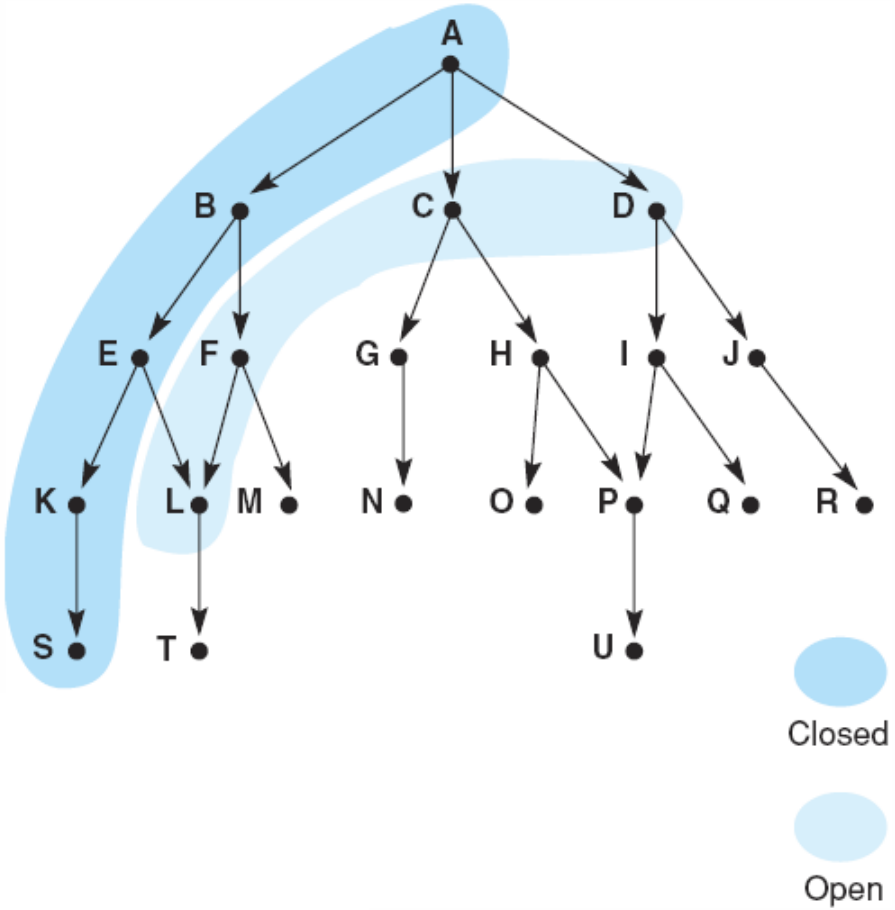
البحث بالعمق أولا
Depth First Search (DFS)

- نقوم بتنفيذ بحث واسع النطاق باستخدام القوائم ، المفتوحة والمغلقة ، لتتبع التقدم في فضاء الحالة open، تقابل NSL في البحث بالتراجع ، القوائم التي تم إنشاؤها ولكن لم يتم فحص أبنائهم . يحدد الترتيب الذي تتم به إزالة الحالات من قائمة الفتح ترتيب البحث .
- قائمة مغلق closed تسجل الحالات التي فحصت بالفعل . قائمة مغلق هو اتحاد قوائم DE و SL لخوارزمية البحث بالتراجع .

البحث في العمق أولاً

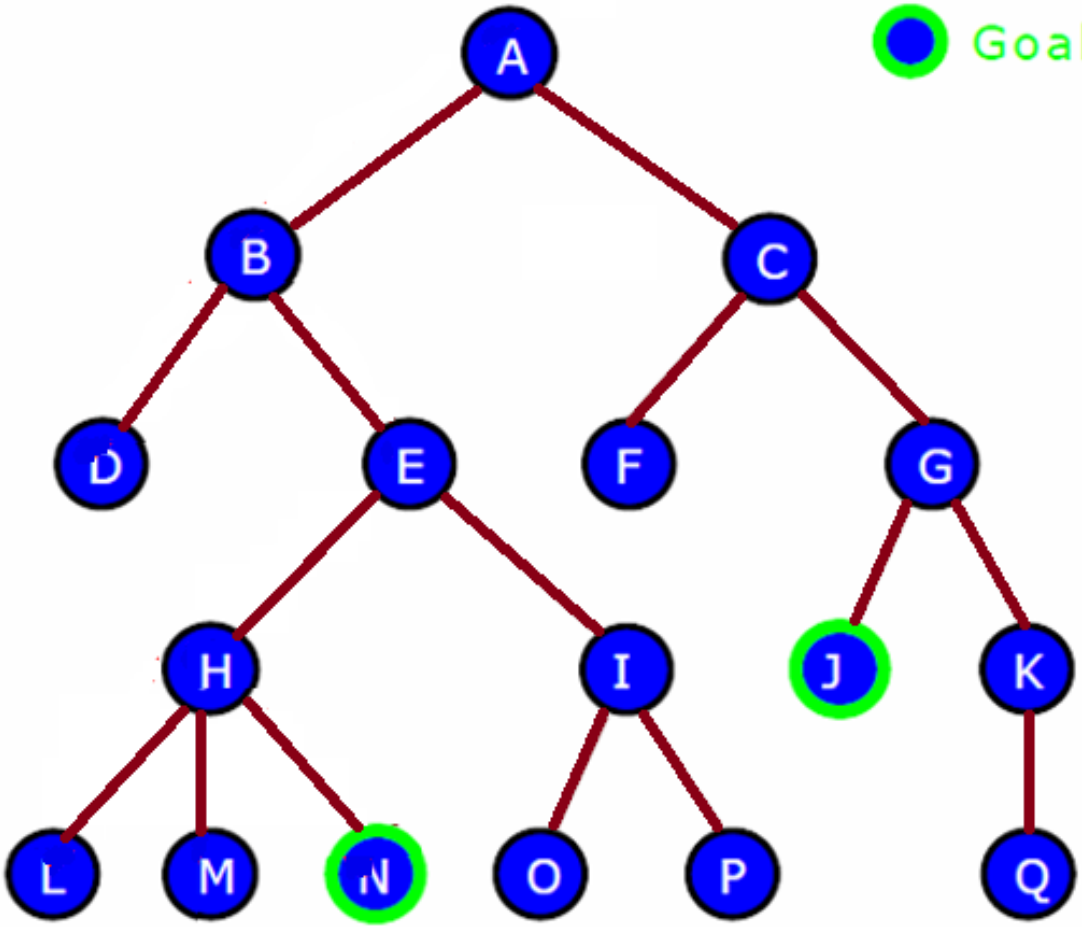
```
begin
   open := [Start];                                                    % initialize
   closed := [ ];
   while open ≠ [ ] do                                                 % states remain
      begin
         remove leftmost state from open, call it X;
         if X is a goal then return SUCCESS                            % goal found
            else begin
               generate children of X;
               put X on closed;
               discard children of X if already on open or closed;     % loop check
               put remaining children on left end of open              % stack
            end
      end;
   return FAIL                                                         % no states left
end.
```

البحث في العمق أولاً
أو تقصي عكسي



1. open = [A]; closed = [ ]
2. open = [B,C,D]; closed = [A]
3. open = [E,F,C,D]; closed = [B,A]
4. open = [K,L,F,C,D]; closed = [E,B,A]
5. open = [S,L,F,C,D]; closed = [K,E,B,A]
6. open = [L,F,C,D]; closed = [S,K,E,B,A]
7. open = [T,F,C,D]; closed = [L,S,K,E,B,A]
8. open = [F,C,D]; closed = [T,L,S,K,E,B,A]
9. open = [M,C,D], as L is already on closed; closed = [F,T,L,S,K,E,B,A]
10. open = [C,D]; closed = [M,F,T,L,S,K,E,B,A]
11. open = [G,H,D]; closed = [C,M,F,T,L,S,K,E,B,A]

مثال :شجرة بحث بالعمق أولا Depth-first search tree



مثال :شجرة بحث بالعمق أولا Depth-first search tree

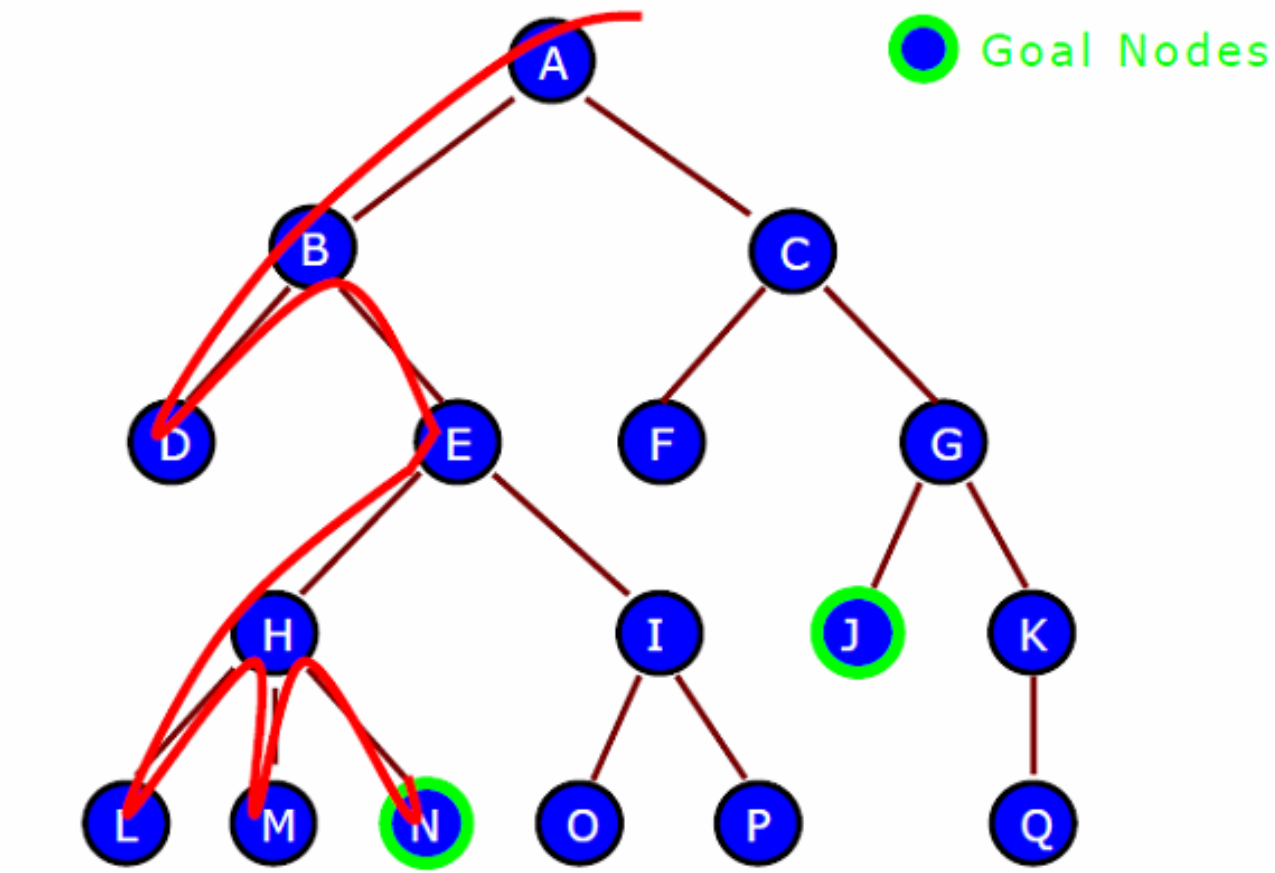


Fig. Depth-first search (DFS)

واضح من الشكل أعلاه أن النقاط تم استكشافها بالترتيب كما يلي:

A B D E H L M N I O P C F G J K Q

بعد البحث في النقطة A ومن ثم B ثم D يتراجع البحث ويحاول مسارا آخر انطلاقا من النقطة B، وعلى ذلك فسيتم الوصول إلى النقطة الهدف N قبل النقطة الهدف J.

البحث محدد العمق التكراري
Iterative Depth-limited search

تضمن هذه التقنية إيجاد عقدة هدف ذات عمق أصغري (إذا كان بالإمكان إيجاد هدف). في العمق التكراري تجري عمليات بحث متتابعة بطريقة العمق أولا (أي لكل بحث حد للعمق يزيد عن الآخر بقيمة 1)إلى أن نجد عقدة هدف. و نلاحظ أن عدد العقد الموسعة بواسطة البحث بطريقة التعمق التكراري لا تزيد كثيرا عن عدد العقد التي توسع بواسطة البحث عرضا أولا.

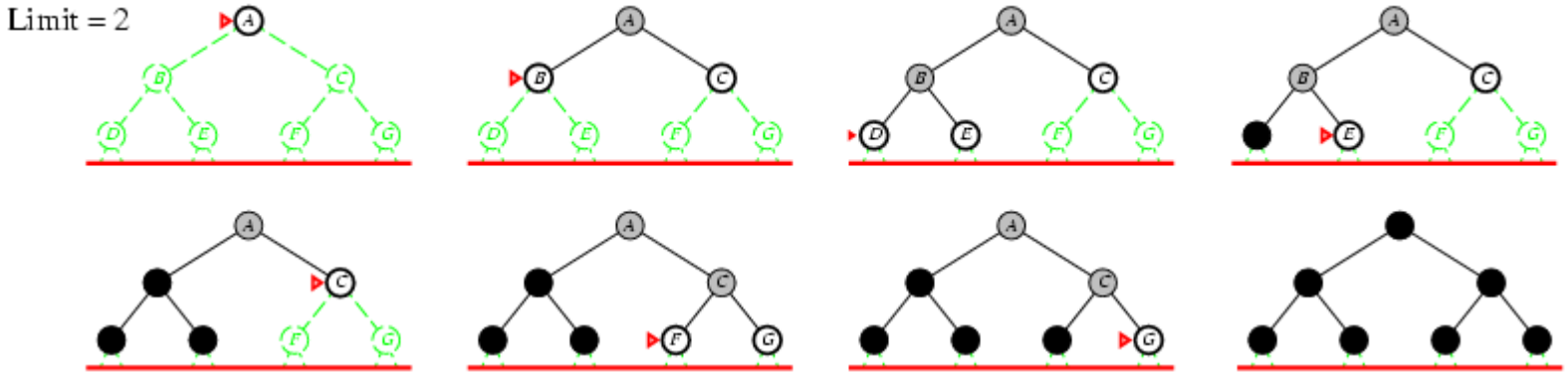
Iterative Deepening Search / =0

$\text{Limit} = 0$

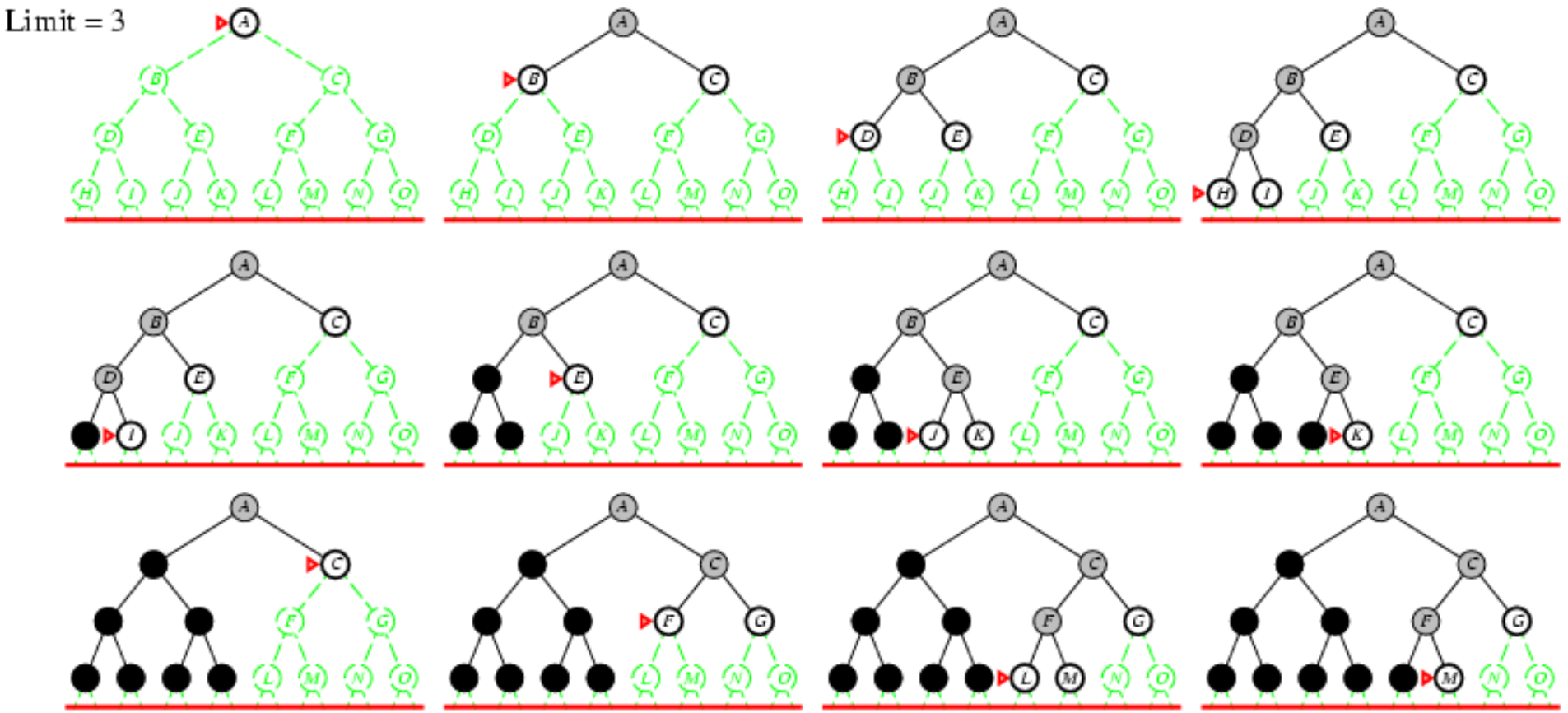
Iterative Deepening Search / =1

Limit = 1

Iterative Deepening Search / =2



Iterative Deepening Search / =3



البحث بالعرض أولا
Breadth First Search(BFS)

In breadth-first search

- explores the space in a level-by-level fashion
- only when there are no more states to be explored at a given level does the algorithm move on to the next level

في البحث بالعرض أولا

- يستكشف الفضاء بطريقة مستوى بمستوى
- فقط عندما لا يكون هناك المزيد من الحالات ليتم استكشافها على مستوى معين ، تنتقل الخوارزمية إلى المستوى التالي

في هذه الخوارزمية : توضع العقد المولدة في رتل FIFO الداخل أولا يخرج أو لا

البحث بالعرض أولا
Breadth First Search(BFS)

مثال لشجرة البحث بالعرض أولا Breadth–first search tree

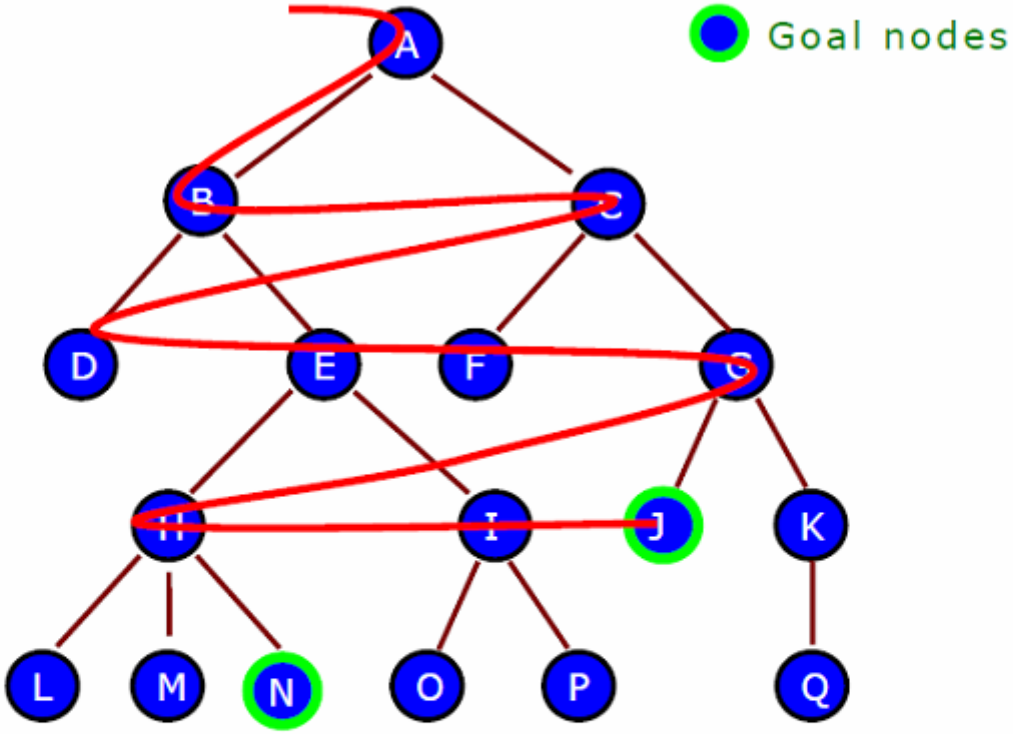


Fig. Breadth-first search (BFS)

لاحظ أن النقاط المستكشفة وفق البحث بالعرض هي:

A B C D E F G H I J K L M N O P Q

بعد البحث في النقطة A ثم B ثم C، يستمر البحث إلى المستوى الثاني في النقطة D ثم E ...حتى نهاية المستوى الثاني، ثم ينزل إلى المستوى الثالث مادامت النقطة الهدف لم تستكشف بعد، ونلاحظ من الشجرة أعلاه أن النقطة الهدف J قد اكتشفت قبل الوصول للنقطة الهدف N .

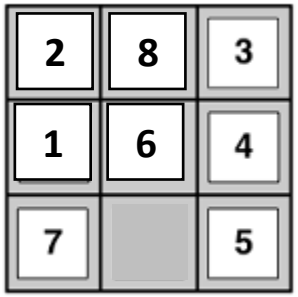
البحث بالعرض أولاً

```
function breadth_first_search;

begin
   open := [Start];                                          % initialize
   closed := [ ];
   while open ≠ [ ] do                                       % states remain
      begin
         remove leftmost state from open, call it X;
            if X is a goal then return SUCCESS               % goal found
               else begin
                  generate children of X;
                  put X on closed;
                  discard children of X if already on open or closed;   % loop check
                  put remaining children on right end of open           % queue
               end
      end
   return FAIL                                               % no states left
end.
```

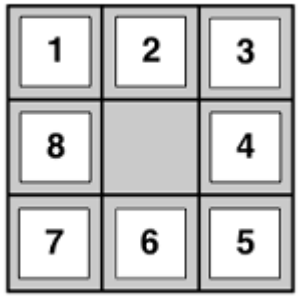
مسألة أحجية القطع الثمانية
8-puzzle

- المطلوب الوصول إلى الحالة الهدف انطلاقاً من حالة ابتدائية ما.



Start State

حالة ابتدائية



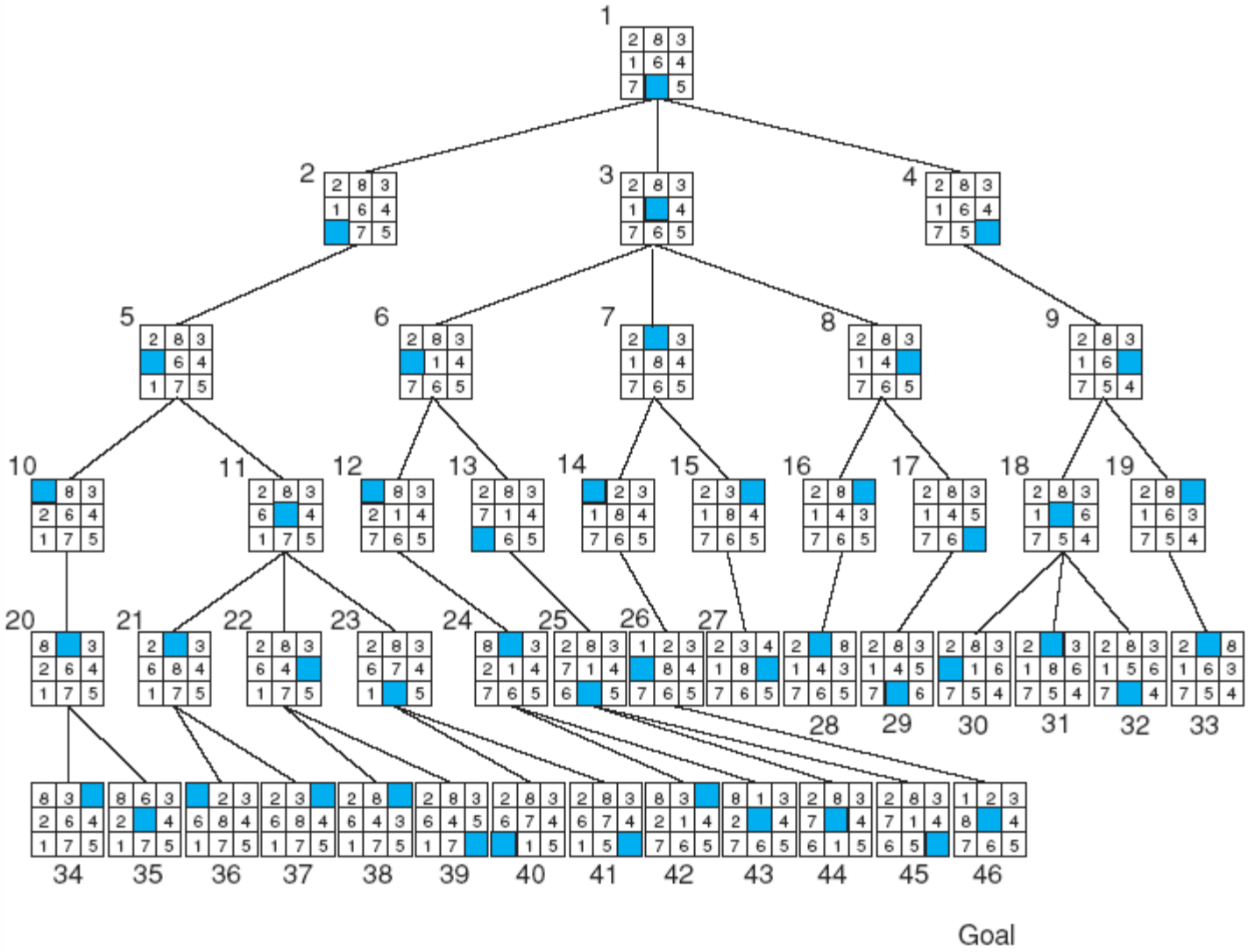
Goal State

الحالة الهدف

مسألة أحجية القطع الثمانية
8-puzzle

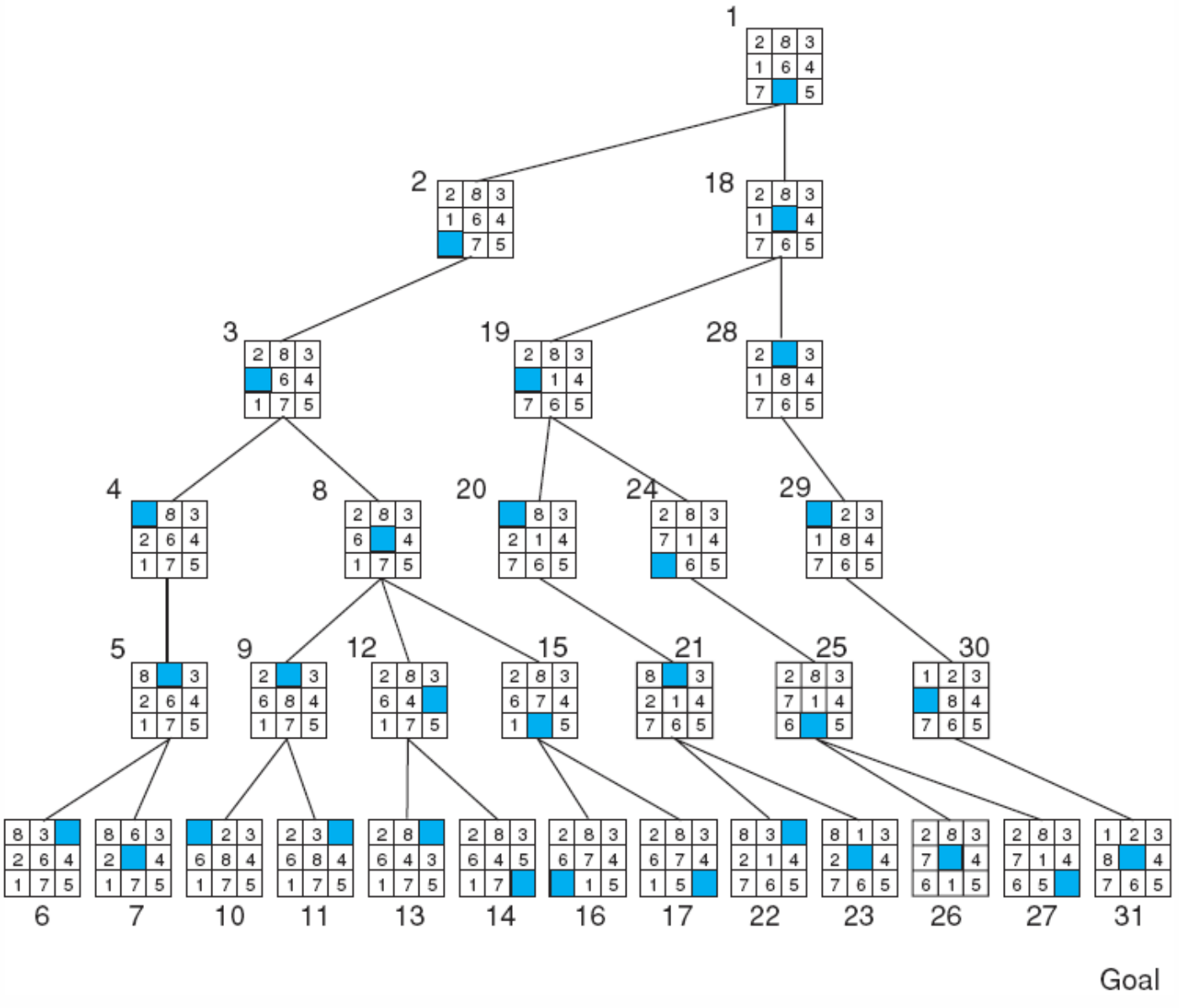
يوضح الشكل الحالات التي تمت إزالتها من قائمة الفتح وفحصها في البحث بالعرض أولا لبيان **8-puzzle** . تتوافق الأقواس مع تحركات الفراغ للأعلى و لليمين و للأسفل و لليسار . يشير الرقم الموجود بجوار كل حالة إلى الترتيب الذي تمت إزالته به من قائمة الفتح . لا تظهر الحالات التي تُركت مفتوحة عند توقف الخوارزمية.

مثال البحث بالعرض أولاً



يعطي الشكل التالي بحثًا بالعمق أولًا عن **(8-puzzle)** .كما في السابق، يتم إنشاء الفضاء بواسطة قواعد "الانتقال إلى الفراغ" الأربعة (للأعلى و للأسفل ولليسار ولليمين). تشير الأرقام الموجودة بجانب الحالات إلى الترتيب الذي تم تقييمها به، أي إزالتها من قائمة الفتح . لا يتم عرض الحالات التي تُركت مفتوحة عند العثور على الهدف . تم فرض حد عمق 5 على هذا البحث لمنعه من الضياع في أعماق الفضاء.

مثال البحث في العمق أولاً بعمق بحث = 5



مقارنة بين البحث بالعرض أولا وبالعمق أولا

يعتمد اختيار "العمق أولاً" أو "العرض أولا" ً على المشكلة المحددة التي يتم حلها.

- تشمل الميزات الهامة أهمية العثور على أقصر طريق إلى الهدف ، وعامل التفرع ضمن الفضاء ، وموارد حساب الوقت والمكان المتاحة ، ومتوسط طول المسارات إلى عقدة الهدف ، وما إذا كنا نريد جميع الحلول أو الحل الأول فقط
- عند اتخاذ هذه القرارات ، هناك مزايا وعيوب لكل نهج

مقارنة بين البحث بالعرض أولا وبالعمق أولا

بالعرض أولا

- يختبر دائمًا جميع العقد في المستوى n قبل المتابعة إلى المستوى n + 1 ، يبحث البحث بالعرض أولا دائمًا عن أقصر مسار إلى عقدة الهدف.
- في حال كان هناك حل بسيط للمشكلة ، يتم إيجاد هذا الحل.
- لسوء الحظ ، إذا كان هناك عامل تفريع سيئ ، أي أن الحالة لديها متوسط عدد مرتفع من الأحفاد ، فقد يمنع الانفجار التوافقي الخوارزمية من إيجاد حل باستخدام الفضاء المتاح و يرجع هذا إلى حقيقة أن جميع العقد لكل مستوى من مستويات البحث يجب أن تظل مفتوحة.
- بالنسبة لعمليات البحث العميقة ، أو فضاء الحالة ذي عامل التفرع العالي ، يمكن أن يصبح هذا مرهقًا للغاية.
- يعد استخدام فضاء البحث بالعرض أولاً ، المقاس من حيث عدد الحالات المفتوحة ، دالة أسية لطول المسار في أي وقت
- إذا كان لكل حالة متوسط عدد الأبناء B ، فهناك B^n states في المستوى

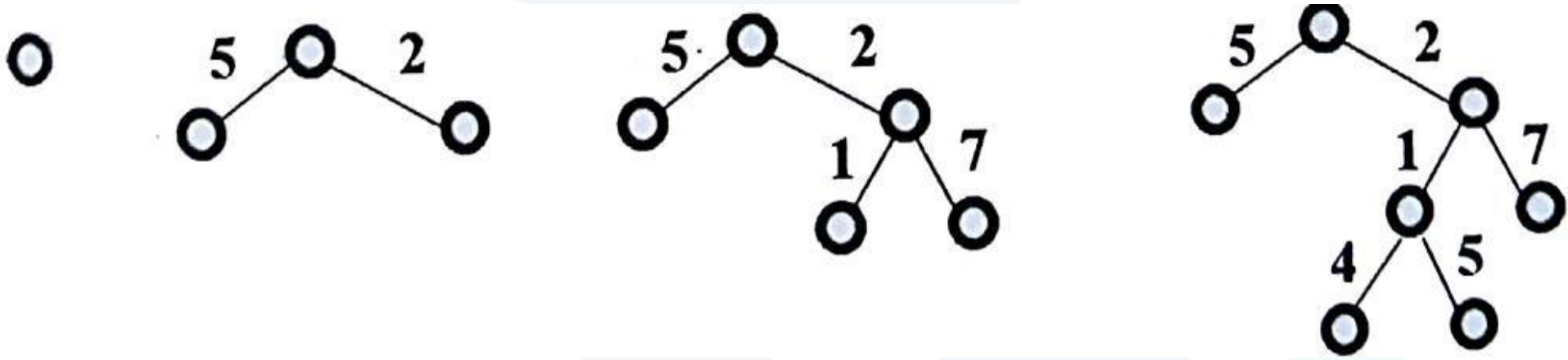
مقارنة بين البحث بالعرض أولا وبالعمق أولا

بالعمق أولا

- يبدأ البحث في العمق أولاً في الوصول بسرعة إلى فضاء بحث عميق.
- إذا علمت أن مسار الحل سيكون طويلاً، فلن يضيع البحث بالعمق أولًا الوقت في البحث عن عدد كبير من الحالات "السطحية" في البيان.
- يمكن أن "يضيع" بحث العمق أولاً في عمق البيان، أو يفقد المسارات الأقصر للوصول إلى الهدف أو حتى أن يعلق في مسار طويل بلا حدود .
- إنها أكثر فاعلية لفضاء البحث الذي يحتوي على العديد من الفروع لأنه لا يتعين عليها الاحتفاظ بجميع العقد عند مستوى معين في القائمة المفتوحة.
- استخدام الفضاء للبحث في العمق أولاً هو دالة خطية لطول المسار إذا كان لكل حالة متوسط عدد الأبناء B لكل حالة، فإن هذا يتطلب استخدامًا إجماليًا لفضاء B x n states للانتقال إلى مستويات n في عمق الفضاء.

البحث بالكلفة الموحدة أولاً
Uniformed Cost –first Search

تعتمد هذه الخوارزمية على مبدأ تطوير العقدة ذات الكلفة الأقل و بالتالي ان أي عقدة هدف نصل اليها تكون الحل الأمثل.



خوارزمية البحث بالكلفة الموحدة أولاً: هنا ننتقل للعقدة ذات الكلفة التراكميّة الأقل مع ملاحظة أننا نقارن بكلف أوراق الشجرة كلها وليس المستوى الذي وصلنا له فقط أي أننا نستخدم رتل أفضلية (تطوير لخوارزمية البحث بالعرض أولا من خلال تحسين الرتل ليصبح رتل مع أفضلية)

خواص خوارزمية البحث بالكلفة الموحدة أولاً
Uniformed Cost-First Search

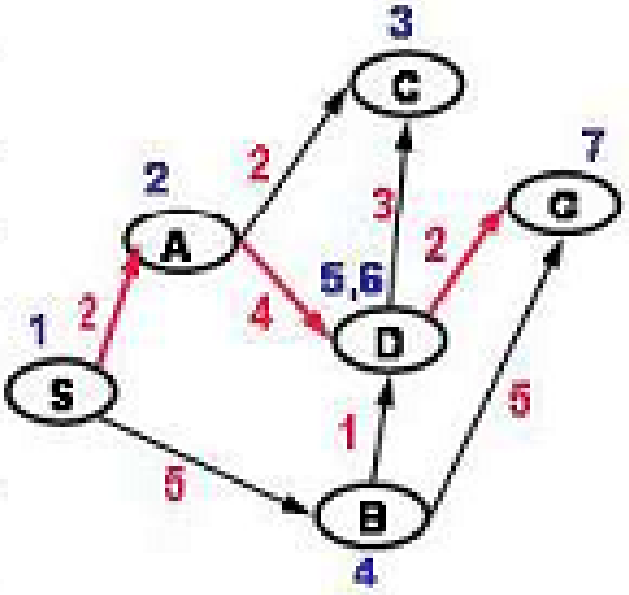
بشكل عام تقوم بزيارة العقد التي لم تتم زيارتها ذات الكلفة الأقل . و يمكن تحقيق هذه المنهجية عن طريق تخزين العقد المكتشفة و التي لم تتم زيارتها ضمن Priority queue الرتل ذو الأولوية وتعتمد فكرة الأولوية هنا على كلفة الطريق (الأصغر إلى الأكبر).

ملاحظة :

بسبب أهمية كلفة الطريق في هذه المنهجية لا نقوم بتطبيق مفهوم visited كما في الخوارزميتين السابقتين بل سيتم تخزين الطريق الأقصر وعند فحص المسارات المتبقية ضمن فضاء الحلول اذا كان هناك طريق آخر أقصر من الطريق المخزن يتم الاحتفاظ به وإهمال الحل القديم (الطريق السابق.)

البحث بالكلفة الموحدة أولاً
Uniform Cost - first Search

	Q
1	(0 S)
2	(2 A S) (5 B S)
3	(4 C A S) (6 D A S) (5 B S)
4	(6 D A S) (5 B S)
5	(6 D B S) (10 G B S) (6 D A S)
6	(8 G D B S) (9 C D B S) (10 G B S) (6 D A S)
7	(8 G D A S) (9 C D A S) (8 G D B S) (9 C D B S) (10 G B S)



Note:

Expand the cheapest unexpanded node

Implementation :

frontier = priority queue ordered by path cost g (n)

معايير تقويم عملية البحث

- الشمولية completeness والتي تعطينا فيما إذا كانت الاستراتيجية المستخدمة تضمن الوصول إلى حل، إذا كان هنالك من حل. و ننطلق من عمق بحث ومعامل تفريع محددين وكلفة أصغرية للوحدة في حال كان عمق البحث لانهائياً.
- التعقيد الزمني time complexity كم نستغرق للوصول إلى حل في الحالة الأسوأ ويرتبط ذلك بعدد العقد الموسعة أو المنشورة.
- التعقيد المكاني space complexity يحدد بحجم الذاكرة التي نستغلها في الحالة الأسوأ، ويرتبط بالعدد الأعظمي للعقد في الذاكرة.
- الأمثلية optimality: هل نصل بشكل دائم إلى الحل ذي الكلفة الأقل (العمق والممر الأفضل) .

التعقيد الزمني والفراغي

يقاس التعقيد الزمني والفراغي من خلال:

- b: معامل التفريع الأعظمي لشجرة البحث (عدد العقد الأبناء الأعظمي انطلاقاً من العقدة التي تم توليدها).
- d: عمق الحل الأقل كلفة.
- m: العمق الأعظمي لفضاء الحالة (قد يكون لانهائي).

خواص خوارزمية البحث بالعمق أولاً
DFS

1- Complete :

- إن خوارزمية DFS مكتملة و منتهية .

2- Optimality :

- كما لاحظنا أن خوارزمية DFS تهدف إلى إيجاد طرق من حالة البداية إلى حالة النهاية ولكن ليس بالضرورة حل أمثلي .

3-Time :

- يمكن التعبير عن الكلفة الزمنية b^m حيث b هو معامل التفرع و m هي عمق شجرة البحث.

4- Space :

- يمكن التعبير عن الكلفة المكانية بما يلي : $b*m$

خواص خوارزمية البحث بالعرض أولاً
BFS

1- Complete :

- هذه الخوارزمية مكتملة كسابقتها

2- Optimality :

- تضمن خوارزمية البحث بالعرض إيجاد حل أمثلي

3- Time :

- يمكن التعبير عن الكلفة الزمنية b^{d+1} حيث تعبر b عن عامل التفرع و تعبر d عن عمق الحل في شجرة البحث.

4- Space :

- يمكن التعبير عن الكلفة المكانية b^{d+1}

خواص خوارزميةالبحث بالكلفة الموحدة أولاً
Uniformed Cost - First Search

1- Complete :

- هذه الخوارزمية مكتملة كسابقتها وذلك في حال كانت الأوزان أكبر أو تساوي الصفر.

2- Optimality :

- تضمن هذه الخوارزمية (UCS) إيجاد حل أمثلي.

3- Time :

- يمكن التعبير عن الكلفة الزمنية $O(b^{c/\epsilon})$ حيث c هي كلفة الحل الأمثل و ε وسطي كلفة الخطوة.

4- Space :

- يمكن التعبير عن الكلفة المكانية $O(b^{c/\epsilon})$.

البحث المعرفي
Informed Search

- **البحث المعرفي :** يعني اتمام البحث باستخدام معارف تخص المسألة قيد البحث

* خوارزميات البحث الأعمى لا تستخدم أي معلومات تخص هيكلية شجرة البحث من أجل تحسين كفاءة البحث و هي تقوم بالبحث ضمن فضاء الحالة حتى الوصول لحل.

* و بالتالي فإن حل المعضلات الحقيقية اعتمادا على هذه الخوارزميات سيؤدي إلى انفجار حسابي (عدد الحالات الممكنة كبير).

* تستخدم خوارزميات البحث الحدسي Heuristic كل المعلومات المتاحة لجعل عملية البحث أكثر كفاءة.

* المعلومات الحدسية Heuristic هي قاعدة أو طريقة تؤدي عموما إلى تحسين كفاءة عملية البحث.

Informed Search

- **يعتمد البحث المعرفي على حساب دالة التقويم**
- و يقيس تابع التقويم البعد عن الهدف.
- يتوجب بعدها اختيار العقدة التي تحصل على أفضل تقويم بما يوحي بأنها العقدة التالية الأفضل في السعي للوصول إلى الهدف.
- يتم تحقيق ذلك من خلال استخدامنا لقائمة نرتب بها العقد التالية ترتيبا تنازليا وفقا لدرجات أفضلياتها.
- و تم دراسة دوال تقويم مطورة وفقا لتقنيتي البحث بأفضل أولا (الجشع) best-first search(greedy) و البحث وفقا لخوارزمية A^* search.
- تعتبر الخبرة , البساطة و التوقع المدروس من العوامل التي تقلل من حيز البحث في فضاء الحلول.

Informed Search

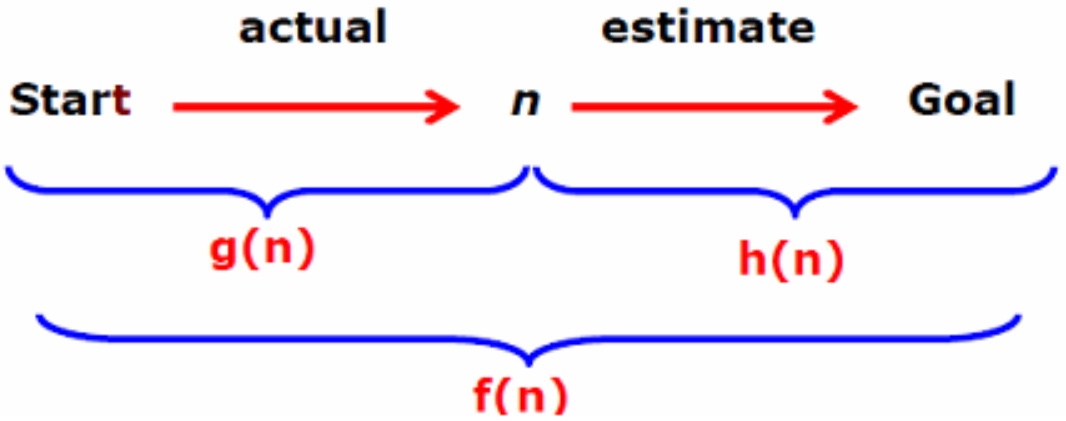
- نستخدم بعض التعاريف و التوابع التي تعرف بحدود المسألة المطروحة مثل:
- تابع التقويم evaluation function ***f(n)*** التابع الذي يحدد لنا نظريا الخطوة التالية. و يعطى بالعلاقة

$$f(n)=g(n)+h(n)$$

- ***h(n)***: الكلفة التقديرية للوصول من العقدة n (قيد الدراسة)إلى الهدف
- ***g(n)***: كلفة الوصول إلى العقدة n.
- ***f(n)***: الكلفة الكلية التقديرية للممر عبر العقدة n وصولا إلى الهدف
- ***h(n)=0*** العقدة قيد البحث هي عقدة الهدف

مقارنة

$$f(n) = g(n) + h(n)$$



- مقارنة بين البحث باستخدام خوارزميات الاستدلال و البحث باستخدام خوارزميات البحث الأعمى

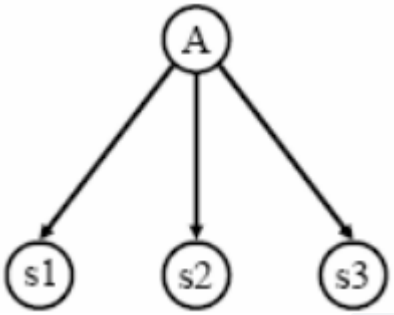
Heuristic search	Blind search
- يقوم بتقدير المسافة إلى الحالة الهدف - يرشد عمليات البحث باتجاه الهدف. - يميز الحالات الأقرب مسافة إلى الحالة الهدف.	- لديها معرفة فقط عن الحالات التي قامت باكتشافها من قبل. - لا توجد معرفة حول بعد أي حالة عن الحالة الهدف

الاجتهاد أو الاستكشاف أو الاستدلال
Heuristic

- **الاستدلال Heuristic:** هو تابع عندما يطبق على حالة يعيد رقم يقدر قرب هذه الحالة من الهدف.
- أي ان الاستدلال يخبر تقريبا كم تبقى للوصول الى الهدف (الأرقام الأصغر هي الأفضل)

أما بالنسبة للبحث في البيانات فإن الاستدلال يعني قواعد اختيار الفروع في فضاء الحالة للوصول إلى حل معقول للمسألة.

- مثال:
- للعقدة A ثلاث عناصر تالية بفرض التابع الحدسي لكل منها :
- $h(s1) = 0.8 \quad h(s2) = 2.0 \quad h(s3) = 1.6$
- متابعة البحث مرورا بالحالة s1 حدسيا هو الأفضل



Heuristic

- سوف نرى أن التجريبيات التي نعتبرها مقبولة admissible هي التجريبيات التي يكون تقديرها دائما أصغر من الكلفة الحقيقية .
- يستخدم الاجتهاد في حل مسائل الذكاء الصنعي في حالتين:
- **الأولى :** في حل المسائل التي ليس لها حل دقيق بسب الغموض المتأصل inherent ambiguities في المسألة أو البيانات المتوفرة. و نقصد بالغموض هنا إمكانية السير بمسارين مختلفين بنفس الشروط مما قد يقود إلى الابتعاد عن الحل.
- و يعتبر التشخيص الطبي أحد الأمثلة على ذلك, إذ يمكن أن يكون لنفس المجموعة من الأعراض أسباب مختلفة مما يدفع بالطبيب لاستخدام الاجتهاد لاختيار التشخيص الأرجح و يضع الخطة الملائمة للعلاج.

Heuristic

- و تعتبر الرؤية vision مثلا آخر على الحالات التي يظهر بها الالتباس أو الغموض و الذي يدفعنا إلى استخدام الاجتهاد, إذ تتصف المشاهد المرئية بالغموض و الالتباس مما يسمح بالتفسيرات المتعددة لصفات و حدود الكينونات المشاهدة, و يعتبر الخداع البصري في مقدمة هذه الالتباسات الغامضة. و هنا يأتي دور الاجتهاد ليجلو غموض المشهد و يرجح التفسير الذي يبدو أكثر منطقية.
- **الثانية:** نستخدم الاجتهاد عندما يكون هنالك نموذج رياضي للمسألة إلا أن العمل بهذا النموذج يجعلنا نعاني من كلفة (حسابية) فائقة قد تؤدي إلى جعل العمل به شبه مستحيل (كما هو عليه في مسائل الشطرنج التي توصلنا إلى الانفجار التراكبي (Combinatorial Explosion)) مما يجعل العمل بالاجتهاد أمرا لابد منه.
- **البحث الحدسي =البحث الاجتهادي = البحث الاستدلالي = البحث الاستكشافي = البحث التجريبي = Heuristic Search**

البحث بأفضل أول البسيط *greedy local search*
تسلق الهضبة (Hill climbing)

- يعتبر تسلق الهضبة أحد أبسط استراتيجيات البحث الاجتهادي. تعتمد هذه الاستراتيجية على توليد أبناء العقد و تقويم أبنائها, و انتقاء الابن الأفضل لتكرار عملية التوليد دون أخذ بقية العقد سواء كانت عقدة أب أو ابن أو عقدة أخوة أو غير ذلك بعين الاعتبار. و من الممكن الوصول إلى منطقة لا يمكن للمتسلق المتابعة بعدها وتدعى هذه النقطة قمة محلية Local Maxima و يتطلب من المتسلق في هذه الحالة الانتقال إلى طريق ممكن آخر, و يعتبر الوصول (الوقوع) في القمة المحلية أحد عيوب هذه الطريقة و بزيادة الخبرة يمكن تجنب الوصول إلى قمة محلية.

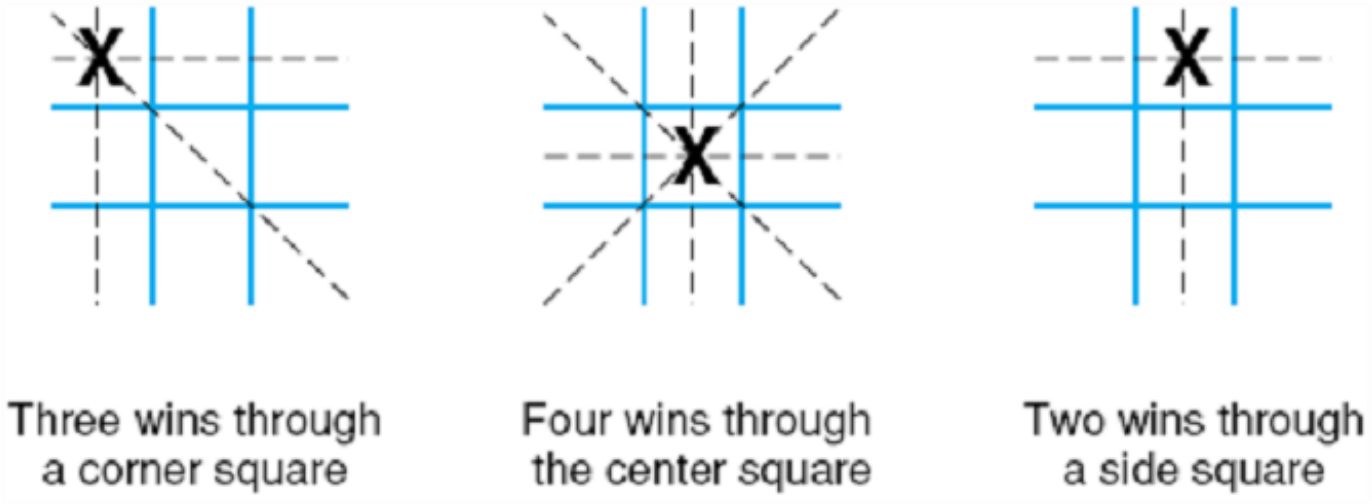
البحث بأفضل أول البسيط *greedy local search*
تسلق الهضبة (Hill climbing)

- إذ أن الخوارزمية الأولية لتسلق الهضبة لا تستطيع تذكر المراحل السابقة مما قد يؤدي إلى الحيرة و حتى إلى التوقف القسري عند الوصول إلى نقطة لا تسمح بالمتابعة لأنها تشكل تابع التقويم الأفضل, أي ليس هناك عقد ذات تابع تقويم أفضل.
- تعتبر لعبة Tic- Tac- Toe مثلا جيدا عن تسلق القمة (الهضبة), إذ يتبع اللاعب بشكل أولي الطريق الذي يقربه من الربح دون التروي و النظر في الحالة التي سيحققها المنافس, إذ ليس المطلوب في هذه الحالة تقصي الأخطاء و الاستفادة منها.

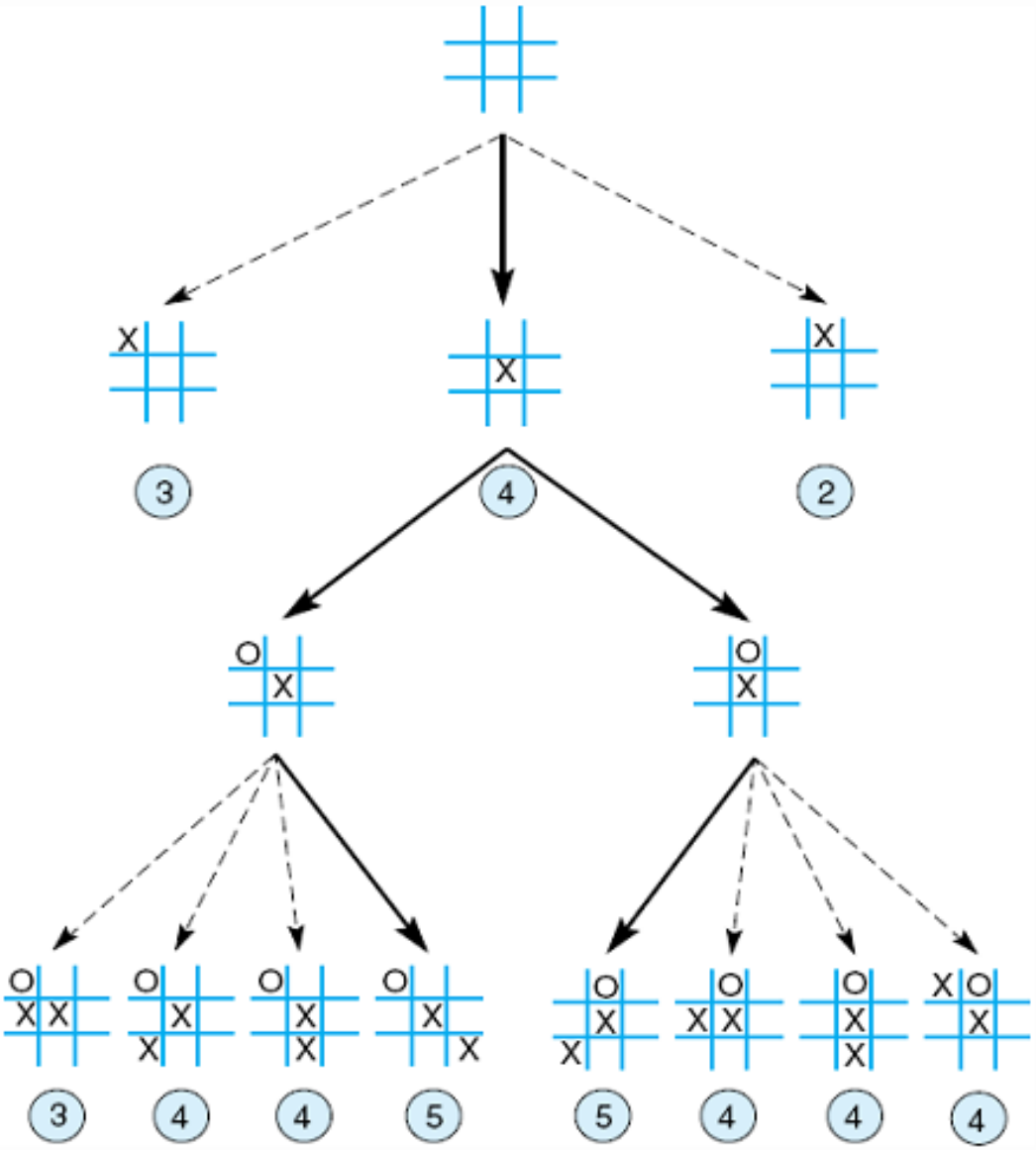
البحث بأفضل أول البسيط *greedy local search*
تسلق الهضبة (Hill climbing)

- لعبة Tic- Tac- Toe تشكل مسألة بفضاء حالة يصل إلى $9! = 362880$ حالة.
- و أحد طرق حلها يعتمد على فكرة القيام بالنقلة التي تقربنا من ربح محتمل.

- تابع التقويم المتبع هو حساب عدد خطوط الربح و اختيار النقلة ذات عدد خطوط الربح الأكبر كما هو مبين في الشكل.

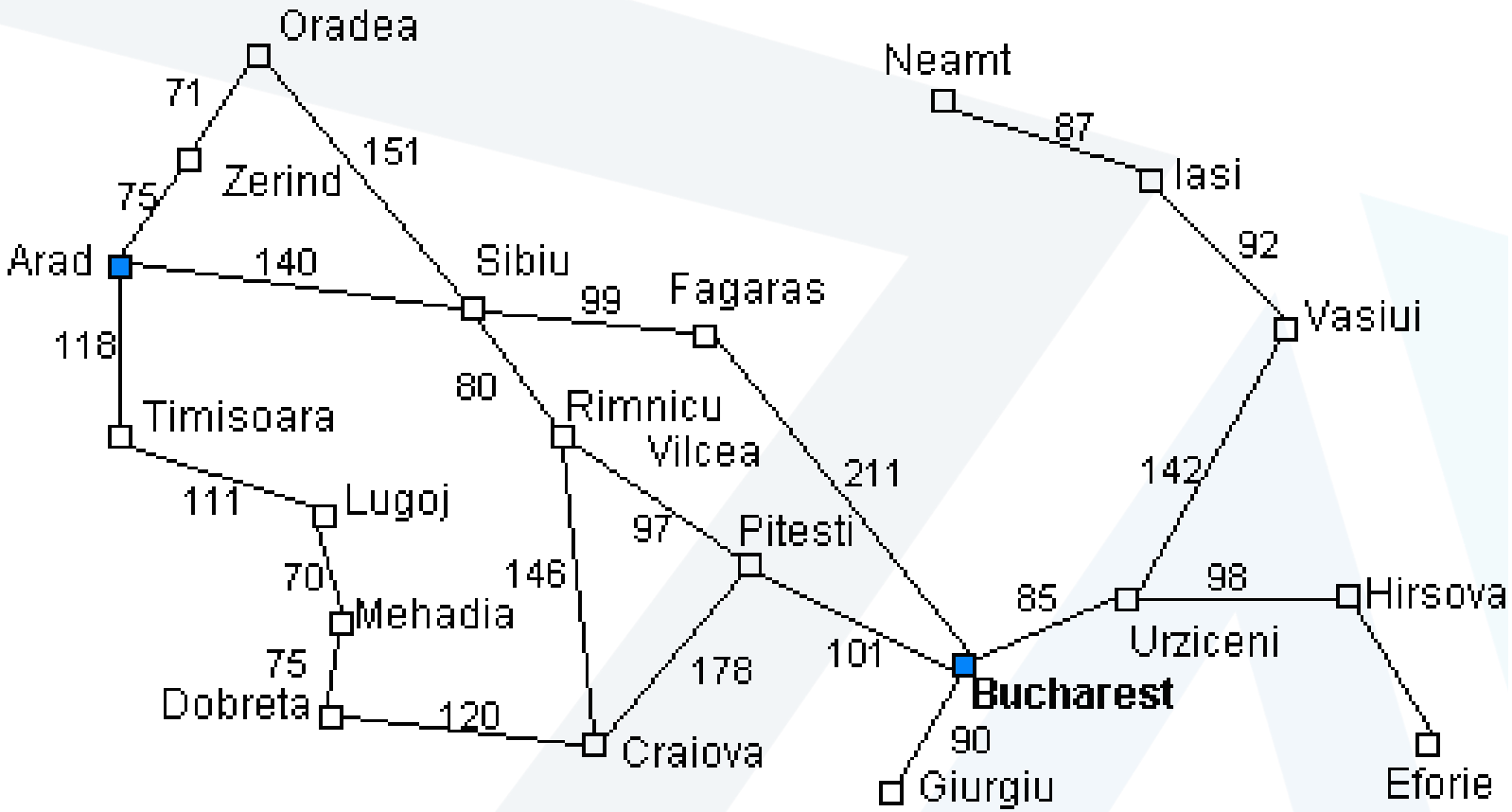


البحث بأفضل أول البسيط *greedy local search*
تسلق الهضبة (Hill climbing)



عندها يمكننا تقليل
عدد خطوات البحث
وفقا لهذا الاجتهاد

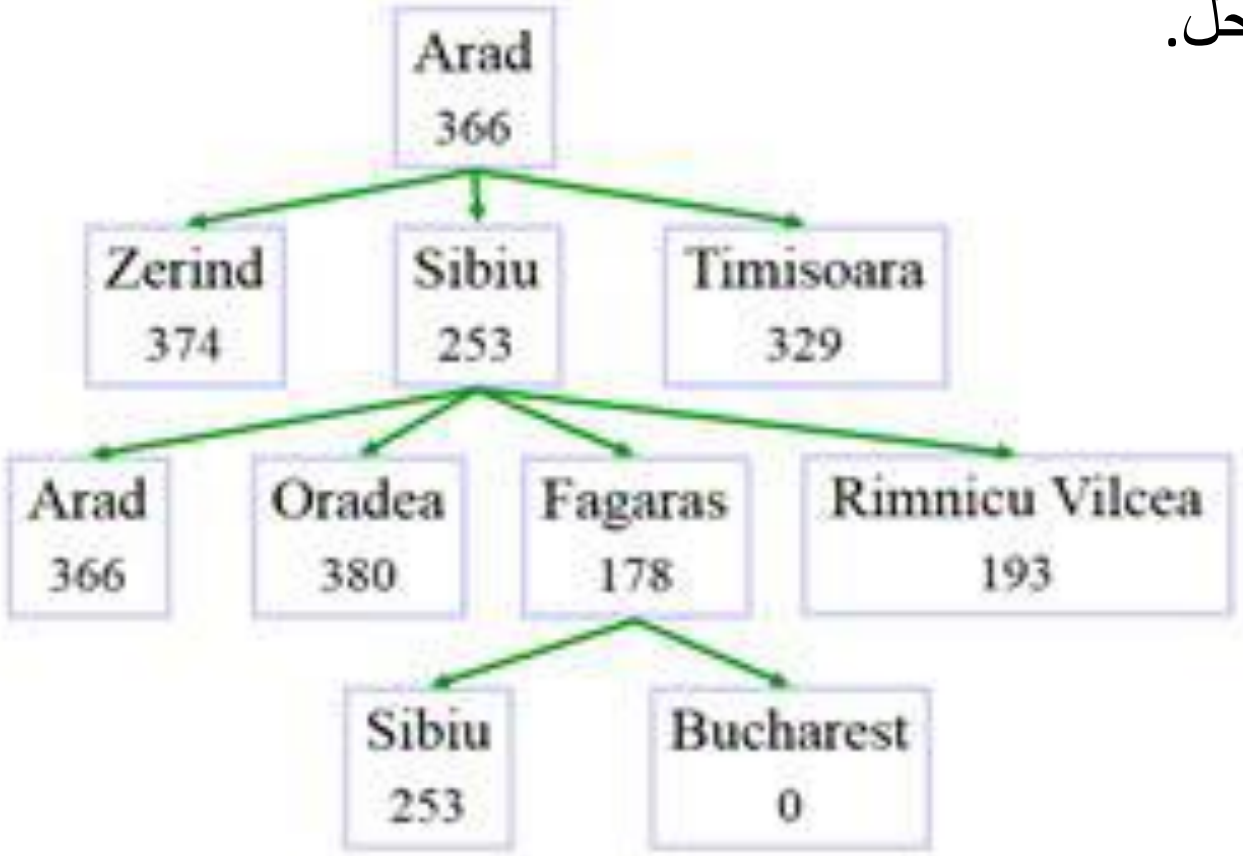
البحث بأفضل أول البسيط *greedy local search*
تسلق الهضبة (Hill climbing)



Straight-line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

البحث بأفضل أول البسيط *greedy local search*
تسلق الهضبة (Hill climbing)

هذه الشجرة هي عبارة عن شجرة الحل لمسألة خريطة رومانيا .يعبر الرقم الموجود في أسفل كل عقدة(مربع)عن الكلفة التقديرية للتجريبية(خط النظر .)ونلاحظ كيفية اختيار العقدة ذات الكلفة الأقل عند كل مستوى لنولد لها أبناء إلى أن نصل للحل.



البحث بأفضل أول البسيط *greedy local search*
تسلق الهضبة (Hill climbing)

- مثال : استخدام خوارزمية تسلق الهضبة (القمة) مع لعبة Pazzle -8:
- بدايةً دعونا نختار التجريبية الملائمة لطبيعة المسألة:
- 1-التجريبية الأولى:
- هي عدد المربعات التي لا تقع بمكانها الصحيح ففي المربع التالي ستكون قيمة التجريبية هي: H(n)=1 .

البحث بأفضل أول البسيط *greedy local search*
تسلق الهضبة (Hill climbing)

1	2	3
4	5	6
7	8	

Goal

1	2	3
4	5	6
7		8

- 2-التجريبية الثانية (Manhattan Distance) هي التي سنقوم باختيارها:

1	2	3
4	5	6
7		8

هي عدد الحركات اللازمة

لإيصال كل في غير موضعه

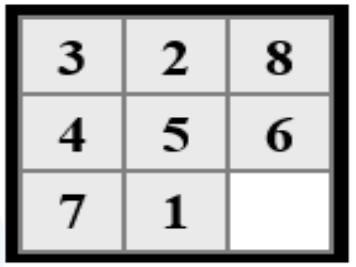
إلى مكانه الصحيح وستكون

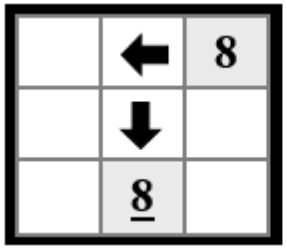
قيمة التجريبية هي H(n)=1 للمربع التالي

البحث بأفضل أول البسيط *greedy local search*
تسلق الهضبة (Hill climbing)

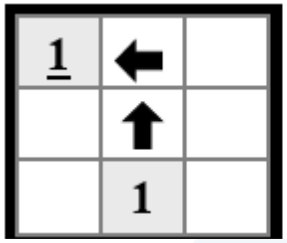
نلاحظ أنه كلاً من المربعات 1 , 3 , 8 ليست في مكانها الصحيح . وسنقوم بحساب عدد الحركات اللازمة لعودة كلاً من هذه المربعات لمكانها كما يلي:

Goal

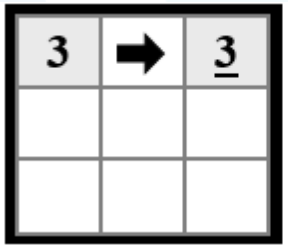




3 spaces



3 spaces

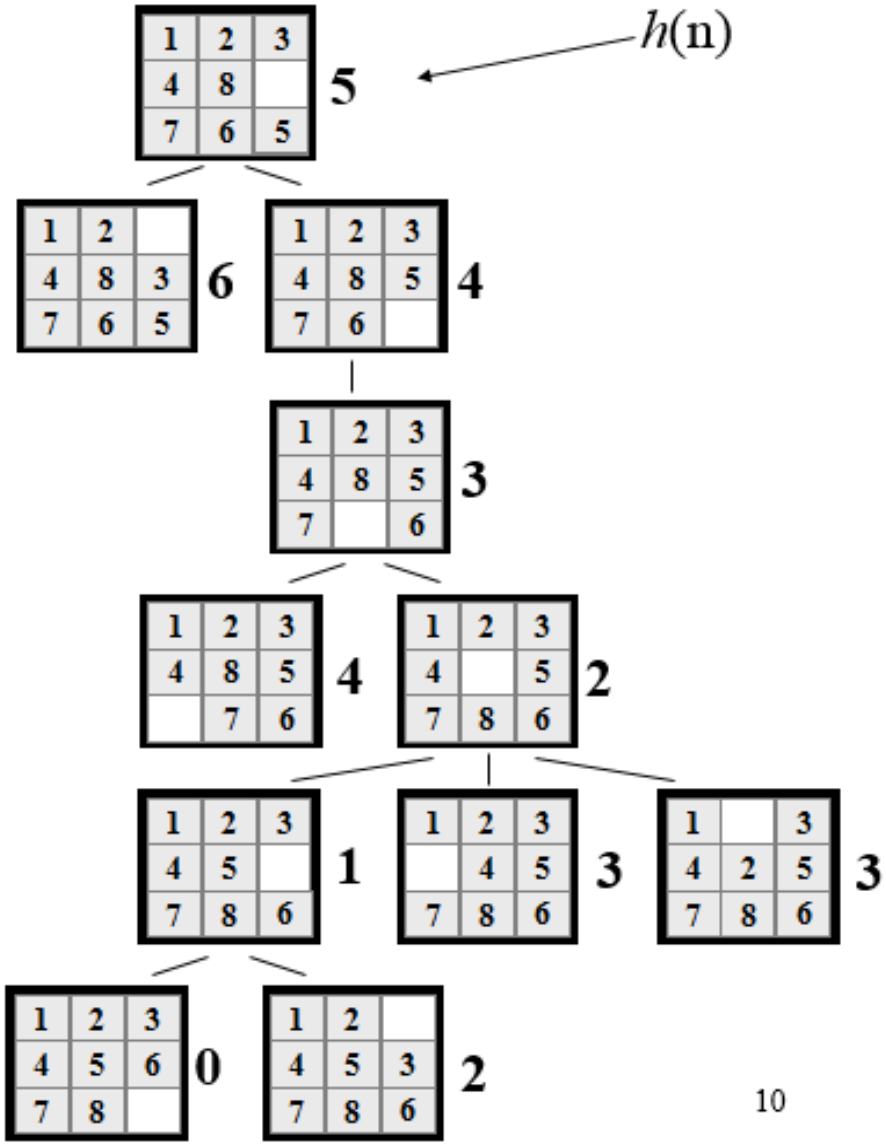


2 spaces

Total= 8 (للأشكال الثلاثة السابقة)

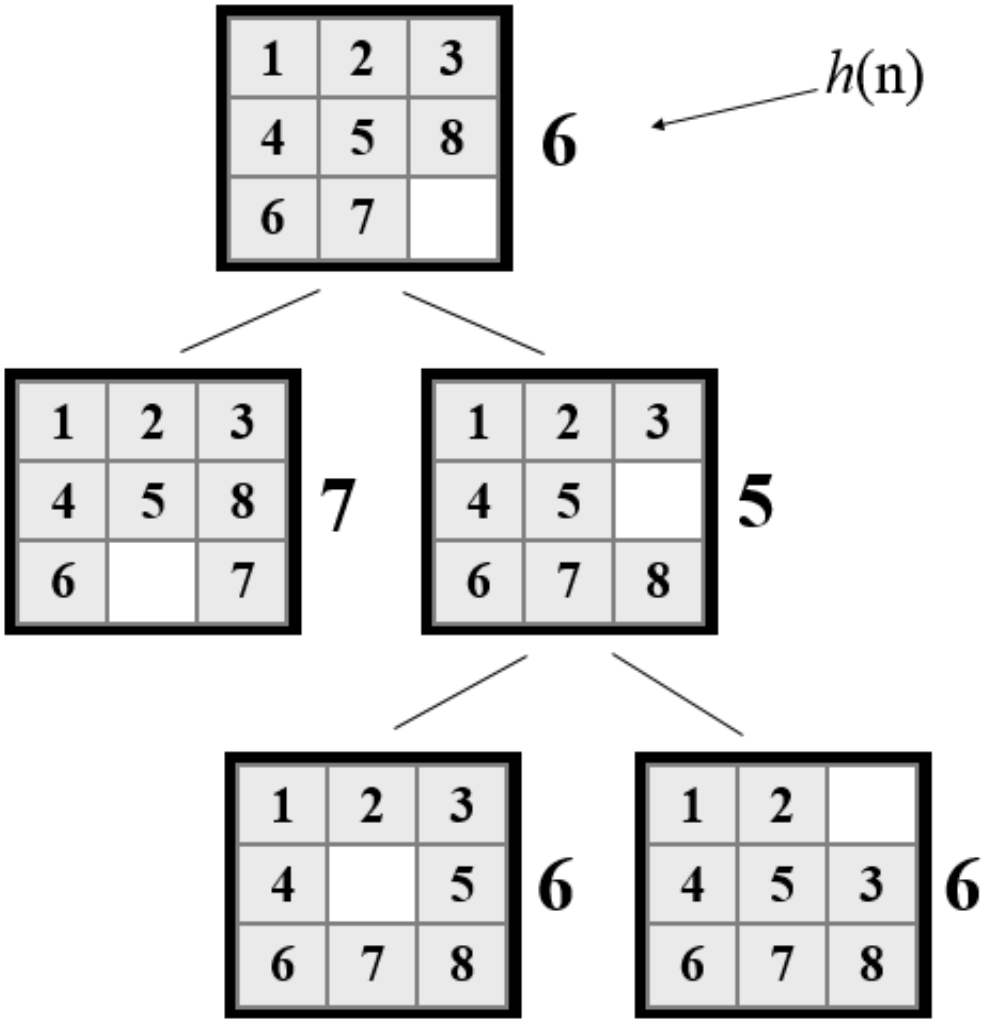
ونستطيع القول هنا أن التجريبية تفكر أنه بإمكاننا أن نصل للهدف من الخلال القيام ب 8 حركات فقط.

الحل



- يعبر الرقم بجانب كل رقعة عن قيمة تجريبية باستخدام Manhattan Distance .
- ولكن هناك مشكلة قد تواجهنا هي أن خوارزمية Hill Climbing لا تعطي دوماً حلاً أمثلياً كما في المثال التالي:

الحل



- نلاحظ في هذا المثال أن قيمة التجريبية ازدادت من مستوى لأخر بدلاً من أن تتناقص وبالتالي وقعت الخوارزمية في حالة Local Minima