كلية العلوم

القسم : **الرياضيات**

السنة : **الرابعة**

المادة : ذكاء صنعي

المحاضرة : الرابعة/نظري/

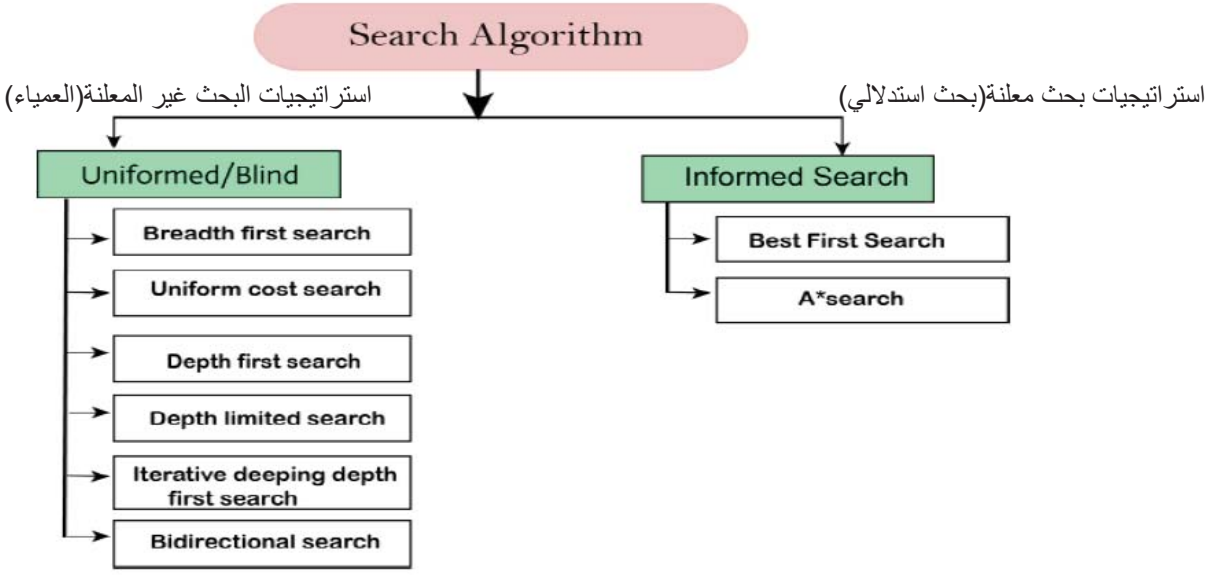
{{ مكتبة A to Z }}

Facebook Group : مكتبة A to Z

SOLVING PROBLEMS BY SEARCHING

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

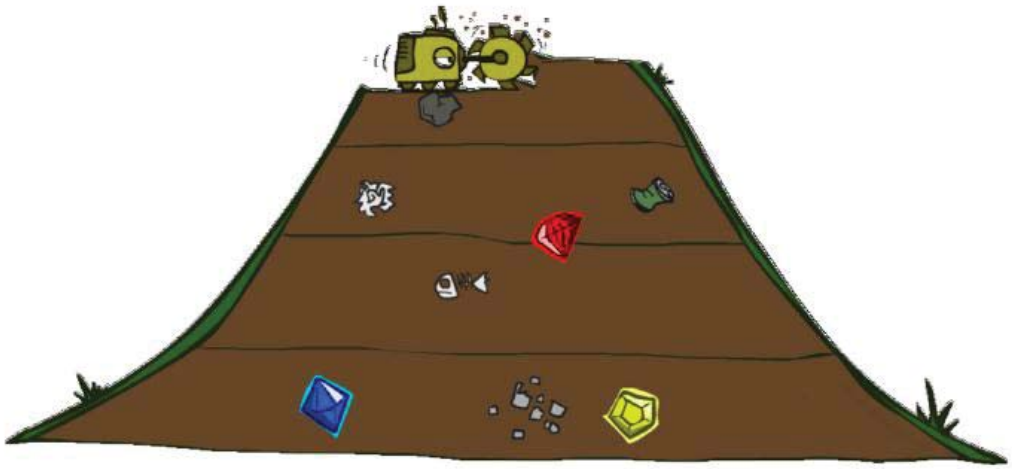
تستخدم استراتيجيات البحث غير الموجهة(العمياء) المعلومات المتاحة فقط في تعريف المشكلة

States, actions, goal test, path cost

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.

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

Breadth-First Search (BFS)



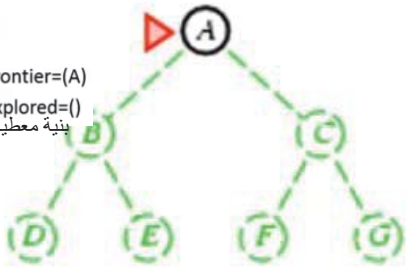
1. Breadth-first search (BFS)

- Expand shallowest unexpanded node
- Nodes are stored in FIFO queue (new successors go at end)

Queue: [A]

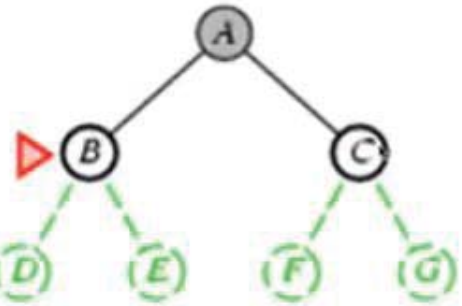
- Frontier=(A)
- Explored=()

بنية معطيات مستقلة تحوي العقد التي تم زيارتها



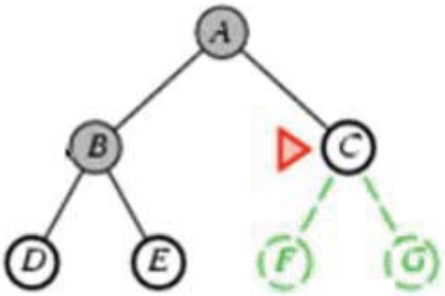
Queue: [B, C]

- Frontier=(B,C)
- Explored=(A)



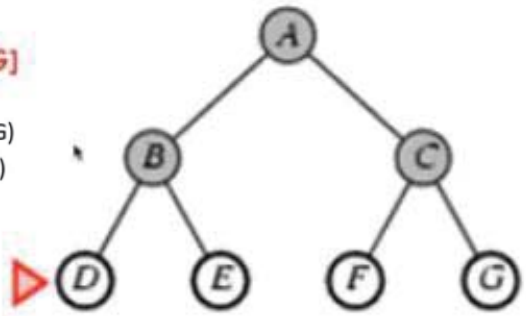
Queue: [C,D, E]

- Frontier=(C,D,E)
- Explored=(A,B)

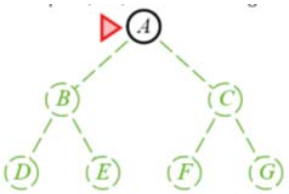


Queue: [D, E,F,G]

- Frontier=(D,E,F,G)
- Explored=(A,B,C)



نريد الوصول من العقدة A إلى العقدة E



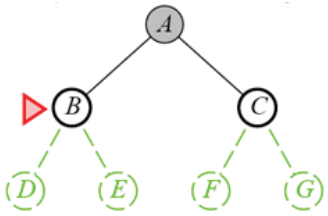
بداية نختبر العقدة الحالية A، هل هي العقدة الهدف؟؟ لا

لذلك نقوم بتوسعتها و إضافة أبنائها إلى الرتل: الأبناء B,C

سنقوم باختيار العقدة المضافة أولاً من الرتل FIFO وهي العقدة B

نختبر العقدة B هل هي الهدف ؟؟ لا ليست كذلك

سنقوم بتوسعة أبناء B وإضافة أبنائها إلى الرتل، أصبح بالرتل:

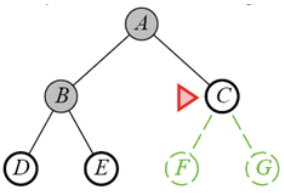


C,D,E حيث D وE هم أبناء B ، الآن من نختار من الرتل؟؟؟

سنختار العقدة الأكثر سطحيةً من الشجرة وهي C (حيث C دخلت للرتل قبل D و E لذلك يتم اختيارها)

هل العقدة C هل هي الهدف؟؟ لا ليست كذلك

أقوم بتوسعتها وإضافة أبنائها إلى الرتل، أصبح الرتل:

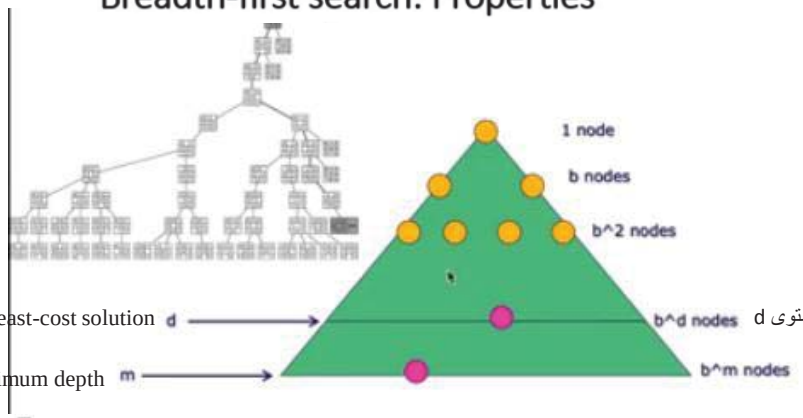


D,E,F,G الآن كل العقدة الموجودة بالرتل ذو مستوى واحد

نحن نعتمد مبدأ FIFO نختار العقدة الداخلة أولاً وهي D

هل هي العقدة الهدف؟ لا ، أقوم بتوسعتها ولكن هي عقدة ورقية لا تملك أبناء
لذلك اختار من الرتل العقدة التالية وهي E واختبرها لتكون هي الهدف

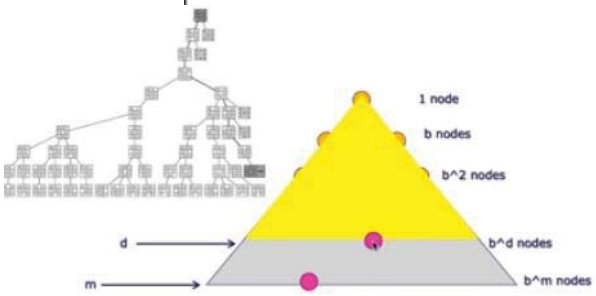
Breadth-first search: Properties



d—depth of the least-cost solution

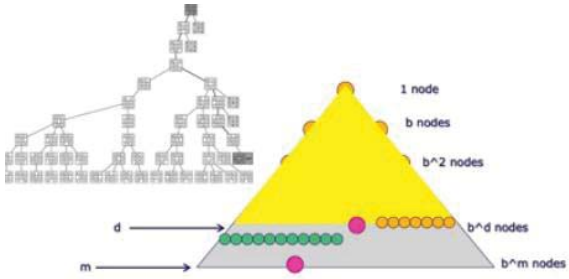
m—maximum depth

عدد nodes في المستوى d



Time? $O(b^d)$

زمن تنفيذ خوارزمية البحث هذه هي الزمن الذي استغرقه لزيارة كل nods لحين وصولها node الهدف
عدد node الموجودة ضمن المثلث



Space? $O(b^d)$

عدد nodes المخزنة في الذاكرة Queue

Properties of breadth-first search

Complete?? Yes (if b is finite)

Time?? $1 + b + b^2 + b^3 + \ldots + b^d + b(b^d - 1) = O(b^{d+1})$, i.e., exp. in d

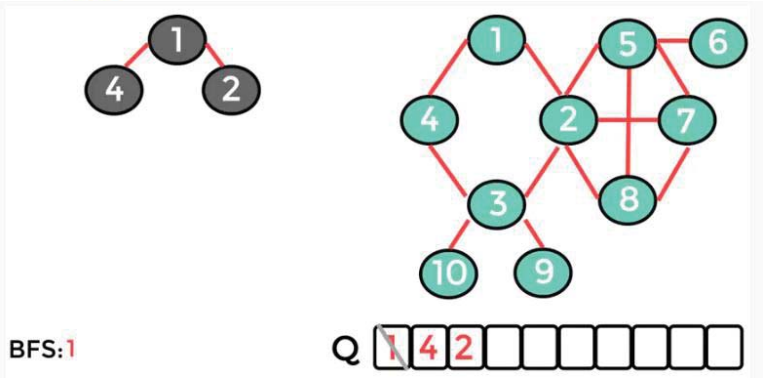
Space?? $O(b^{d+1})$ (keeps every node in memory)

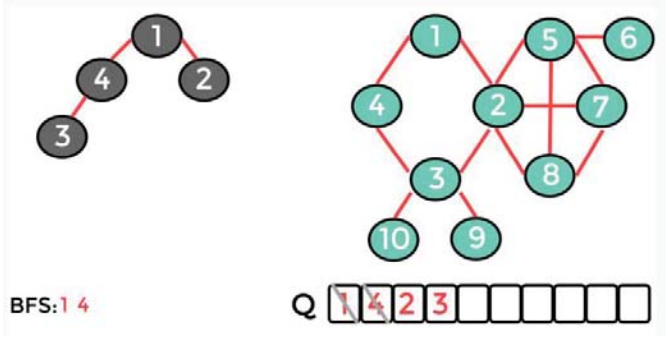
Optimal?? Yes (if cost = 1 per step); not optimal in general

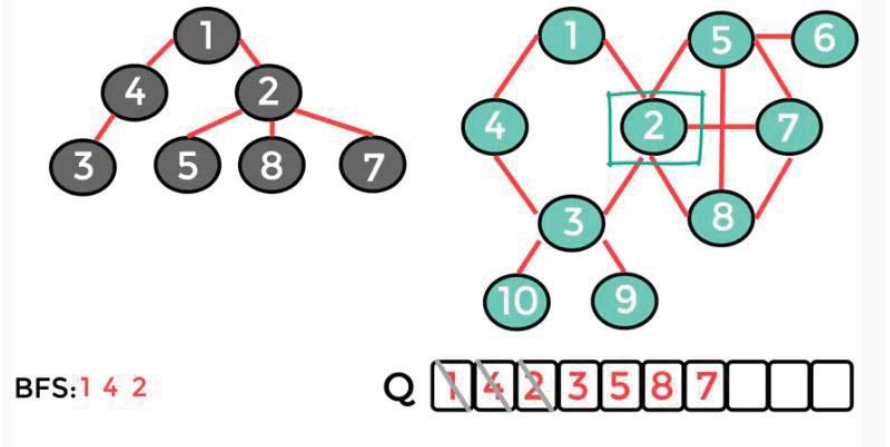
كل المسارات نفس الكلفة واحد مثلا

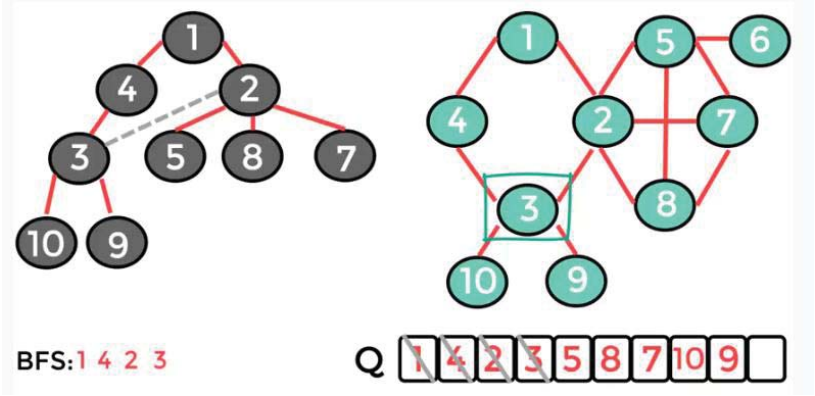
Space is the big problem; can easily generate nodes at 100MB/sec
so 24hrs = 8640GB.

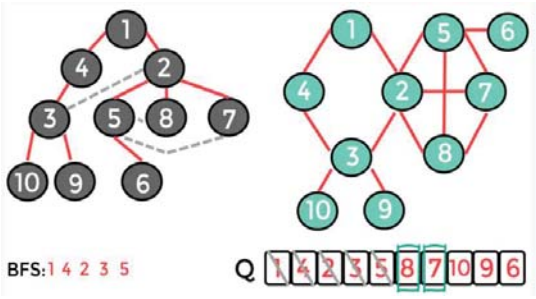
Breadth-first search (BFS)

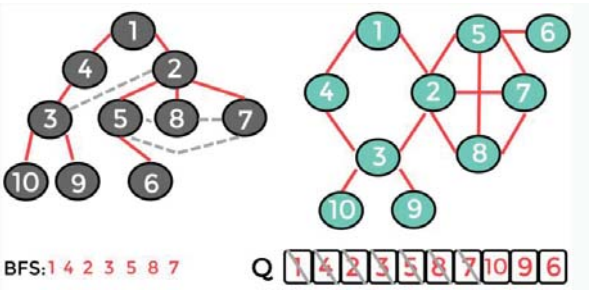


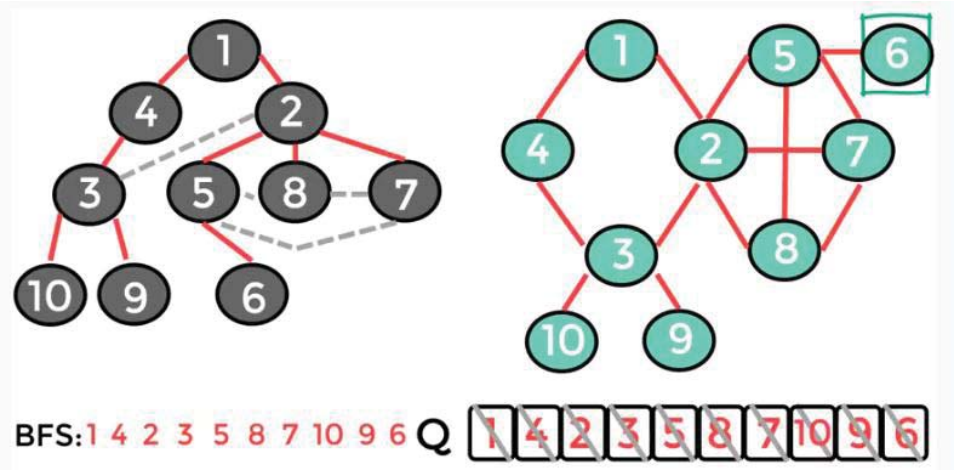


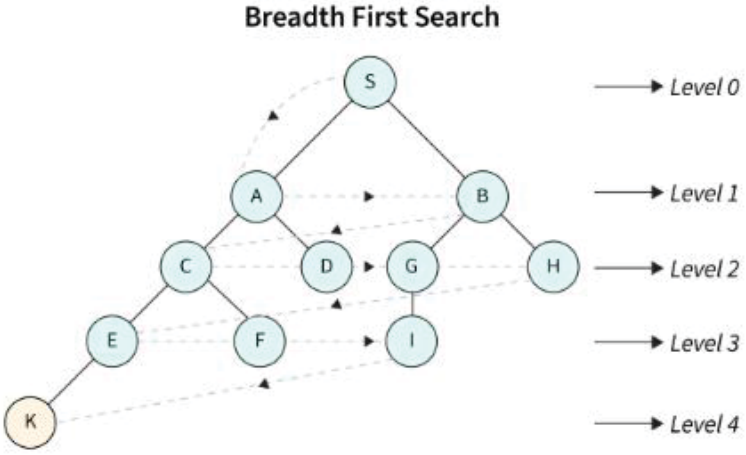




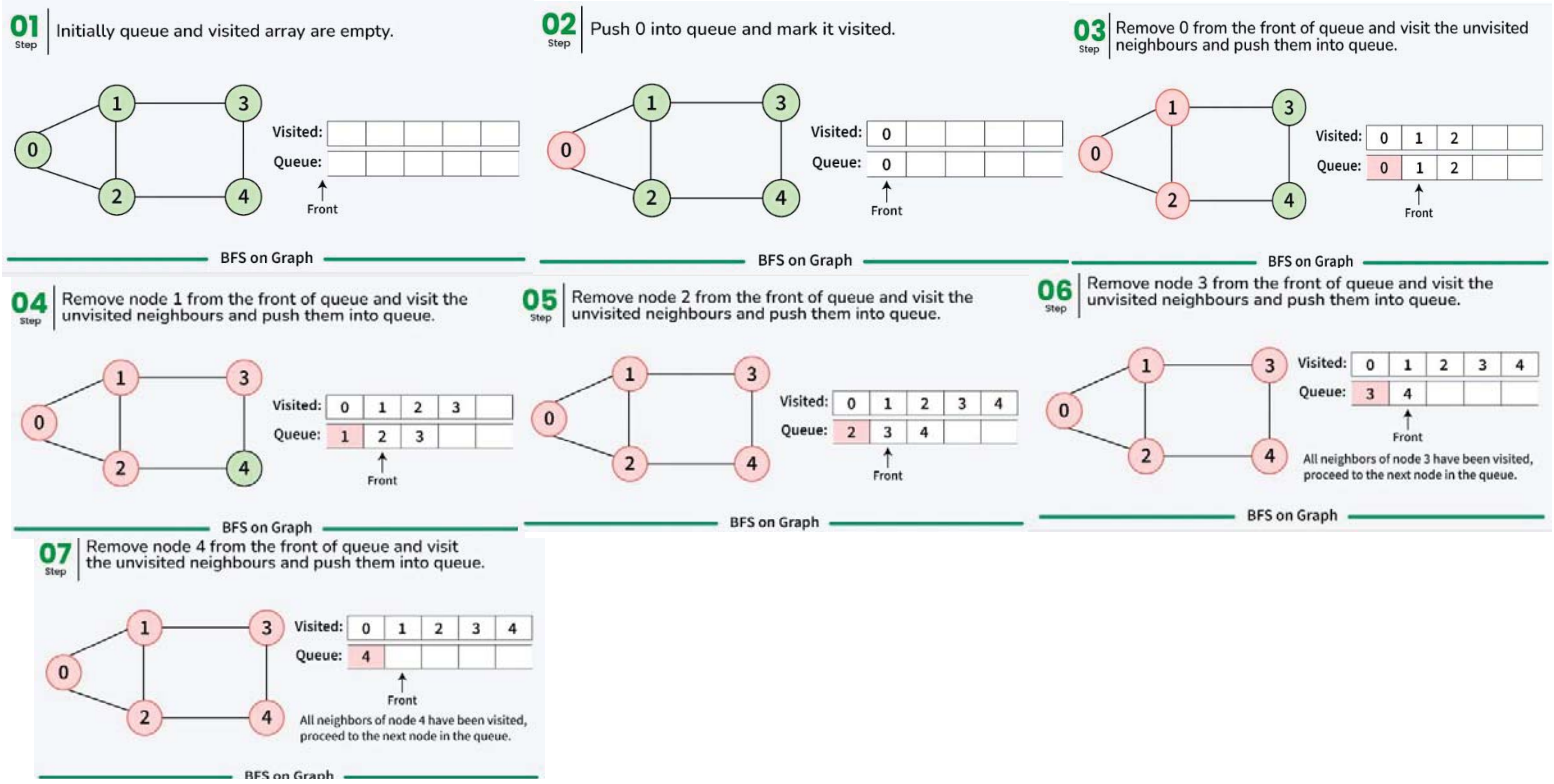

BFS: 1 4 2 3 5
Q 1 4 2 3 5 8 7 10 9 6


BFS: 1 4 2 3 5 8 7
Q 1 4 2 3 5 8 7 10 9 6

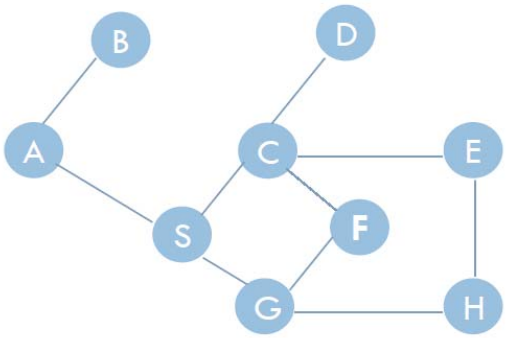

BFS: 1 4 2 3 5 8 7 10 9 6
Q 1 4 2 3 5 8 7 10 9 6

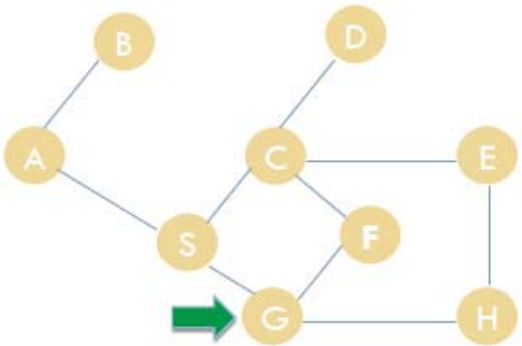

Breadth First Search
S
A
B
C
D
G
H
E
F
I
K
Level 0
Level 1
Level 2
Level 3
Level 4

https://www.geeksforgeeks.org/breadth-first-search-or-bfs-for-a-graph/



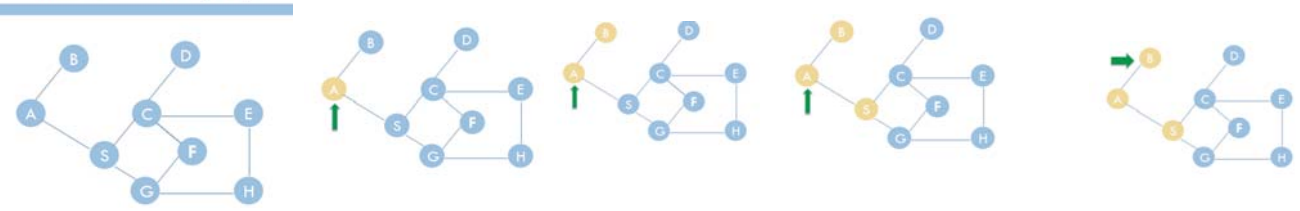
Breadth-First Search (BFS)

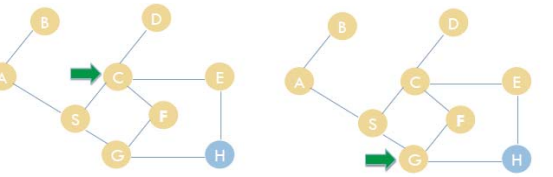


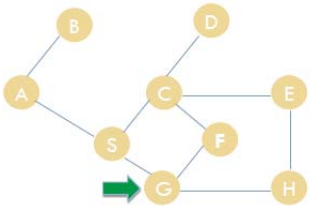


{A B S C G D E F H}

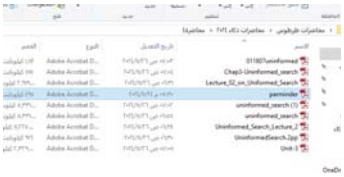
Breadth-First Search (BFS)







{A B S C G D E F H}



Uniform Cost Search (UCS)

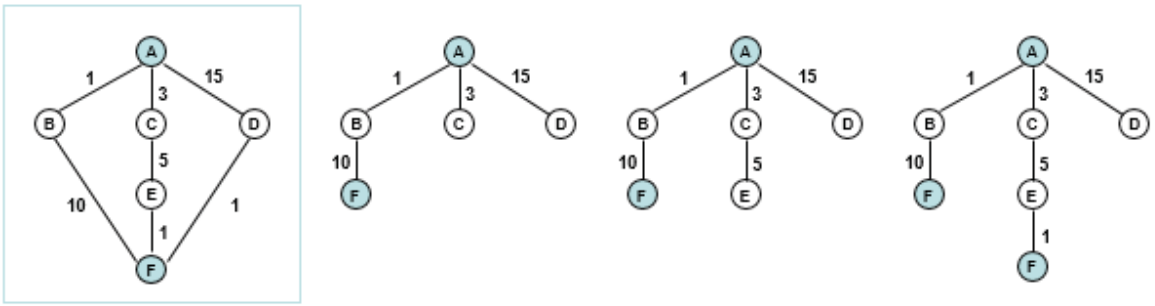
2. Uniform-cost search (UCS)

استراتيجة البحث بالكلفة المنتظمة

- Expand least-cost unexpanded node اختار node التي لها cost اقل
- Nodes are stored in Ordered queue (order by cost) First-In-First-Out (FIFO)

يتم التفريق بين الخوازميات طريقة ترتيب nod المزارة بمعنى اختيار ال nod من Queue

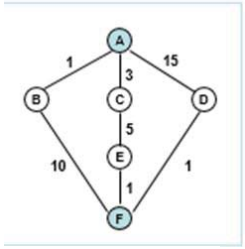
تشبه الطريقة السابقة ولكن في كل مرة تقو م بتوسعة عقدة ما وإضافة أبنائها، تعيد ترتيب الرتل من جديد حسب الكلفة order by cost بشكل تصاعدي، لتقوم باختيار العقدة الأقل تكلفة دوما، عمليا هي العقدة الأولى بالرتل المرتب Ordered queue

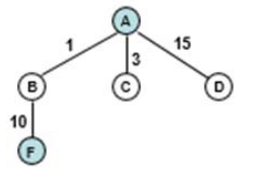


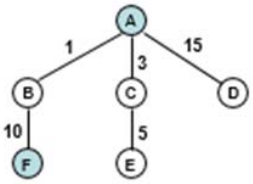
C* is cost of optimal solution
ε is minimum action cost

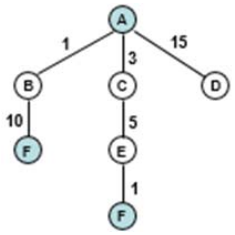
Optimal: Yes
Complete: if ε > 0

Time: $O(b^{ceiling(C^*/\varepsilon)})$
Space: $O(b^{ceiling(C^*/\varepsilon)})$









عقدة البداية هي A نريد الوصول للعقدة F

هل العقدة A هي الهدف؟؟ لا ، لذلك نقوم بتوسعتها و إضافة أبنائها للرتل

الرتل : B,C,D نرتبهم حسب الكلف يبقوا على حالهم كونهم مرتبين بالأصل

نختار من الرتل العقدة الأولى وهي الأقل تكلفة(1) وهي B

ونقوم باختيارها هل هي الهدف؟؟ لا ليست كذلك ، أقوم بتوسعتها وإضافة أبنائها، تملك ابن واحد وهو F نضيفها للرتل ونعيد ترتيب الرتل

أصبح الرتل : C,F,D نختار العقدة الأقل كلفة(3) وهي C

يسأل البعض ، في حال ضفنا العقدة الهدف لماذا لم ننتهي؟؟ العقدة المضافة لا نعرف ماهيتها إلا حين نختارها ونفحصها

نختبر العقدة C هل هي الهدف؟؟ لا ليست كذلك

نقوم بتوسعتها وإضافة أبنائها للرتل ونرتب الرتل من جديد، أصبح :

E,F,D نختار العقدة الأولى منه وهي الأقل كلفة (8) وهي العقدة E

الكلف يتم حسابها بشكل تراكمي ، من العقدة الجذر للعقدة الحالية

هل العقدة E هي الهدف ؟؟ لا لذلك نوسعها ونضيف أبنائها للرتل ونرتب الرتل:

F,F,D نلاحظ أصبح لدينا عقدتين F الأولى هي الكلفة الأقل والدها E و الثانية كلفتها أكبر والدها B ، نختار العقدة الأولى من الرتل المرتب وهي F

ونفحصها وهي العقدة الهدف بكلفة 9

سنقوم بتطبيق استراتيجية البحث بالعرض على المثال السابق ، ونلاحظ الفرق

بداية نأخذ العقدة A ليست الهدف نقوم بإضافة أبنائها للرتل :B,C,D

نختار العقدة المضافة أولاً وهي B ليست الهدف أقوم بتوسعتها وإضافة أبنائها للرتل:

C,D,F نختار العقدة المضافة أولاً بين العقد وهي العقدة C ليست الهدف لذلك أقوم بتوسعتها وإضافة أبنائها للرتل : D,F,E أقوم باختيار العقدة المضافة أولاً وهي D وليست الهدف ، أقوم بتوسعتها وإضافة أبنائها للرتل: F,E,F اختار العقدة المضافة أولاً وهي F التي والدها B وهي التي تمت إضافتها أولاً نفصحها لنكتشف أنها الهدف بكلفة 11

الفرق واضح بين الكلف بين الطريقتين

ملاحظة بطريقة البحث بالكلفة المنتظمة يوجد زمن مضاف وهو زمن ترتيب الرتل في كل عملية اختيار ويكبر هذا الزمن كلما كانت الفروع عريضة فيزداد حجم الرتل

البحث بالعرض أولاً يبحث عن الحل ذو العمق الأول، بينما البحث بالكلفة المنتظمة يبحث عن الحل بالكلفة الأقل

Breadth-first search (BFS) is a **special case** of **uniform-cost search** when all edge costs are **positive and identical**.

Breadth-first always expands the shallowest node
- Only optimal if all step-costs are equal

Uniform-cost considers the overall path cost
- Optimal for any (reasonable) cost function
  - non-zero, positive
- Gets stuck down in trees with many fruitless, short branches
  - low path cost, but no goal node

Both are complete for non-extreme problems
- Finite number of branches
- Strictly positive search function

Breadth-first Search is a special case of Uniform-cost search

Deriving BFS from UCS:

The cost function in BFS: C(x') = C(x) + 1 ...(1)

The cost function in UCS: C(x') = C(x) + d(x, x') ...(2)

C(x0) = 1 in BFS and UCS ...(3)

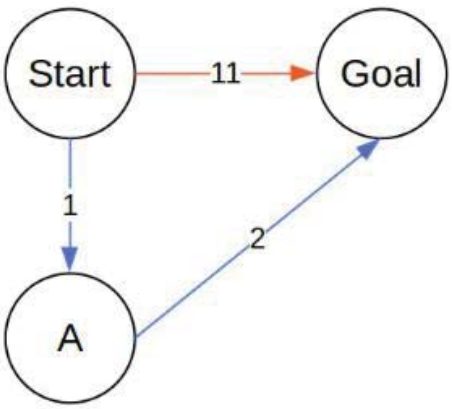
If we make transition cost from node x to x' = 1 then;

UCS: C(x') = C(x) + 1 ...(4)

Thus, from (1) and (4), we conclude that BFS is derived from UCS by making transition cost as 1.

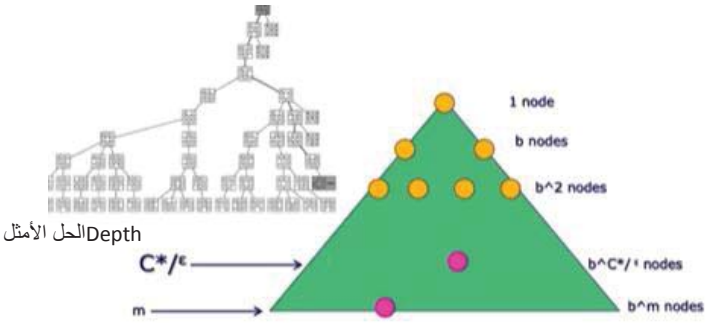
Therefore, BFS is a special case of UCS.

https://www.geeksforgeeks.org/breadth-first-search-is-a-special-case-of-uniform-cost-search/?ref=oin_asr1



Breadth-First Search returns the shortest path to the goal, even though there is a lower-cost path with more edges.

https://www.baeldung.com/cs/uniform-cost-search-vs-best-first-search



- Process all the nodes with the cost least than the cheapest solution
- Cost of optimal solution is C* and the cost of every action is ε

خصائص استراتيجية البحث بالكلفة المنتظمة:

الشمولية : تعتبر طريقة شاملة ، إذا كانت الكلفة أكبر من مقدار صغير موجب e

التعقيد الزمني : عدد العقد بالمسار الأقل كلفة ، وهو أصغر أو يساوي كلفة الحل الأمثلي

O(b^[C*/e]) حيث C* هي كلفة الحل الأمثلي.

تعقيد الذاكرة : O(b^[C*/e])

الأمثلية : نعم لأنها تعثر على الحل بكلفة أصغرية

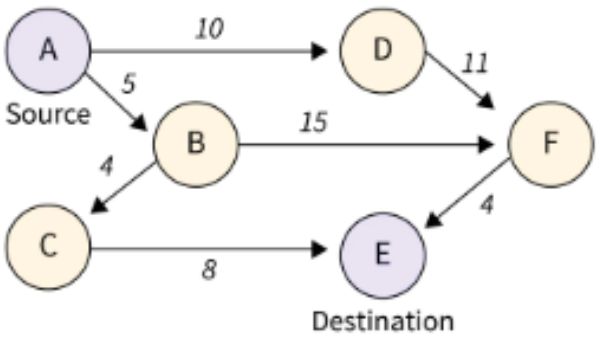
لاحظ أن هذه الطريقة تكافئ طريقة البحث بالعرض في حال كانت كلف التنقل بين كل العقد متساوية

C* is cost of optimal solution
ε is minimum action cost

Optimal: Yes
Complete: if $\varepsilon > 0$

Time: $O(b^{ceiling(C^*/\varepsilon)})$
Space: $O(b^{ceiling(C^*/\varepsilon)})$

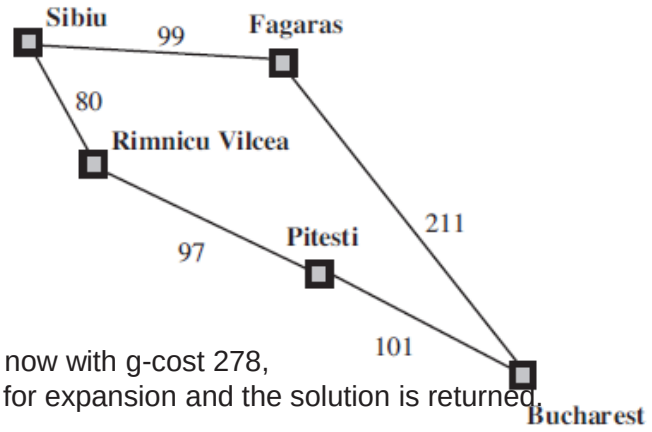
Uniform-cost search example



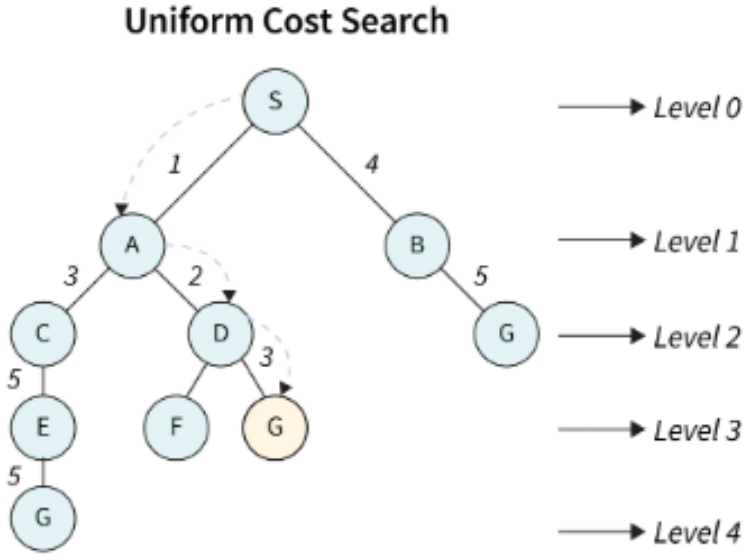
Output: 17

https://www.scaler.com/topics/uniform-cost-search/

Uniform-cost search example

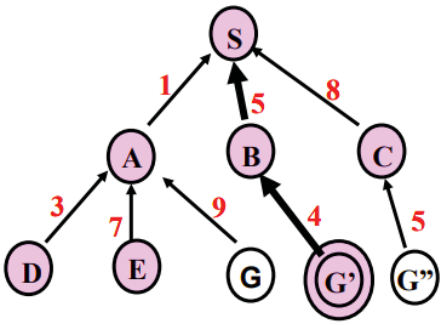


Bucharest, now with g-cost 278,
is selected for expansion and the solution is returned.



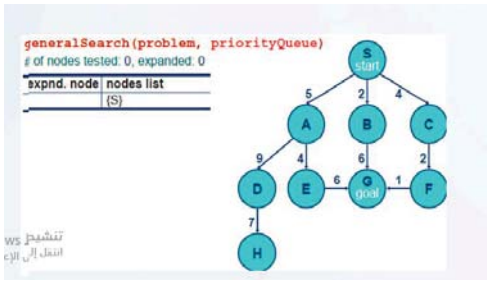
exp. node nodes list CLOSED list

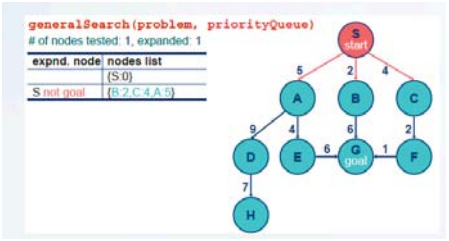
{S(0)}
S {A(1) B(5) C(8)}
A {D(4) B(5) C(8) E(8) G(10)}
D {B(5) C(8) E(8) G(10)}
B {C(8) E(8) G'(9) G(10)}
C {E(8) G'(9) G(10) G"(13)}
E {G'(9) G(10) G"(13) }
G' {G(10) G"(13) }

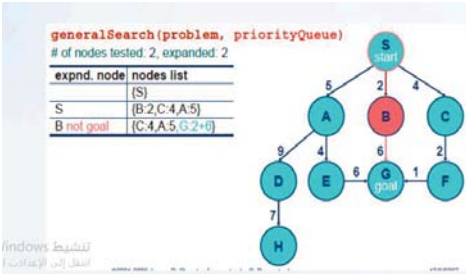


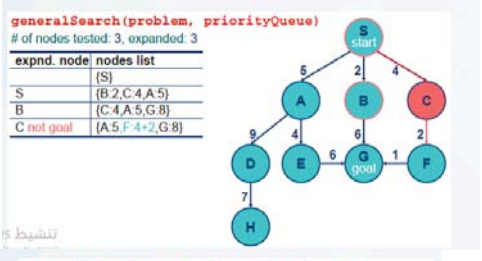
Solution path found is S B G <-- this G has cost 9, not 10
Number of nodes expanded (including goal node) = 7

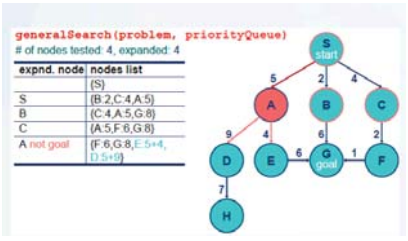
https://pg.its.edu.in/sites/default/files/AI%20Unit%202.pdf

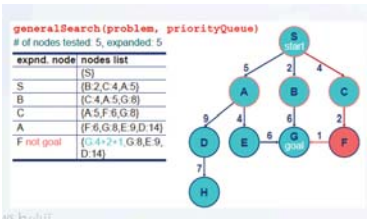


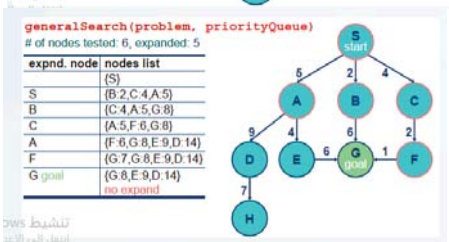


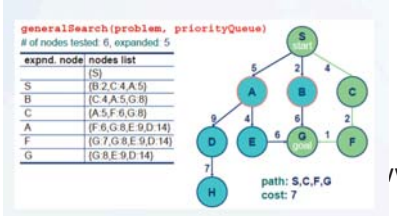






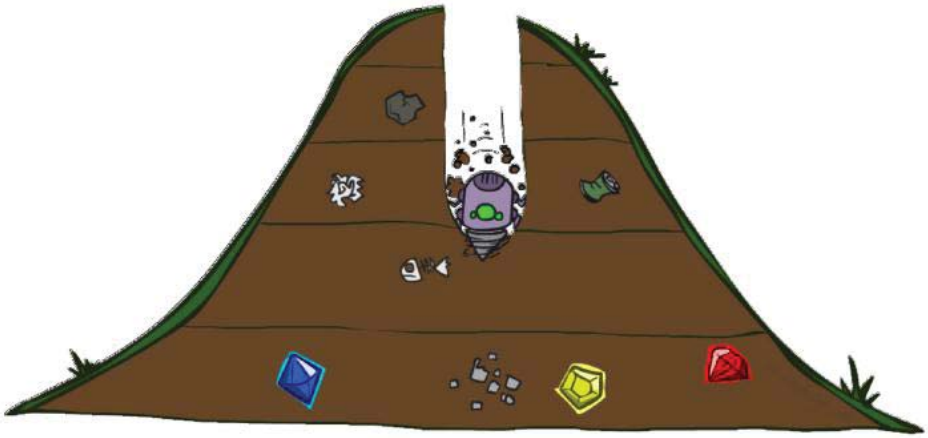






w.jarrar.info/courses/AI/Jarrar.LectureNotes.Ch3.Uninformedsearch.pdf

Depth-First Search (DFS)



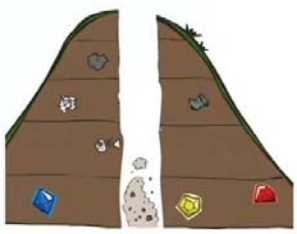
البحث بالعمق أولاً:

تقوم هذه الطريقة بتوسعة العقدة ذو العمق الأكبر، حيث تعتمد على مبدأ LIFO في اختيار العقد من الرتل، الداخل بالأخير يخرج أولاً

3. Depth-first search (DFS)

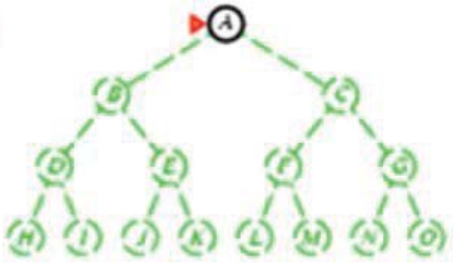
- Expand deepest unexpanded node
- Nodes are stored in LIFO stack(put successors at front)

Stack: [A]

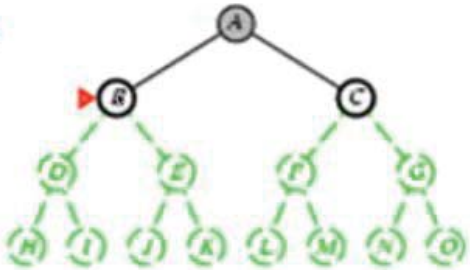


Depth-First Search (DFS)

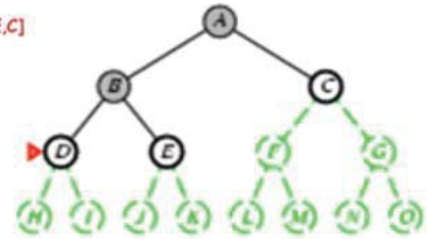
Stack: [A]



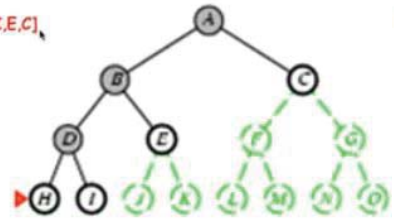
Stack: [B,C]



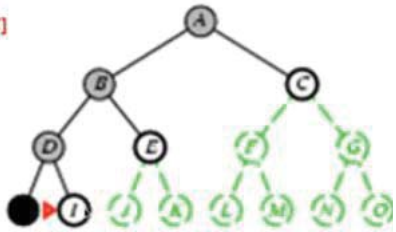
Stack: [D,E,C]



Stack: [H,I,E,C]

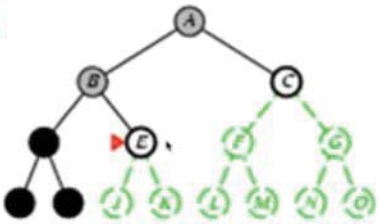


Stack: [I,E,C]

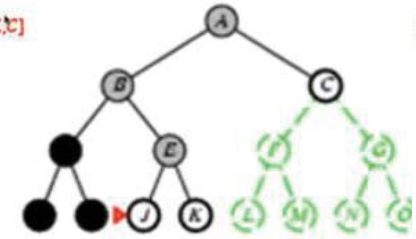


Depth-First Search (DFS)

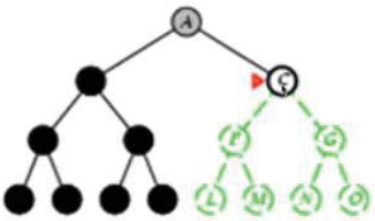
Stack: [E,C]



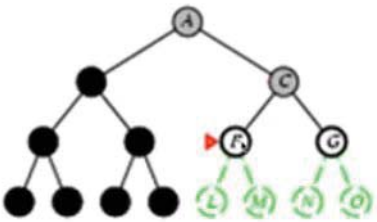
Stack: [J,K,C]



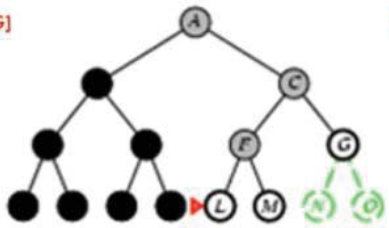
Stack: [C]



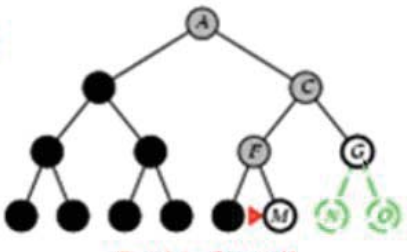
Stack: [F,G]



Stack: [L,M,G]



Stack: [M,G]



Goal is found!

3. Depth-first search(DFS)

- Complete? No (fails in infinite-depth spaces, spaces with loops)
- Time? $O(b^m)$
- Space? $O(b.m)$
- Optimal? No

خصائص استراتيجية العمق أولاً:

الشمولية: في حال كان فضاء الحل منتهي تعتبر شاملة ، في حال كان الفضاء غير منتهي ، لا تعتبر شاملة لأن العمق غير منتهي.

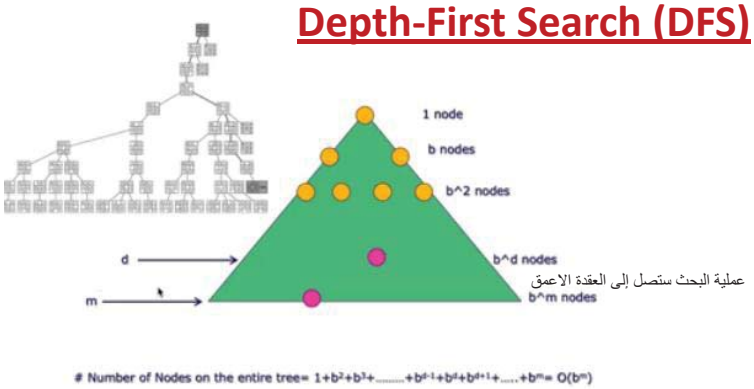
التعقيد الزمني : نحن نستمر بالتوسعة على كامل فروع الشجرة، تذكر المقياس b والذي يعبر عن عدد الأعظمي لفروع الشجرة ، لذلك التعقيد كبير جداً O(b^m)

في حال كان الحل بأقصى اليسار تعتبر هذه الطريقة سريعة جداً

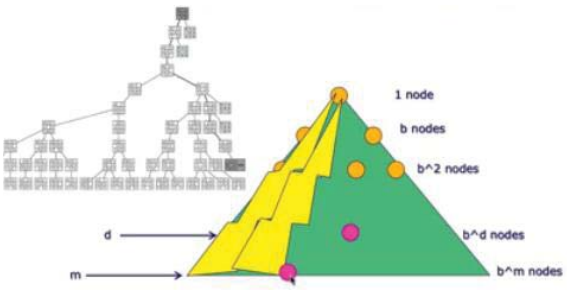
الذاكرة : O(bm) حيث من أجل كل مستوي قمنا بفصحه بشكل كامل ولم نعثر على حل نقوم بحذفه لذلك تعتبر هذه الطريقة جيدة بالتعامل مع الذاكرة.

الأمثلية : لا تعتبر مثالية.

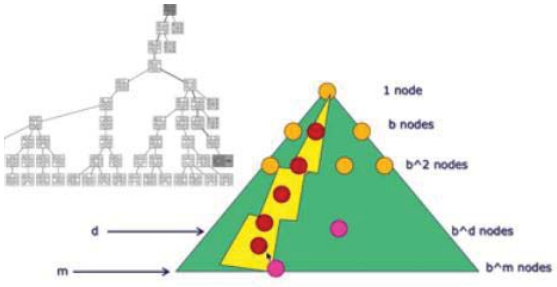
Depth-First Search (DFS)



Number of Nodes on the entire tree= $1+b^2+b^3+\ldots\ldots+b^{d-1}+b^d+b^{d+1}+\ldots\ldots+b^m = O(b^m)$

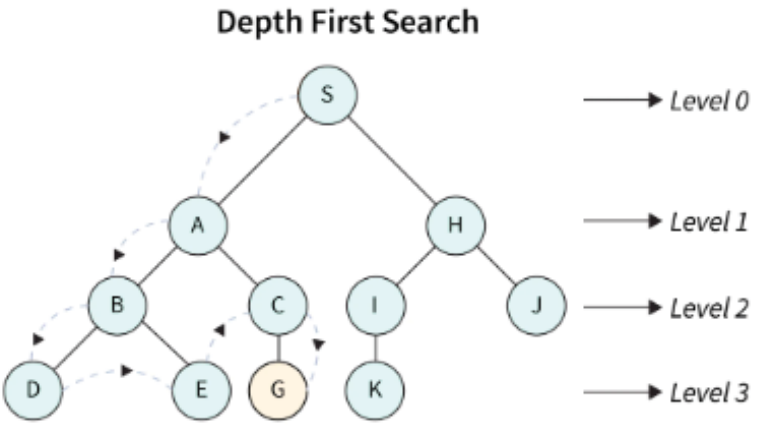


Time? $O(b^m)$

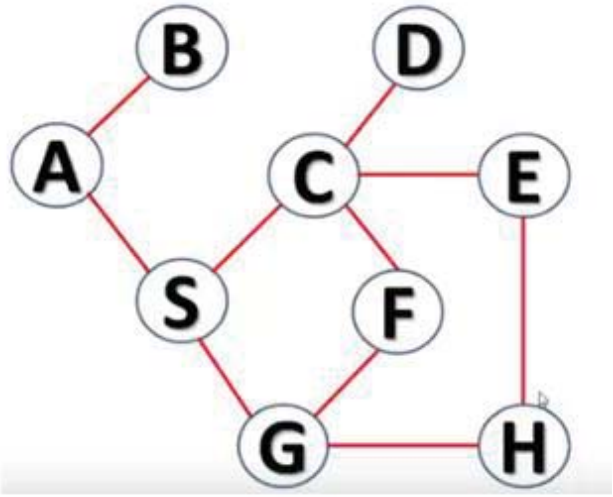


Space? $O(b*m)$

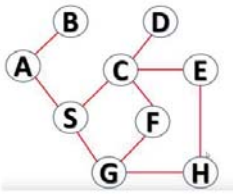
يخزن فرع واحد في tree وبالتالي عدد node :
في المستوى الاول 1 وفي المستوى الثاني b والمستوى الثالث b وهكذا
1+b+b+...+b=m*b
عدد nodes m*b

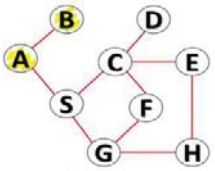


Depth-First Search (DFS)

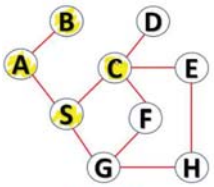


DEPTH FIRST SEARCH

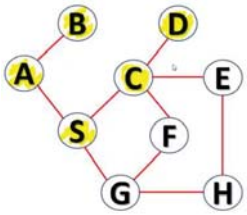




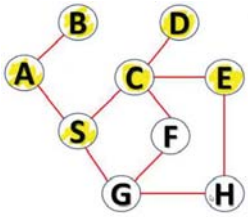
OUTPUT : A B



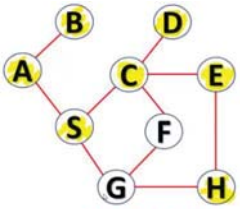
OUTPUT : A B S C



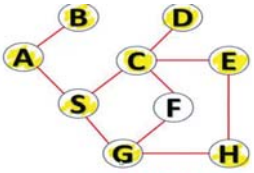
OUTPUT : A B S C D



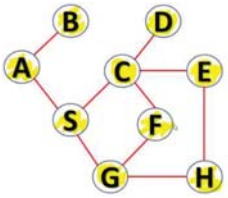
OUTPUT : A B S C D E



OUTPUT : A B S C D E H

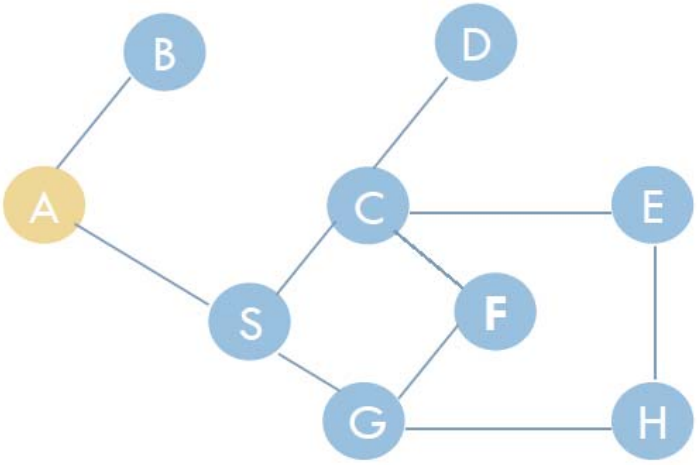


OUTPUT : A B S C D E H G

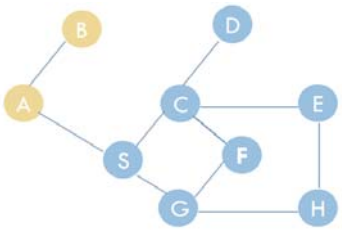
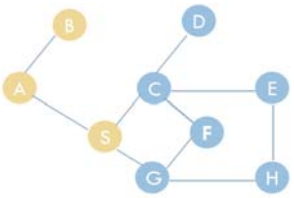
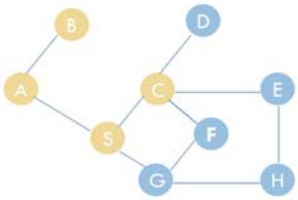
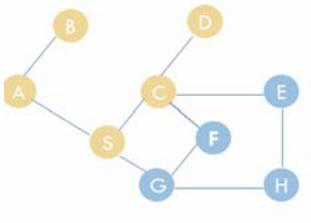
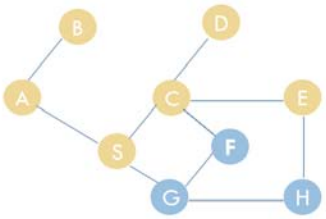
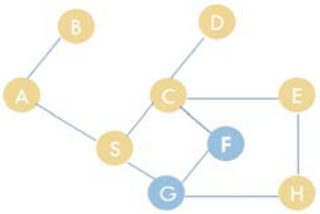
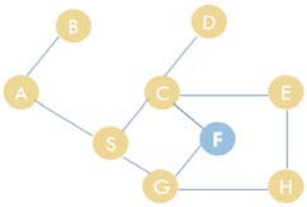
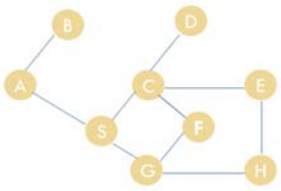


OUTPUT : A B S C D E H G F

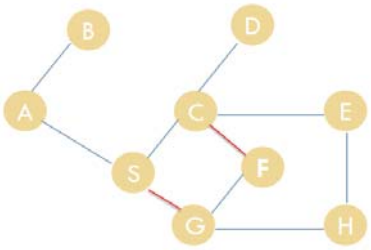
Depth-First Search (DFS)



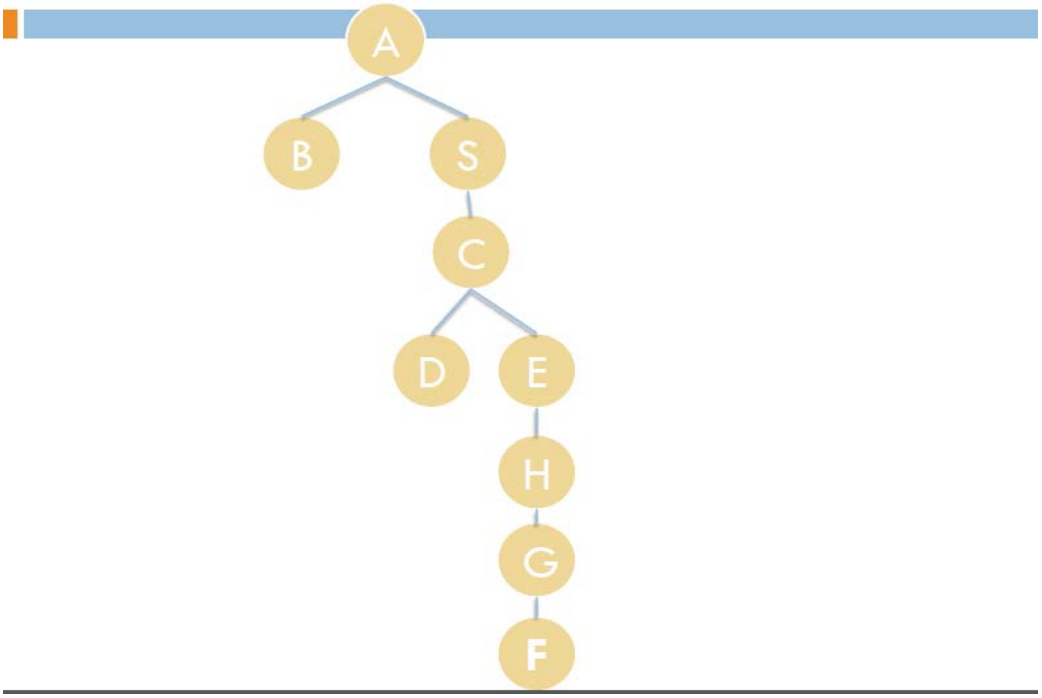
Depth-First Search (DFS)

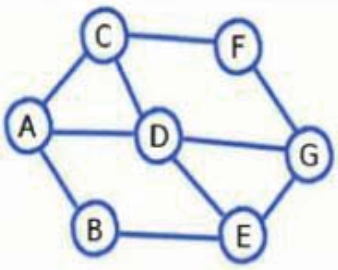
{A B S C D E H G F}



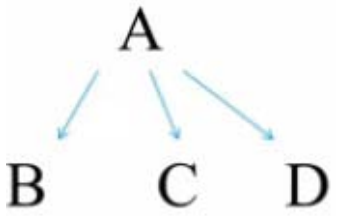
Tree after DFS run and edges in G

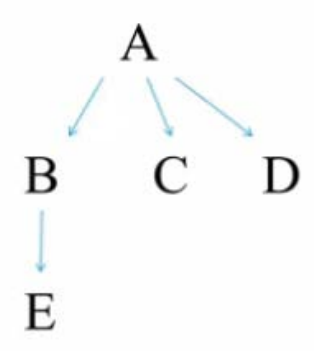


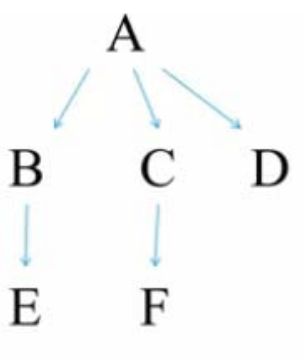
Breadth-First Search (BFS)

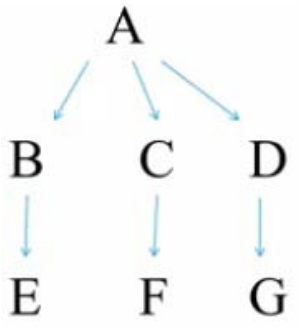


A

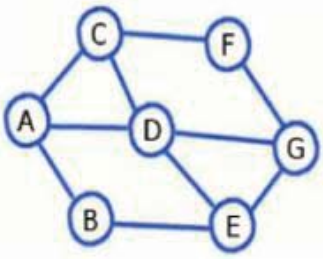




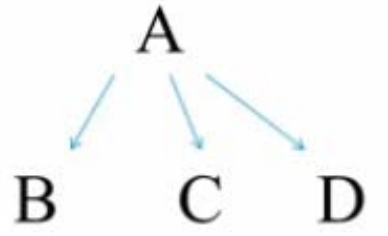


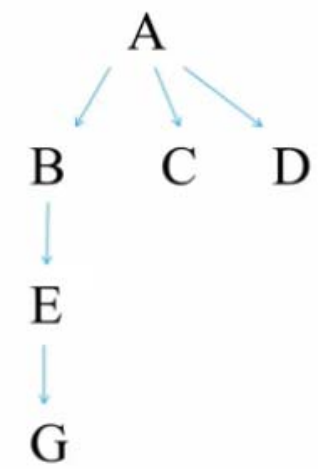


Depth-First Search (DFS)

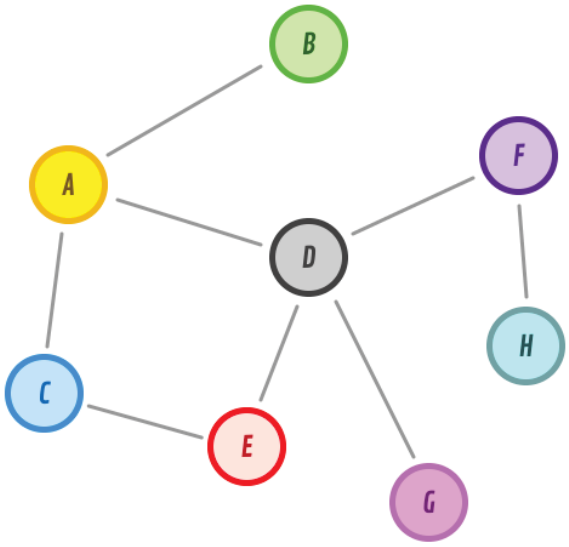


A

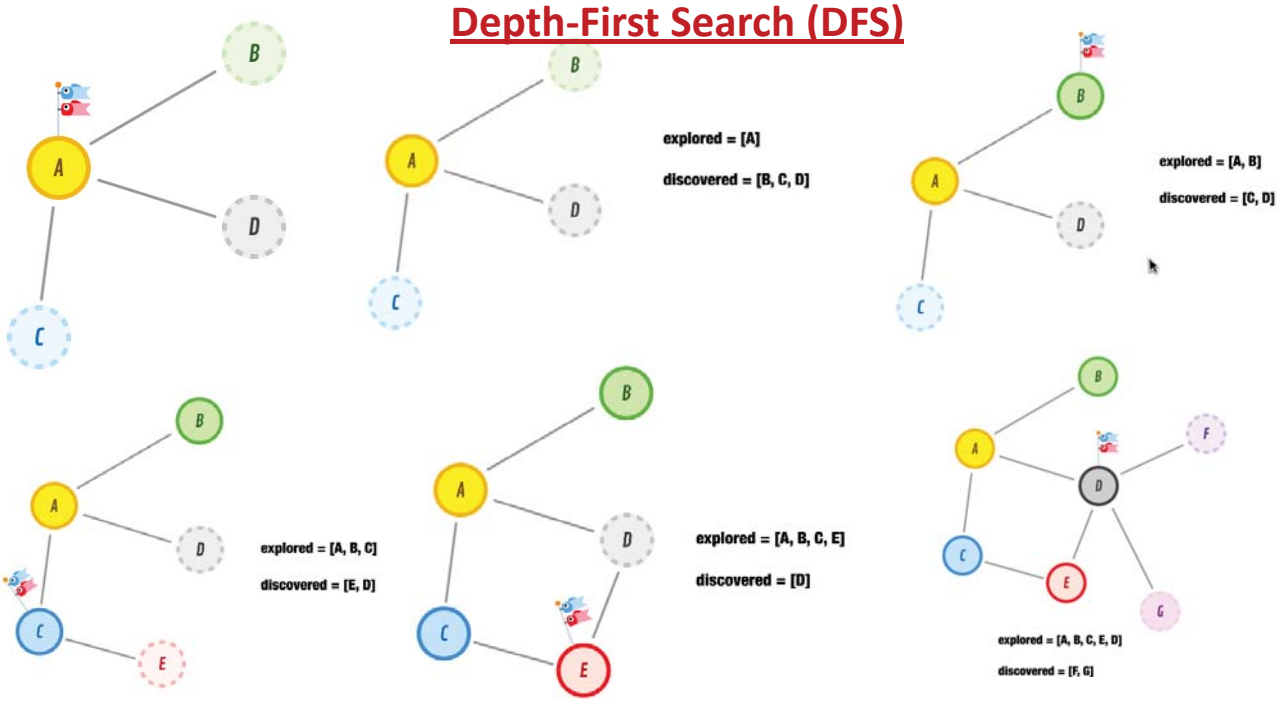




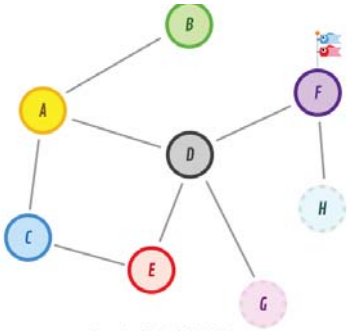
Depth-First Search (DFS)



Depth-First Search (DFS)

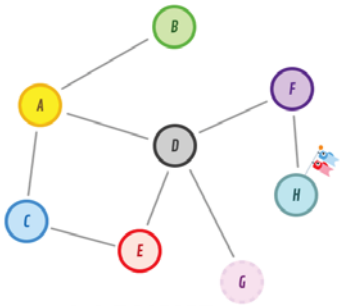


Depth-First Search (DFS)



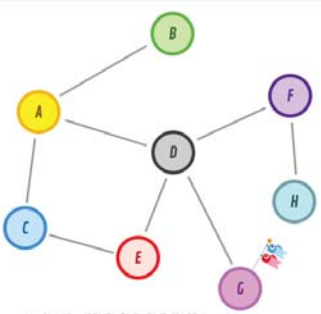
explored = [A, B, C, E, D, F]

discovered = [H, G]



explored = [A, B, C, E, D, F, H]

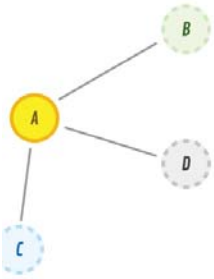
discovered = [G]



explored = [A, B, C, E, D, F, H, G]

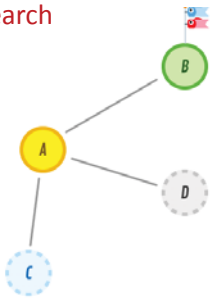
discovered = [ ]

Breadth-first search



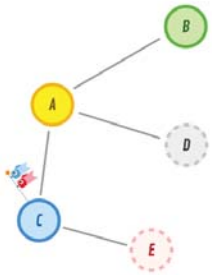
explored = [A]

discovered = [B, C, D]



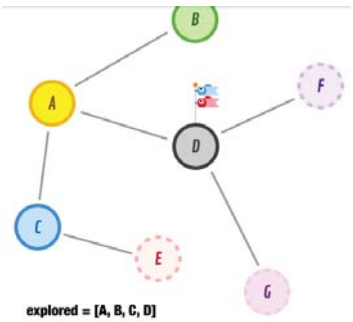
explored = [A, B]

discovered = [C, D]



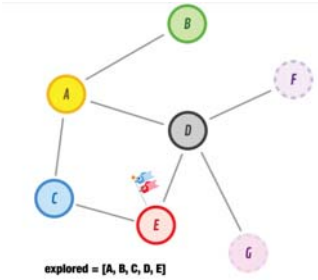
explored = [A, B, C]

discovered = [D, E]



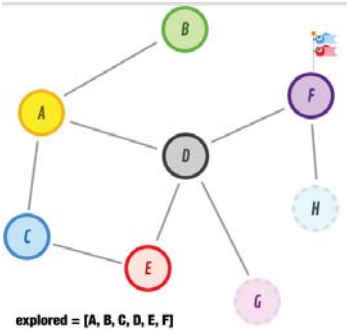
explored = [A, B, C, D]

discovered = [E, F, G]



explored = [A, B, C, D, E]

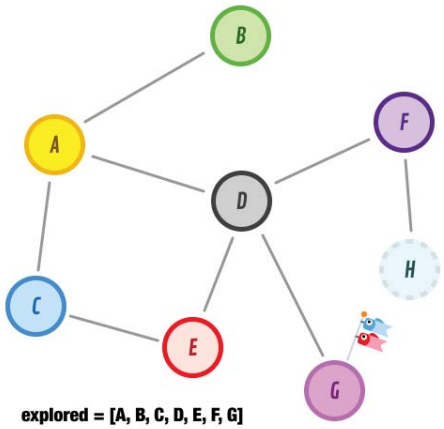
discovered = [F, G]



explored = [A, B, C, D, E, F]

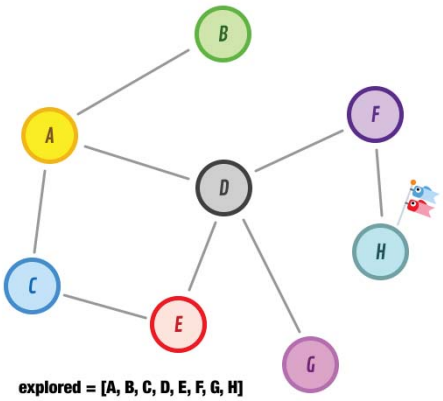
discovered = [G, H]

Breadth-first search



explored = [A, B, C, D, E, F, G]

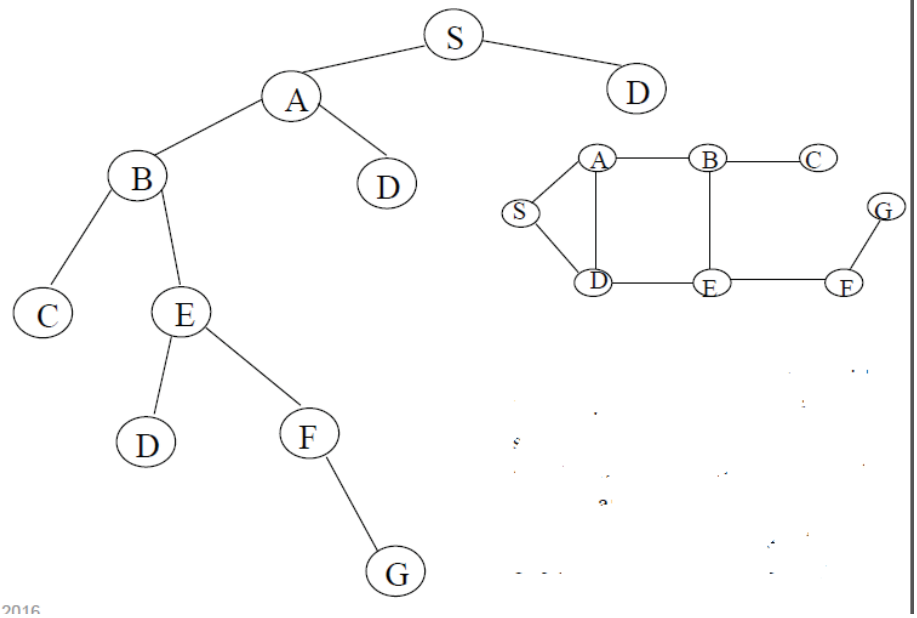
discovered = [H]



explored = [A, B, C, D, E, F, G, H]

discovered = [ ]

Depth-First Search (DFS)



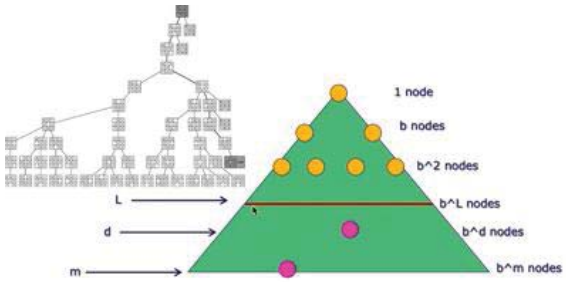
4. Depth-limit search (DLS)

- Expand deepest unexpanded node until reach limit L
- Equivalent to depth-first search with depth limit L

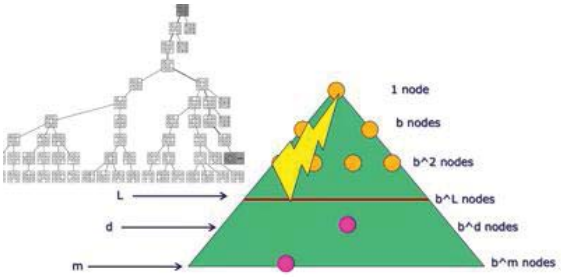
❖ **Ex**: Let L=1

Goal is not found !

البحث المحدود بالعمق هو نسخة معدلة من البحث بالعمق أولا والتي تفرض حدًا على عمق البحث.
وهذا يعني أن الخوارزمية ستستكشف فقط العقد حتى عمق معين، مما يمنعها فعليًا من النزول إلى مسارات عميقة للغاية من غير المرجح أن تؤدي إلى الهدف.



Number of Nodes on the entire tree= $1+b^2+b^3+.....+b^{d-1}+b^d+b^{d+1}+.....+b^m = O(b^m)$



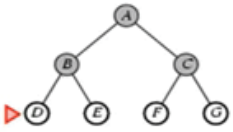
4. Depth-limit search(DLS)

❖ Complete? No (if d > L) d: goal depth, L: depth Limit value

❖ Time? $O(b^l)$

❖ Space? $O(b.l)$

❖ Optimal? No



Summary of Uninformed Tree Search Strategies

Criterion	Breadth-First	Uniform-Cost	Depth-First	Depth-Limited	Iterative Deepening
Complete?	Yes	Yes	No	No	Yes
Time	$O(b^{d+1})$	$O(b^{\lceil C^*/\epsilon \rceil})$	$O(b^m)$	$O(b^l)$	$O(b^d)$
Space	$O(b^{d+1})$	$O(b^{\lceil C^*/\epsilon \rceil})$	$O(bm)$	$O(bl)$	$O(bd)$
Optimal?	Yes	Yes	No	No	Yes

Informed Search

The **informed search** strategies uses three main algorithms. These are:

1- Best First Search Algorithm (Greedy search)

2-A* Search Algorithm الفكرة في هذه الطريقة أنها تستخدم تابع تقييم خاص بها من أجل كل عقدة

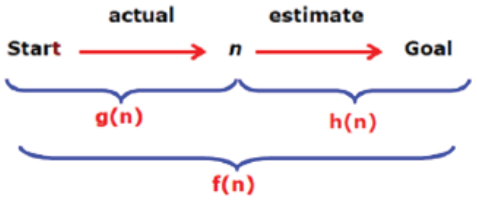
3-Hill Climbing

Heuristic Function

- h(n) = estimated cost of the cheapest path from node n to a goal node
- h(goal node) = 0
- Contains additional knowledge of the problem

التكلفة المقدرة لأرخص مسار من العقدة nإلى عقدة الهدف

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



g(n) : الكلفة الحقيقية للوصول من عقدة البداية إلى العقدة n

H(n): هو تخمين الكلفة من العقدة n إلى العقدة الهدف

Informed Search

Greedy Best first search

"Always chooses the successor node with the best *f* value"
where *f(n) = h(n)*

We choose the one that is nearest to the final state among all possible choices

"يختار دائمًا العقدة اللاحقة ذات أفضل قيمة f" حيث f(n) = h(n)
نختار العقدة الأقرب إلى الحالة الهدف بين جميع الخيارات الممكنة

البحث الأفضل أولا

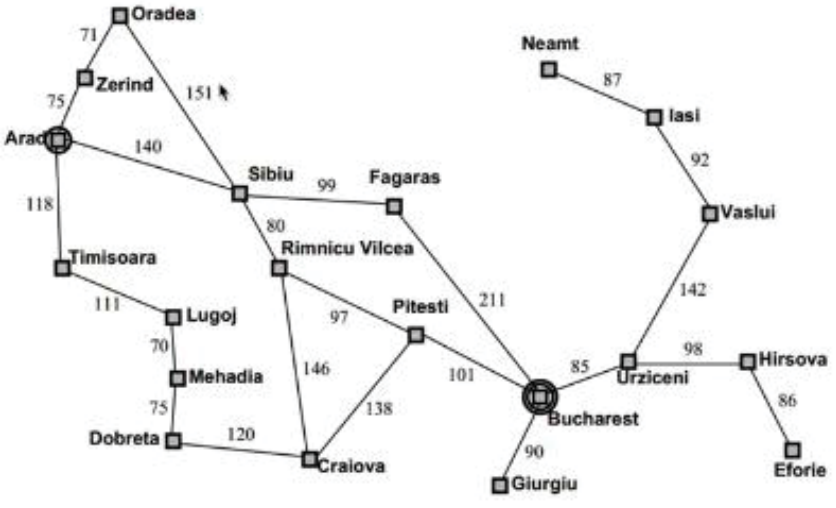
- تم اختيار العقدة للتوسيع بناءً على تابع التقييم f(n) أي قم بتوسيع العقدة التي تبدو الأفضل
- تم تحديد العقدة ذات التقييم الأدنى للتوسيع
- يستخدم priority queue
- سنتحدث عن **Greedy Best-First Search** و **A* Search**

1. Greedy search

- Evaluation function f(n) = h(n)
- h(n) is the heuristic function
- Greedy best-first search expands the node that appears to be closest to goal

choose node with minimum f(n)

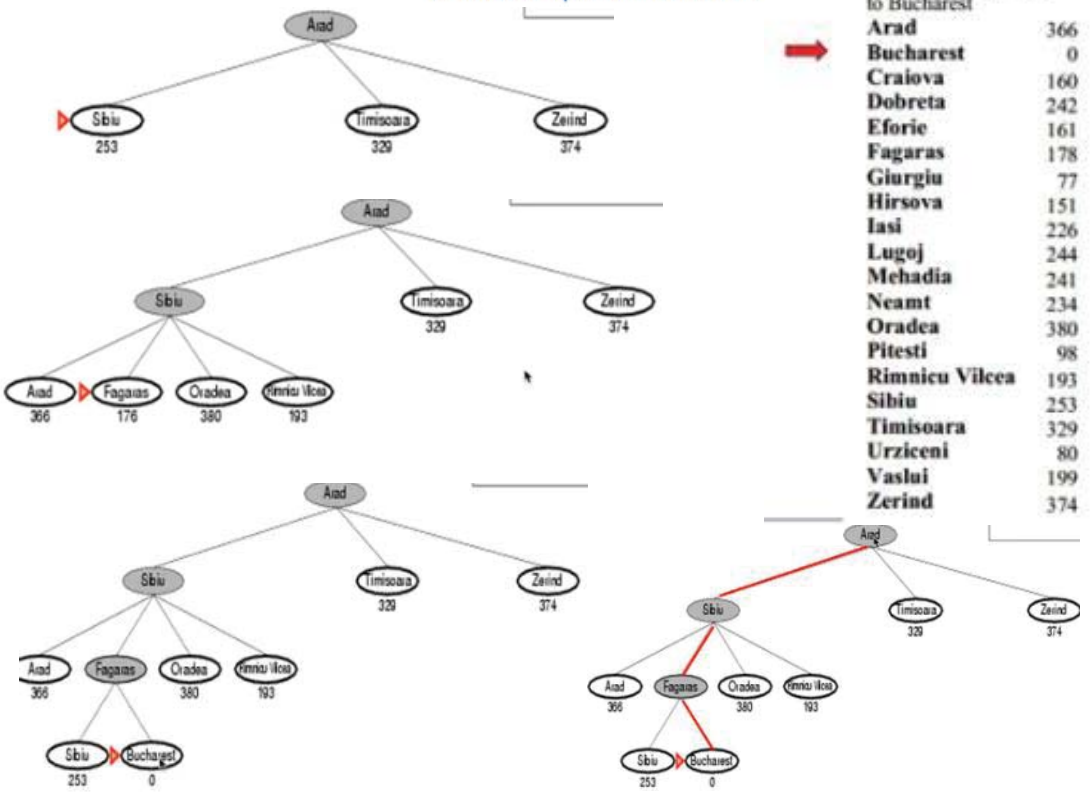
- h(n) = straight line distance (SLD) from node to goal



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

Total coast=140+99+211= 450

Is this the optimum solution ?

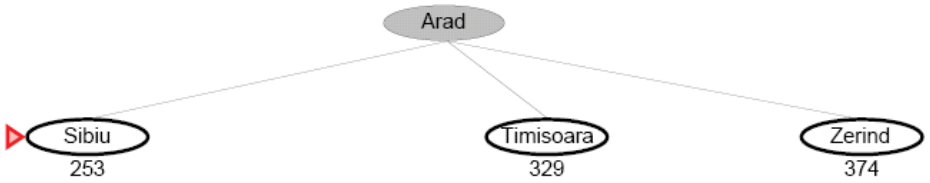


Straight-line distance to Bucharest

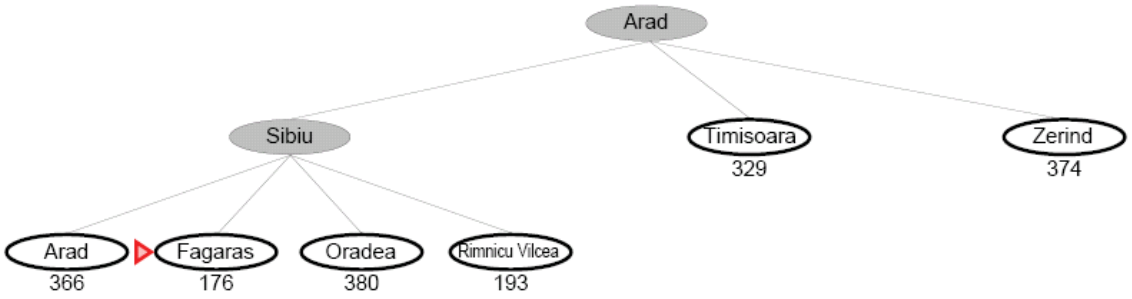
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Greedy search example

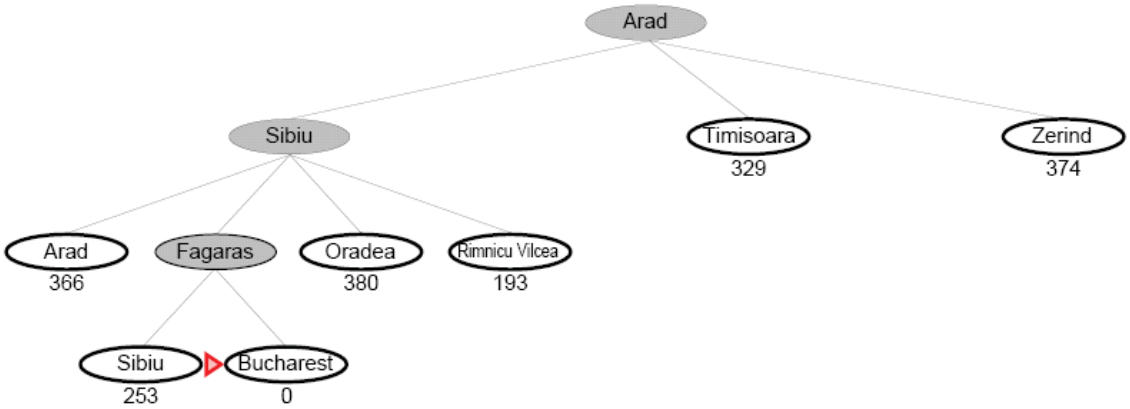
Greedy search example



Greedy search example



Greedy search example



Evaluating Greedy Best-First Search

Complete?	No (could start down an infinite path)
Optimal?	No
Time Complexity	$O(b^m)$
Space Complexity	$O(b^m)$

2. A* search

- Avoid expanding paths that are already expensive
- Evaluation function f(n) = g(n)+h(n)

  - g(n) = cost so far to reach n (actual)
  - h(n) = expected cost from n to goal (estimated)

طريقة بحث A* :

يتم حل المشاكل الموجود بالطريقة السابقة ، حيث يتفادي توسيع مسارات ذات الكلفة العالية

تستخدم هذه الطريقة التابع التالي :F(n) = g(n) + h(n) حيث :

g(n) : الكلفة الحقيقية للوصول من عقدة البداية إلى العقدة n

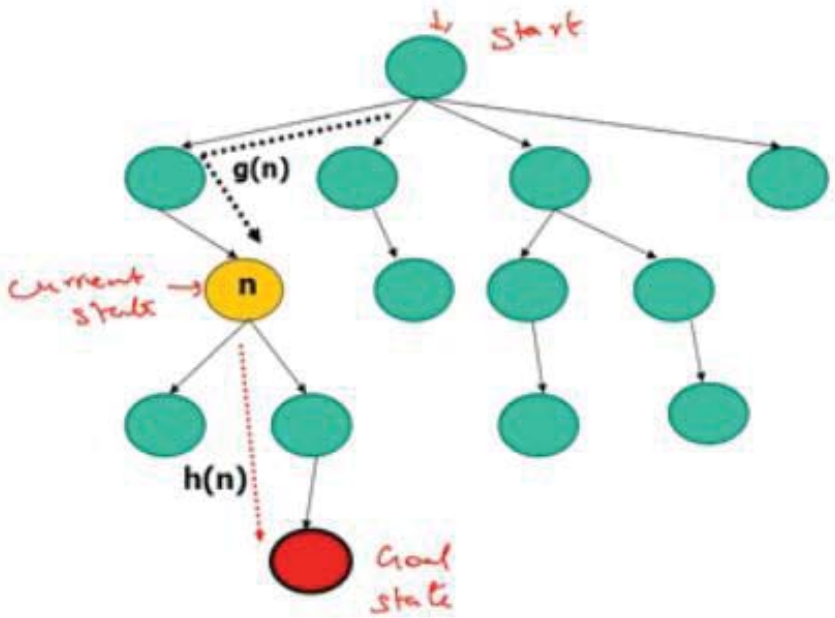
H(n): هو تخمين الكلفة من العقدة n إلى العقدة الهدف

وبالتالي F(n): هي الكلفة الكلية المتوقعة من عقدة البداية إلى عقدة الهدف مرورا بالعقدة n

طريقة A* تستخدم تخمين مقبول كمثال على ذلك:

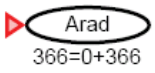
h(n) أصغر أو تساوي h*(n) حيث تعبر h*(n) عن الكلفة الحقيقية للعقدة n عن الهدف

يجب أن يكون h(n)>= 0 وكذلك يجب أن يكون h(G) = 0 من أجل أي هدف نسعى إليه

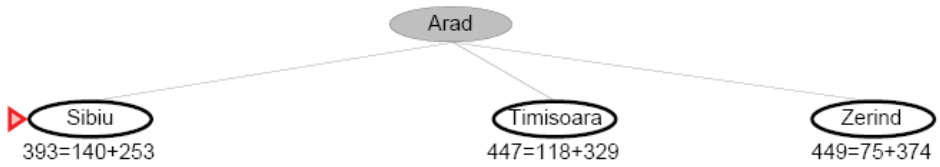


$$f(n) = g(n)+h(n)$$

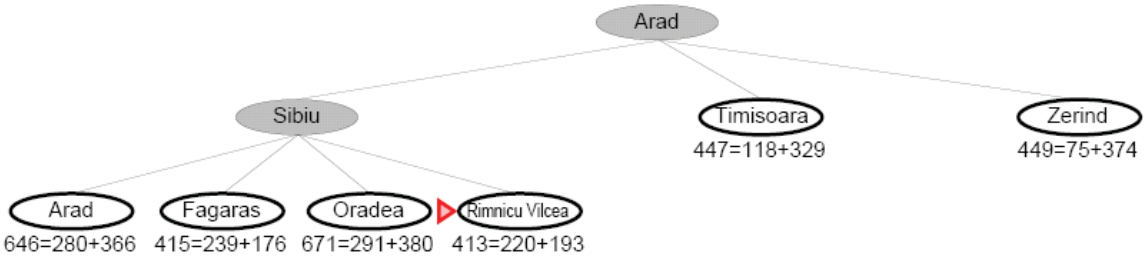
A* search example



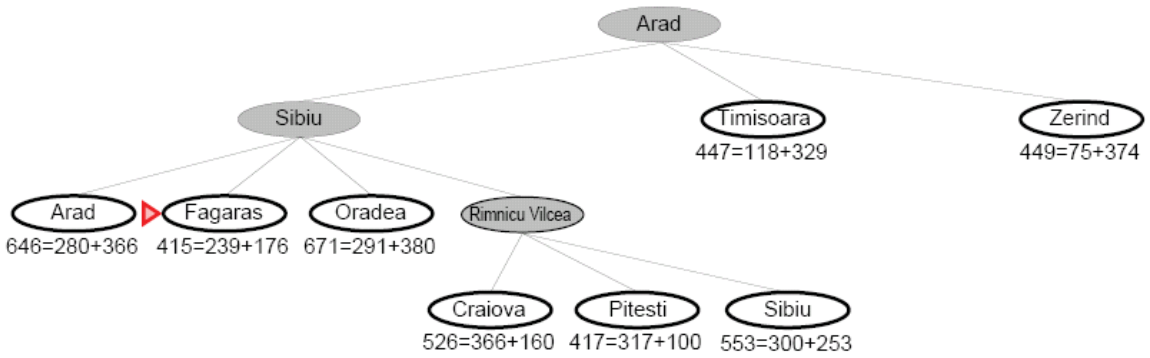
A* search example



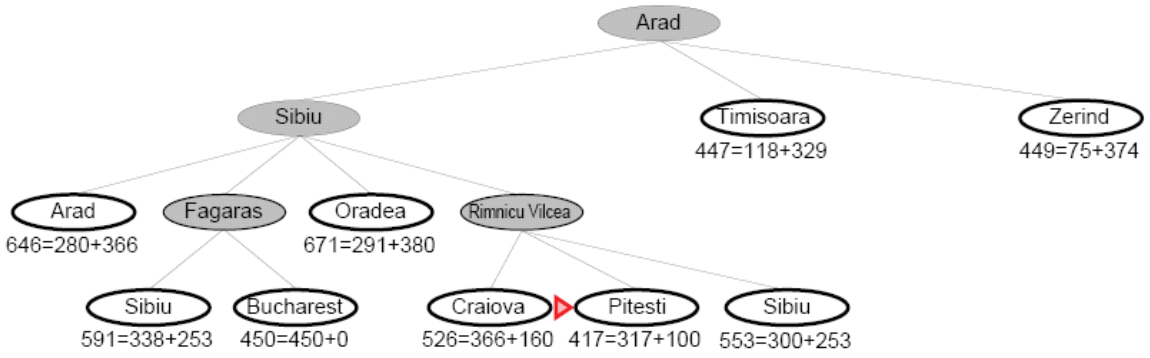
A* search example



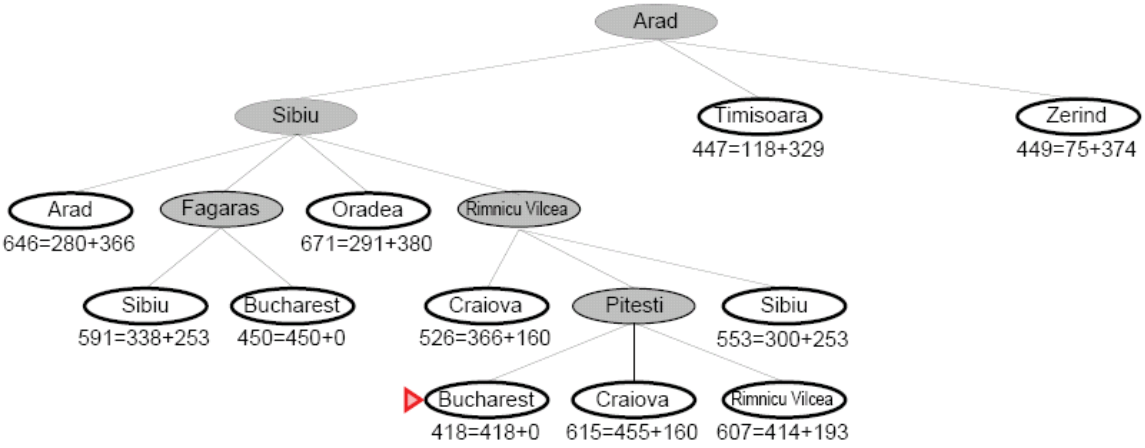
A* search example



A* search example



A* search example



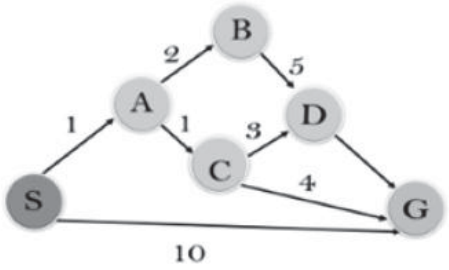
Total coast =418

Is this the optimum solution ? yes

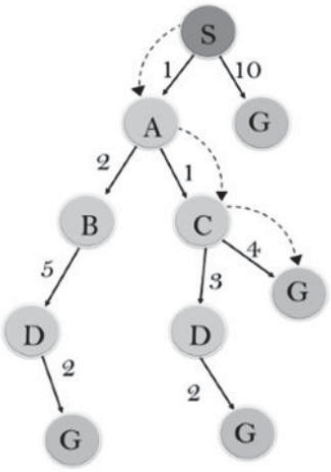
2. A* search

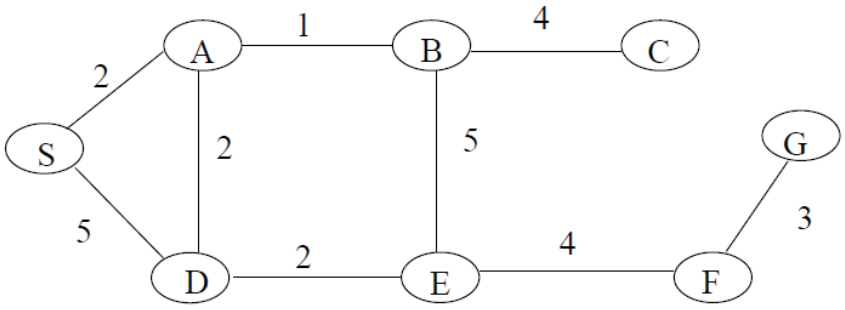
- Complete? yes
- Time? $O(b^d)$ Exponential
- Space? $O(b^d)$ keeps all nodes in memory (look at this)
- Optimal? yes

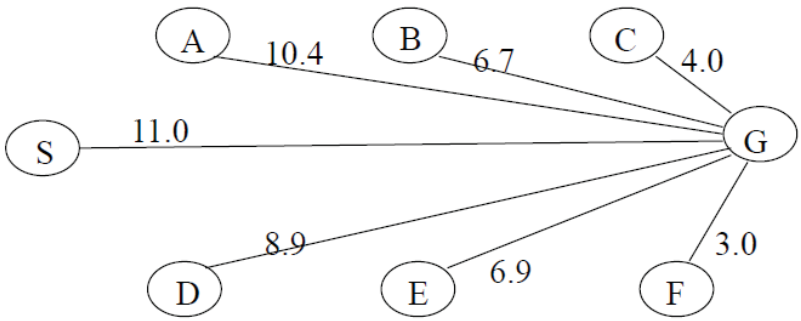
A* SEARCH ALGORITHM



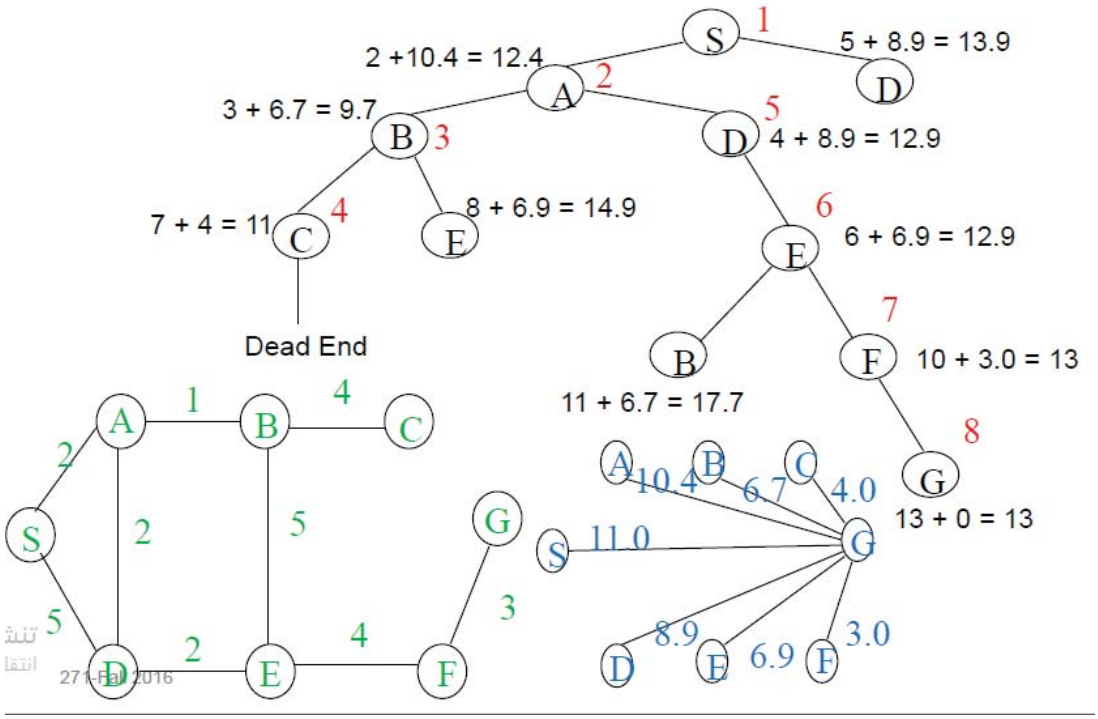
State	h(n)
S	5
A	3
B	4
C	2
D	6
G	0

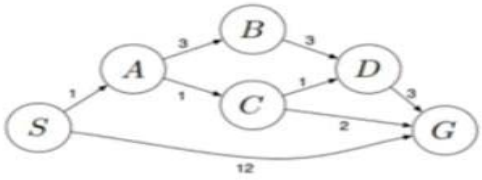


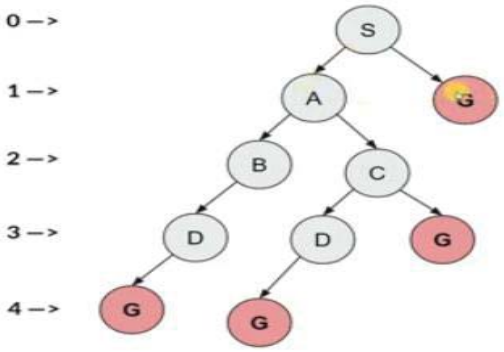




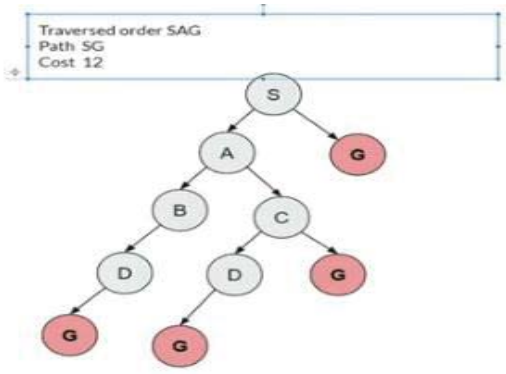
Example of A* Algorithm in Action

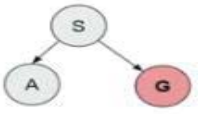




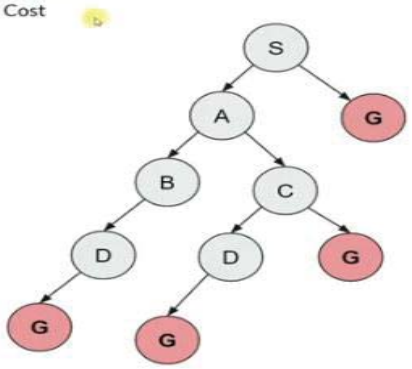


BFS Breadth First Search

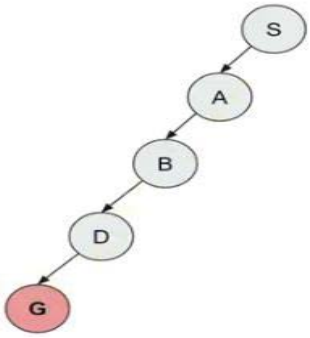




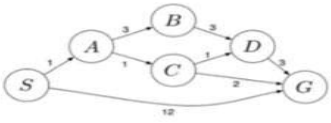
Visited
Path
Cost



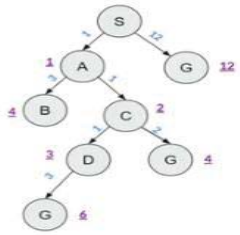
DFS Depth First Search

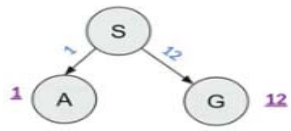


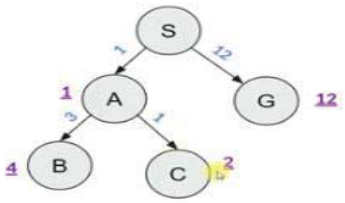
Visited
Path
Cost

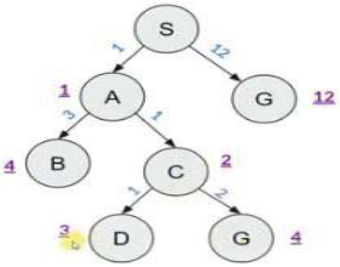


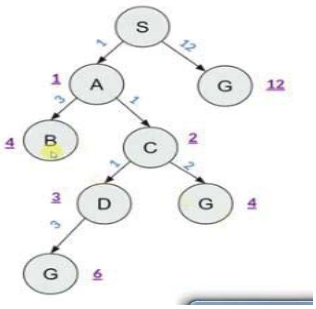
UCS Uniform Cost Search

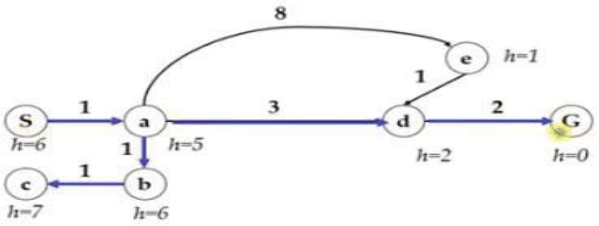




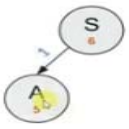


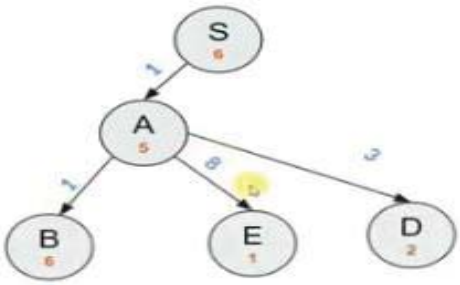


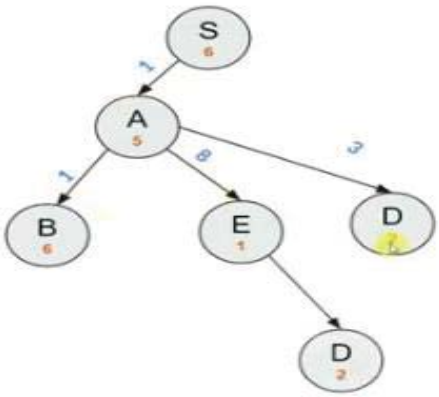


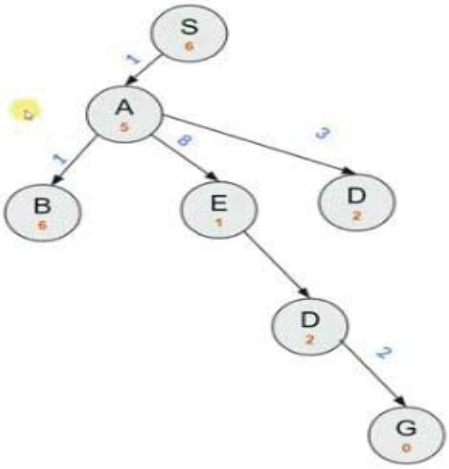


Best First Search

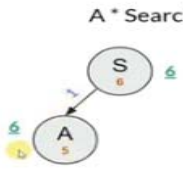




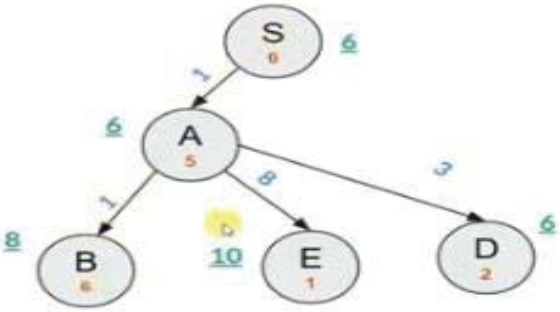


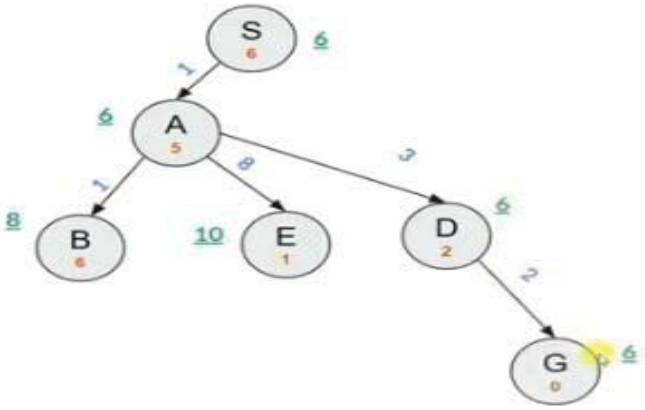


A * Search

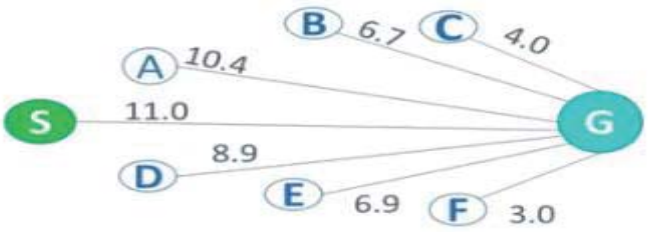


A * Search





مثال1: الانتقال من العقدة Sإلى العقدة G باستخدام خوارزمية A^* وذلك بأقل كلفة . الحل باستخدام شجرة البحث.

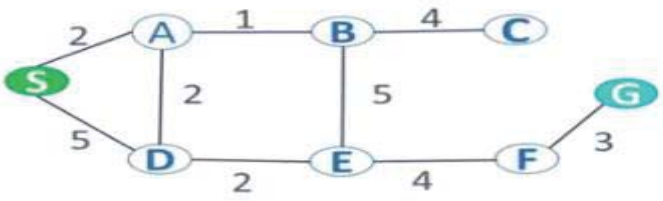


H(n):قيمة تجريبية لبعد النقطة الحالية عن الهدف

هذه الخوارزمية تحتاج إلى قيم تقديرية لبعد النقطة الحالية عن الهدف.
مثلا بعد النقطة A عن الهدف هي 10.4 وهكذا

A* algorithm

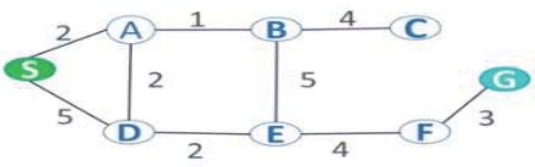
F(n)=G(n) +H(n)

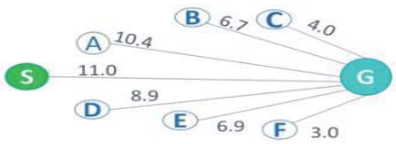


G(n):تكلفة الوصول من نقطة البداية إلى النقطة الحالية

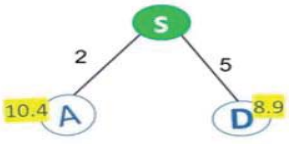
الأرقام الموجودة على المسارات هي تكلفة النقاط بين كل نقطين
مثلا تكلفة الانتقال من S للنقطة Aهي 2 وتكلفة الانتقال من Sللنقطة B هي 3 وهكذا

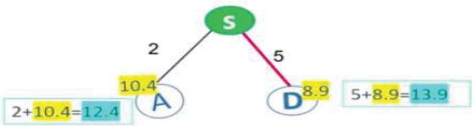
A* algorithm

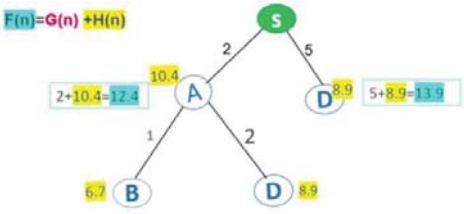




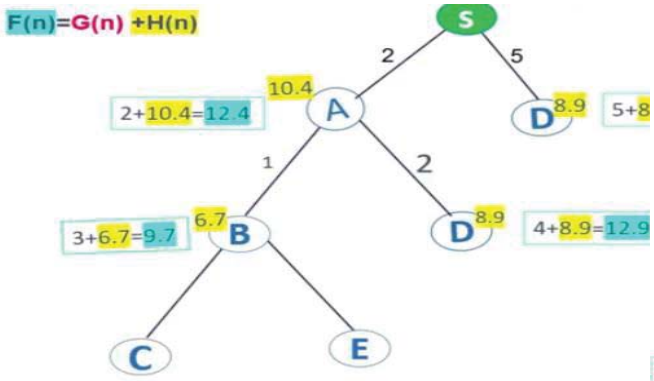
H(n):قيمة تجريبية لبعد النقطة الحالية عن الهدف

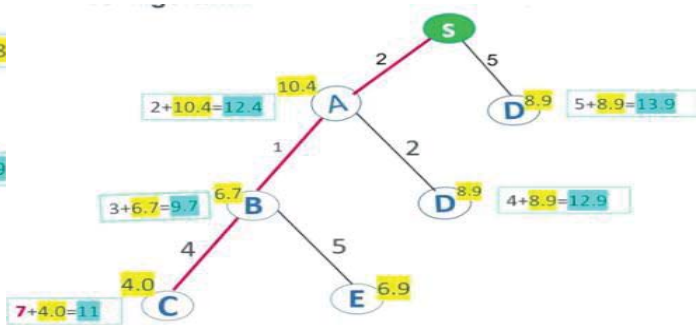




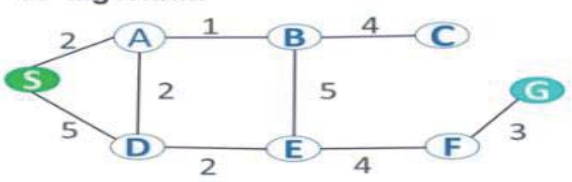


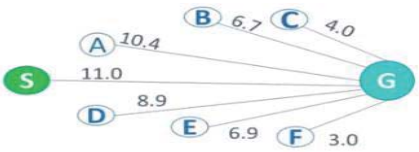
نبحث عن أصغر قيمة للتابع F ونتابع من خلالها



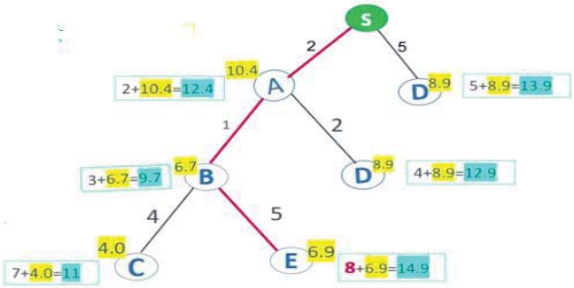


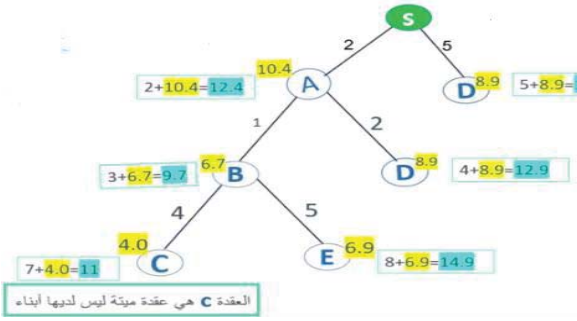
A* algorithm



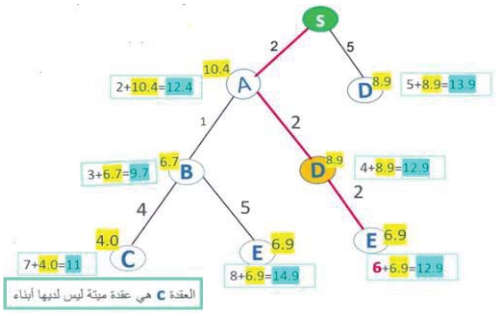


H(n):قيمة تجريبية لبعد النقطة الحالية عن الهدف

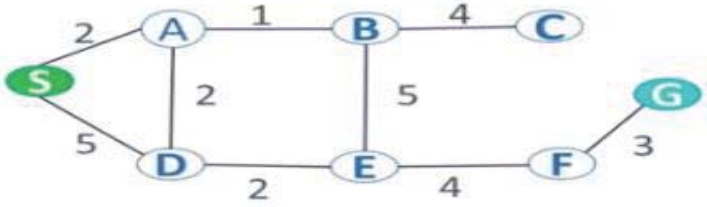


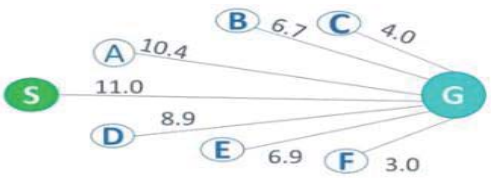


أصغر قيمة لـ F هي 11 عند النقطة C *ولكنها نقطة ميتة* لايمكن الوصول منها لأي مسار.
نبحث عن القيمة الأصغر نجدها عند النقطة D ولكن عن طريق النقطة A

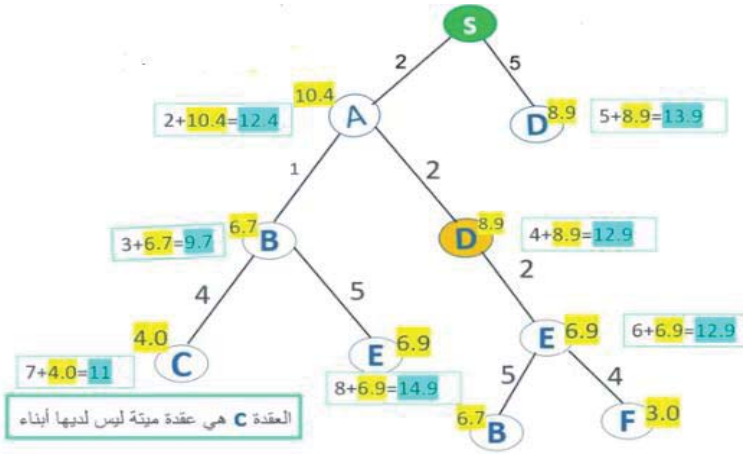


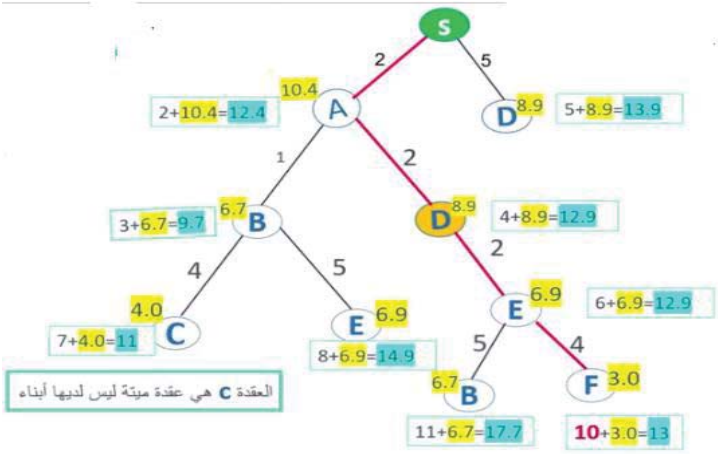
A* algorithm



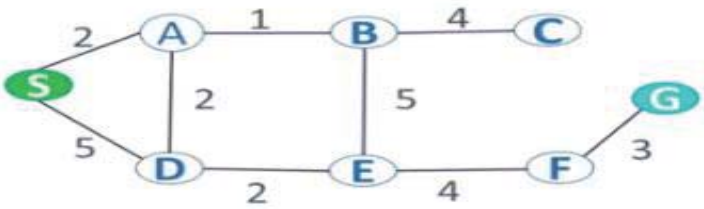


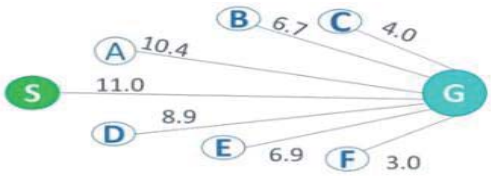
H(n):قيمة تجريبية لبعد النقطة الحالية عن الهدف



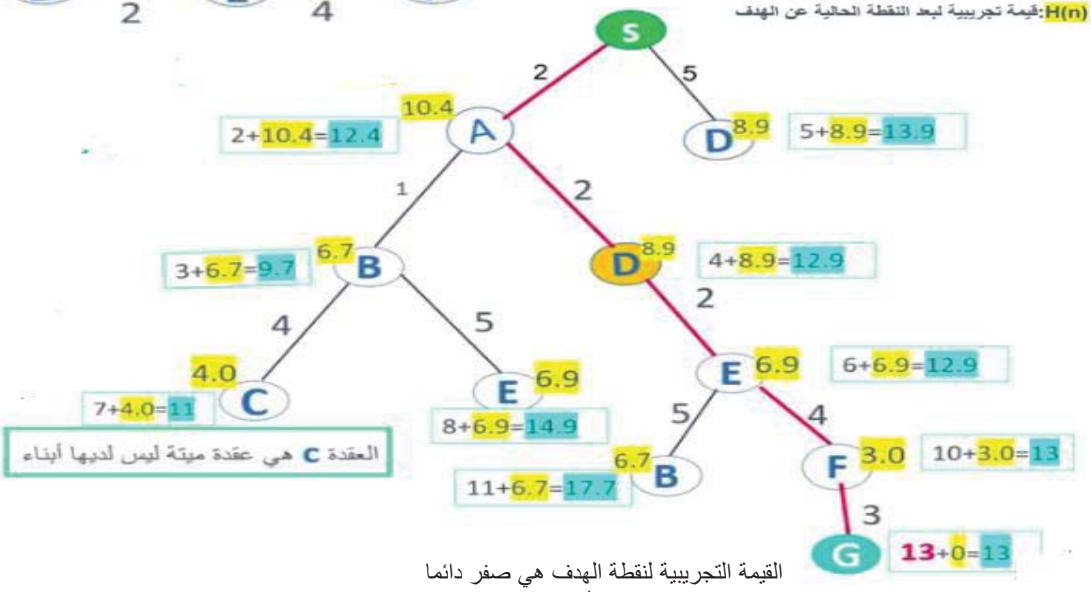


A* algorithm





H(n):قيمة تجريبية لبعد النقطة الحالية عن الهدف



القيمة التجريبية لنقطة الهدف هي صفر دائما
وبالتالي وصلنا للهدف بأقل تكلفة ممكنة