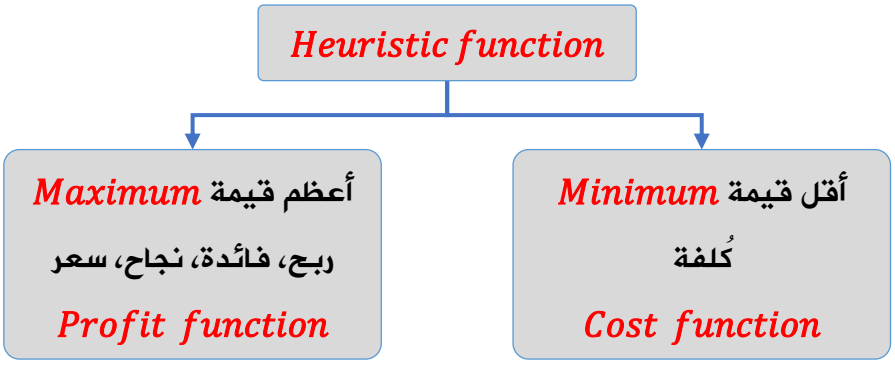
بِسْمِ اللَّهِ الرَّحْمَٰنِ الرَّحِيمِ

طريقة البحث الحدسية (Heuristic Search Method)

بالاعتماد على الحدس او التضمين أو البحث الاستدلالي (البحث الارشادي)

الحدس (Heuristic): هو الحكم على الأشياء بالخبرة التجريبية وهو يساعد الانسان على اتخاذ القرار فيما سيفعله مستقبلاً، لذا يعتبر الحدس هو أحد عناصر الذكاء الاصطناعي الأساسية وهو حالة تخمينية (حدسية) لأفضلية اختيار نقطة عن أخرى داخل الحل كبعض الأهداف او الأجزاء من المسألة التي تستخدم الدالة الحدسية او التخمينية في بعض خواصها ولها حالتين:



يمكن تعريف الحدس وفق مفهوم الذكاء الاصطناعي كالاتي:

هو ذلك النوع من علوم الحاسوب الذي يتخذ أسلوب المعالجة المرمزة (symbolic processing) لتمثيل المعرفة وليس المعالجة العددية (numerical processing) كذلك الحدس يحاكي (simulate) أسلوب الحدس عند الانسان في معالجة المعلومات مثل استخدام الحدس في توقع الأحداث المستقبلية وتحديد الأفعال المستقبلية.

البحث الحدسي (البحث الارشادي او البحث الاستدلالي)

هي استراتيجيات (Problem – Solving) التي في العديد من الحالات تجد الحل أسرع من البحث وهي غير مضمونة أي انها دالة تقييم او (تقدر) كل نقطة في مسألة الحل بالقيم العددية أو الأرقام الحقيقية.

في هذا من طُرق (استراتيجيات ، خوارزميات) البحث الحدسية سوف تجري عملية مفاضلة بين نقطة وأخرى بالاعتماد على قيمة عددية وهذه القيمة العددية تكون كبيرة أو صغيرة حسب نوع المسألة فمثلاً حساب أقل كُلفة فنأخذ أقل قيمة أو حساب أكبر ربح ففي هذه الحالة تأخذ أكبر قيمة.

يتضمن البحث الحدسي ما يلي:

1) دالة (كُلفة $Minimum$ ، ربح $Maximum$).
2) خوارزمية تستخدم تلك الدالة للبحث في فضاء الحالة $Tree$ أو $Graph$.

ان مبدا عمل هذه الطريقة انها لا تبحث في جميع العقد وأن مشكلــة هذه الطريقة تكمن في أننا قد لا نتوصل الى الهدف لأنها تعتمد على مبدأ معين وهو مقادير الكُلفة (حيث تأخذ الأقل كُلفة) ولكنها تكون سريعة جداً ورخيصة لأن البحث في اتجاه معين ومن مساوئها ليس من الضروري ايجاد الحل.

ملاحظة: في حالة عدم تحديد الهدف نختار الأقل كُلفة هي الهدف (عقدة الهدف الأقل كُلفة)

من طُرق البحث الحدسية او الاستدلالي او الارشادي

1- خوارزمية التسلق الشاهق تسلق الجبال $Hill\ climbing\ algorithm$
2- خوارزمية البحث الأفضل أولاً (الاسبقية للأفضل) $Best-first\ search\ Algorithm$.
3- خوارزمية A^* ($Algorithm\ A^*$).

مميزات طُرق (خوارزميات) البحث الحدسية او الإرشادية

1- مرونة كبيرة في التعامل مع التراكيب الصعبة والمشاكل المعقدة.
2- كفاءة عالية في حالة عدم مطابقة البيانات للواقع أو قد تكون البيانات المتوفرة غير كافية للاستنتاج واتخاذ القرار.
3- قدرة على التحليل النوعي لذا فهي تناسب حالات اتخاذ القرار.
4- يمكن استخدامها كجزء من إجراءات تكرارية لضمان الوصول الي الحل الأمثل اي انه يمكن دمجها مع طرق أخرى للحصول على أسلوب جيد للحل.
5- عند تمثيل المشكلة باستخدام الشجرة البحثية فان استخدام الطرق الحدسية يؤدي الى إنقاص حجم الشجرة وذلك بحذف العقد التي لا نتوقع أن تساعد كثيراً في الوصول للحل الأمثل $non\ problem\ nodes$

خوارزمية التسلق الشاهق (تسلق الجبال) *Hail climbing algorithm*

هي طريقة بحث حدسية مستخدمة أو تستخدم لمسائل الامثلية الرياضية في حقل او مجال الذكاء الاصطناعي (التقنيات الذكائية)

مُعطى لدينا مجموعة كبيرة من المدخلات والدالة الحدسية تحاول إيجاد الحل الجيد بشكل كافي للمسألة هذا الحل قد لا يكون الحل الأعظم الأمثل العالمي *global optimal maximum*

1- هنا يتم التعامل مع تكلفة ونختار الأقل تكلفة وتهمل بقية الكُلف الكبيرة (يتم ترتيب الكُلفة تصاعدياً).

2- كُلفة العقدة = كُلفة العقدة السابقة + كُلفة العقدة الحالية.

3- ان مشكلة هذه الطريقة قد لاتصل الى الهدف (إيجاد الحل) لأنها تعتمد على مقادير الكُلفة وهذا يؤدي الى الفشل في إيجاد الهدف (فشل في طريقة او خوارزمية *Hill climbing*

4- في هذه الطريقة لا يمكن الرجوع الي باقي العقد.

5- لا تؤخذ العقد المأخوذة مسبقاً حتى لا تتكرر.

6- في حالة تشابه القيم لعقدتين مختلفتين (الكُلفتين متساويتين) فتكون الأولوية لعقدة الأصل اي يتم اخذ القعدة التي على اليسار (من أقصى اليسار للمتكررة).

7- فيها كُلفة معينة لإيجاد الهدف ولا تحتاج الى جهد او خزن.

خطوات خوارزمية طريقة التسلق الشاهق

1- يتم فتح حقل التكرارات وحقل الـ ***Current State*** C_s الحالة الحالية الذي يمثل رؤوس انطلاق البحث.

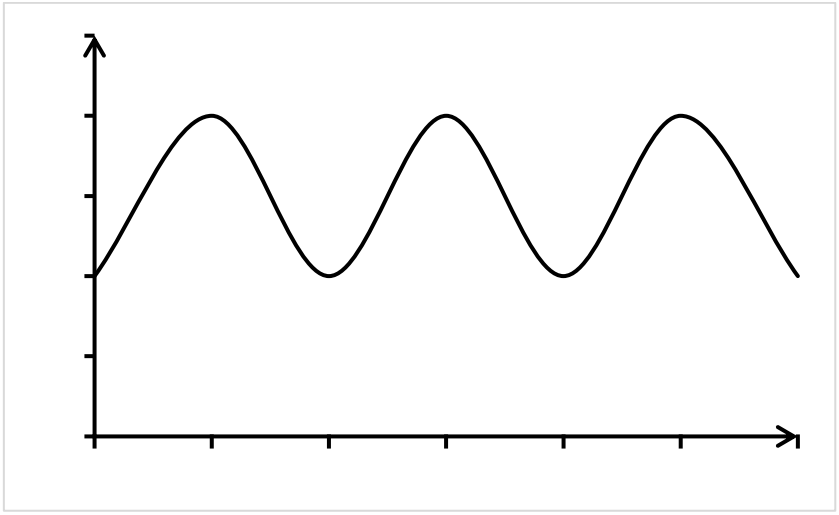
2- يتم فتح حقل الـ (***Open***) والذي يمثل الرؤوس التي تقع بنفس مستوى رأس الانطلاق.

3- يتم فتح حقل الـ (***Path***) والذي يمثل الطريق الذي يقطعه البحث من الرأس حسب الخاصية الحدسية انطلاقاً الى الرؤوس الاخرى.

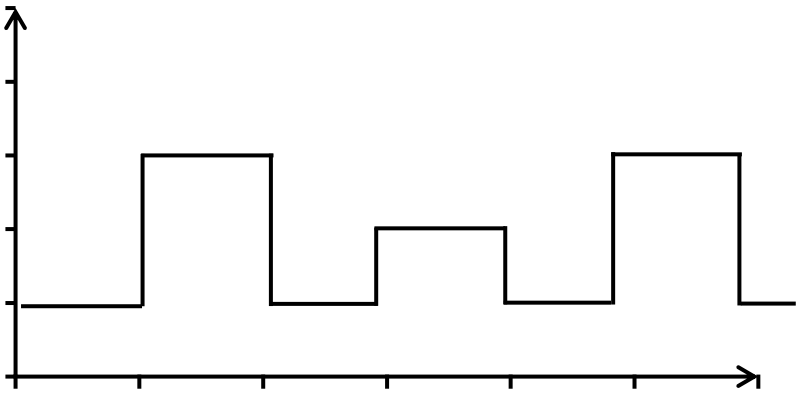
4- في الحقل الأخير (***Optimal***) يتم حساب العقد (الرأس) الأفضل (أقل قيمة للعقدة أو الرأس إذا كانت الدالة ***min*** وأعلى قيمة للرأس وإذا كانت الدالة ***max*** لحين الوصول الى الهدف.

مشاكل او مساوئ طريقة (خوارزمية)التسلق الشاهق

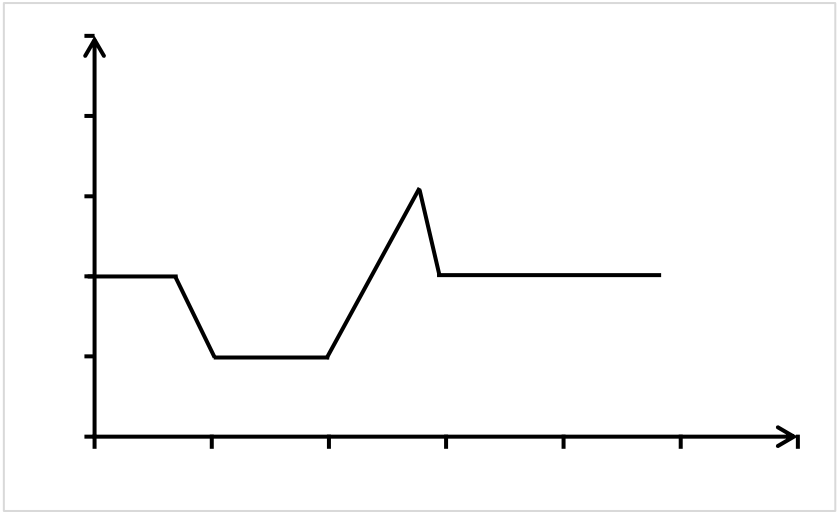
1- احتمال وجود صعود او نزول موضعي (النهايات) فيمكن اعتباره نقطة عظمى محلية او موضعية ***Local maximum*** اي اعتبار ان النقطة التي سوف نحصل عليها تكون موضعية



2- في بعض الأحيان نحصل على مساحة مستوية ***Plate*** بمعنى كلما نطبق القاعدة او طريقة يظهر لنا نفس الحل.



3- في بعض الأحيان قد يصادف او قد تأتي منطقة النزول ***Ridge***. او ما يسمى بالحافة وعند تمثيل الخطوات اللاحقة او التالية للخوارزمية سوف تؤدي الى تكرار تنفيذ القوانين مما يؤدي الى ضعف او إنهيار في الدالة الحدسية



4- احتمالية الكلفة في البداية قليلة ولكن قد تظهر الكلفة في النهاية الكبيرة

5- لا تستطيع التمييز بين شيئين متشابهين فمثلاً اذا كانت هنالك شخصين يمتلكون نفس الطول لا نستطيع التمييز بينهما

6- قد لا نصل الى او لا نحصل على الحل الأمثل دائماً (لا نصل الى الهدف) لأنه لا يوجد إغلاق ***Close*** ولعدم وجود أبناء جدد للرأس أو العقدة (اي لا يوجد أبناء او لا يوجد علاقة) إذا لا يوجد حل.

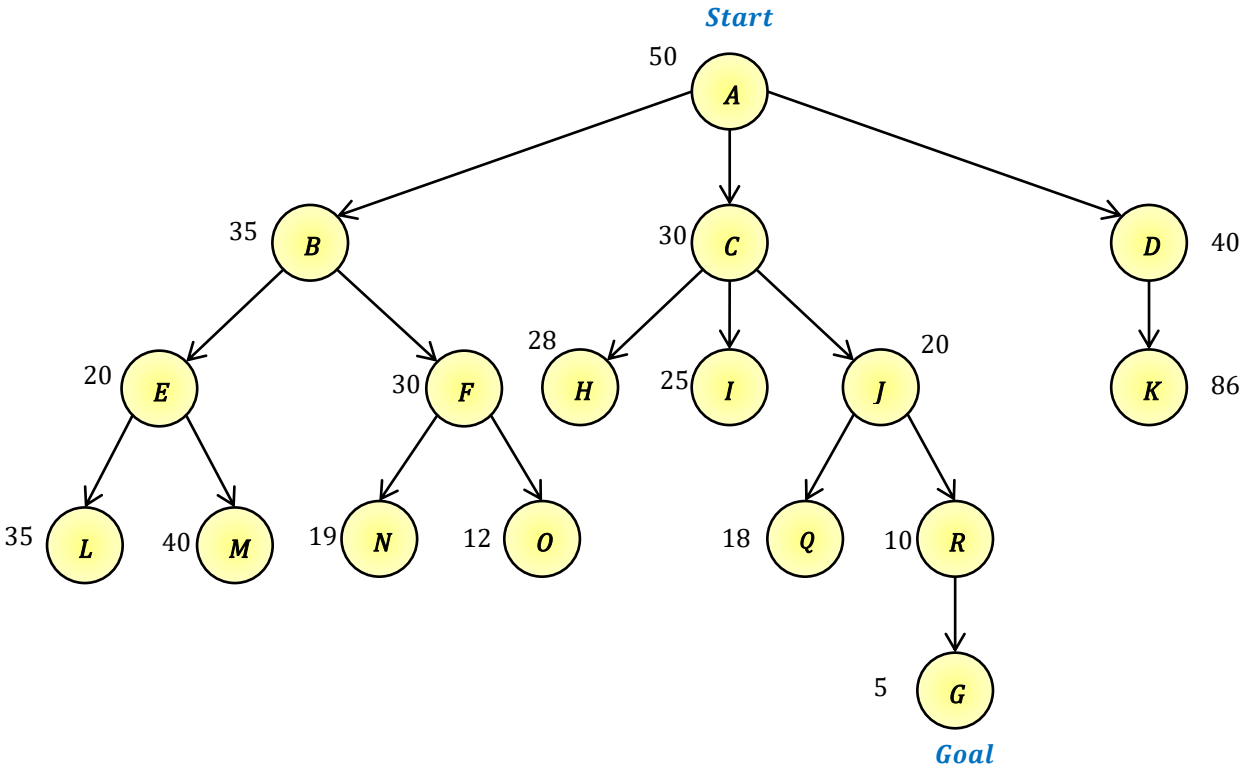
الحلول لهذه المشكلة

1- الرجوع الى الوراء (الى القعدة السابقة) ومحاولة الذهاب في اتجاهات مختلفة.

2- عمل قفزة طويلة (جديدة) في بعض الاتجاهات للوصول الى مجال جديد في البحث (مجال بحث جديد).

3- طبق القاعدتين أو أكثر (يتم تطبيق قاعدة او قاعدتين او أكثر) قبل عمل الاختبار (الفحص) اي عمل حركة بعد اتجاهات قبل عملية الاختبار.

مثال:



Iteration	C_s	*Open*	*Path*	*Optimal*
1	$[A_{50}]$	[ ] *or* ∅	[ ] *or* ∅	[ ] *or* ∅
2	$[C_{80}]$	$[C_{80}, \cancel{B_{85}}, \cancel{D_{90}}]$	$[A_{50}, C_{80}]$	$B = 85, C = 80, D = 90$ **$X = C_{80} = 80$ اقل تكلفة**
3	$[J_{100}]$	$[J_{100}, \cancel{I_{105}}, \cancel{H_{108}}]$	$[A_{50}, C_{80}, J_{100}]$	$H = 108, I = 105, J = 100$ **$X = j_{100} = 100$**
4	$[R_{110}]$	$[R_{110}, \cancel{Q_{118}}]$	$[A_{50}, C_{80}, J_{100}, R_{110}]$	$Q = 118, R = 110$ **$X = R_{110} = 110$**
5	$[G_{115}]$	$[G_{115}]$	$[A_{50}, C_{80}, J_{100}, R_{110}, G_{115}]$	**$X = G = 115$** *Optimal solution*
التكرار	رؤوس الانطلاق	الرؤوس بنفس مستوى الانطلاق	المسار الذي يقطعه البحث	الرأس الأمثل

$$C_s: A \rightarrow C \rightarrow J \rightarrow R \rightarrow G$$

بسم الله الرحمن الرحيم

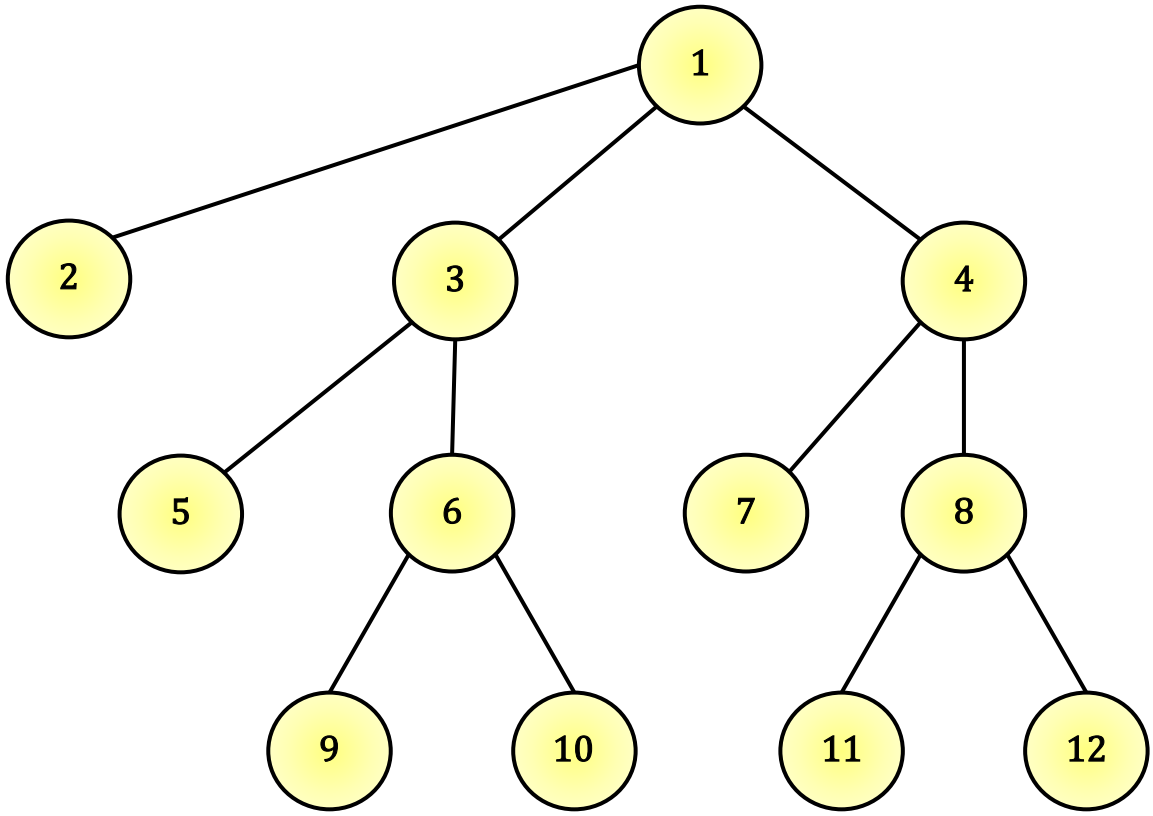
خوارزمية البحث المستعرض (البحث الافقي أو الموسع)

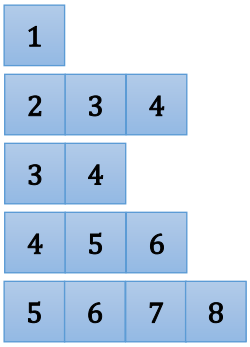
$Breadth - first\ search\ Algorithm$

هي ايضاً من طرق البحث الاعمى $(Blind\ Search)$ وهي تعتمد في عملية البحث على السير في مستويات أفقية حيث تنتقل من مستوى الى المستوى الذي يليه الى ان نصل الى الهدف.

تبدأ من اليسار (تأخذ مستوى مستوى) تشبه طريقة البحث بعمق ولكن توضع العقد المتفرعة في نهاية مصفوفة $(Open)$ وعملية فحص $(Nodes)$ تكون مستوى مستوى.

- تبحث بشكل مستوى مستوى وبشكل افقي أي تبحث كل مستوى على حدة من اليسار الى اليمين.
- يتم إدخال الأبناء من جهة اليمين في خانة الـ $(Open)$ بعكس طريقة الـ $(Depth)$ يتم ادخال الأبناء من جهة اليسار.





المبدأ الذي تستخدمه خوارزمية البحث المستعرض هو الطوابير $(Queu)$ أي:

$(First\ In\ Last\ Out)\ (FILO)$

$(Last\ In\ Last\ Out)\ (LILO)$ أو

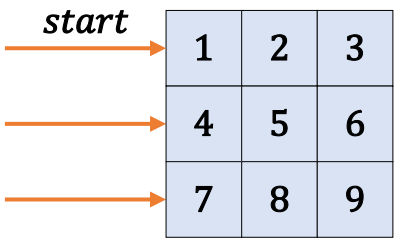
تعتبر خوارزمية البحث المستعرض من خوارزميات البحث الأعمى لأنها لا تعتمد في عملية الانتقال من عقدة الى أخرى على قانون معين ولكن في هذا النوع من الخوارزميات تكون عملية الوصول الى الحل اكيدة حيث ان عملية البحث تسير في مستويات افقية فتبدأ بأول مستوى وتكون الحالات فاذا كانت هذه الحالات لا تحتوي على الهدف ننتقل الى المستوى الثاني وهكذا الى أن نصل الى الهدف.

آلية عمل خوارزمية البحث المستعرض

$Breadth - first\ Search\ Algorithm$

1) تبحث في كل صف مستوى (صف تلو صف).

2) البحث في اليمين ليس مثل $(Depth)$ حيث البحث من جهة اليسار.



3) ان خطوات الحل في هذه الطريقة مشابهة لطريقة الـ $(Depth)$ لكن تختلف في النقطة (1).

خصائص طريقة (خوارزمية) البحث المستعرض أو البحث الافقي

1) تكون مُكلفة في الوقت والثمن.

2) البحث يكون مستوى مستوى لحين الوصول للهــدف.

3) يتم إدخال الأبناء من جهة اليمين.

4) أول عقدة تدخل هي أول عقدة تخرج

5) لا تحتوي على التراجع الخلفي $(Back\ tracing)$ لان البحث فيها بشكل مستوى.

الفرق بين خوارزمية البحث بعمق وخوارزمية البحث المستعرض

مهمة جداً

خوارزمية البحث بعمق (العمودي) $Depth - First\ Search$	خوارزمية البحث المستعرض (الافقي) $Breadth - First\ Search$
1)	
تستخدم مبدأ التكديس $Stack$ اي $FIFO$ or $LIFO$	تستخدم مبدأ الطوابير $(Queues)$ $(LILO)$ or $(FILO)$
2)	
تستخدم فضاء الحالة بعيداً عن الهدف	تستخدم فضاء الحالة قريباً عن الهدف
3)	
البحث عمودياً (بشكل عمودي) حيث يتم البحث بعمق في جميع المستويات حتى نصل الى الهدف وتبدأ من اليسار (تبحث عمقاً يسارًا)	يتم البحث بشكل افقي في جميع المستويات حتى نصل الى الهدف وتبدأ من اليسار (يكون البحث فيها مستوى مستوى أي بشكل مستوى)
4)	
يحتوي هذا البحث على تعقب الأثر الرجعي وهذا يؤدي الى ضياع الوقت أي نستطيع الرجوع الخلفي (التراجع الخلفي) $(Back\ tracing)$	لا تحتوي على تعقب الاثر الرجعي أي لا يوجد او لا يمكن الرجوع او (لا يمكن التراجع الخلفي)
5)	
يتم إضافة (ترتيب) العقد (الأبناء) من جهة اليسار	يتم إضافة (ترتيب) العقد (الأبناء) من جهة اليمين
6)	
العقدة الأولى التي تدخل تكون العقدة الأخيرة التي تخرج	العقدة الأولى التي تدخل تكون العقدة الأولى التي تخرج

7)

تكون حركة العقد فيها بالشكل الاتي:	تكون حركة العقد فيها بالشكل الاتي:
1 4 7 2 5 8 3 6 9	1 2 3 4 5 6 7 8 9

8)

قد لا تحتاج الى وقت و جهد عالي	تحتاج الى وقت جهد

9)

ليست مضمونة في الوصول الى الحل دائماً (ليس دائماً متقارب)	مضمونة الوصول الى الحل دائماً

10)

أقل تعقيداً	أكثر تعقيداً

11)

شخص يبحث عن قاعة	لعبة الورق

تعليمات الخوارزمية مهمة جداً

Depth–First Search Algorithm (FIFO)

1- $Put\ the\ start\ node\ in\ open$

2- $Loop: if\ open\ is\ empty\ then\ exit\ (fail)$

3- $N: first\ (open)$

4- $If\ goal\ (n)\ then\ exits\ (success)$

5- $Generate\ a\ children\ of\ \boldsymbol{n}$

6- $Remove\ (n, open), add\ (n, closed)$

7- $Expand\ \boldsymbol{n}\ by\ replacing\ that\ node\ by\ it\ child\ at\ the\ right\ of\ open$

8- $Go\ to\ loop\ in\ step\ 2$

1- نضع عقدة البداية في $Open$

2- $Loop$: اذا كان $open$ فارغ اذاً فشل تم الخروج

3- N: هو أولا في $open$

4- اذا كانت (n) هي الهدف اذاً نجح البحث (تحقق النجاح) وخروج

5- هل الـ n لها أبناء (تولد أبناء من n) اضفها في يسار الـ $close$

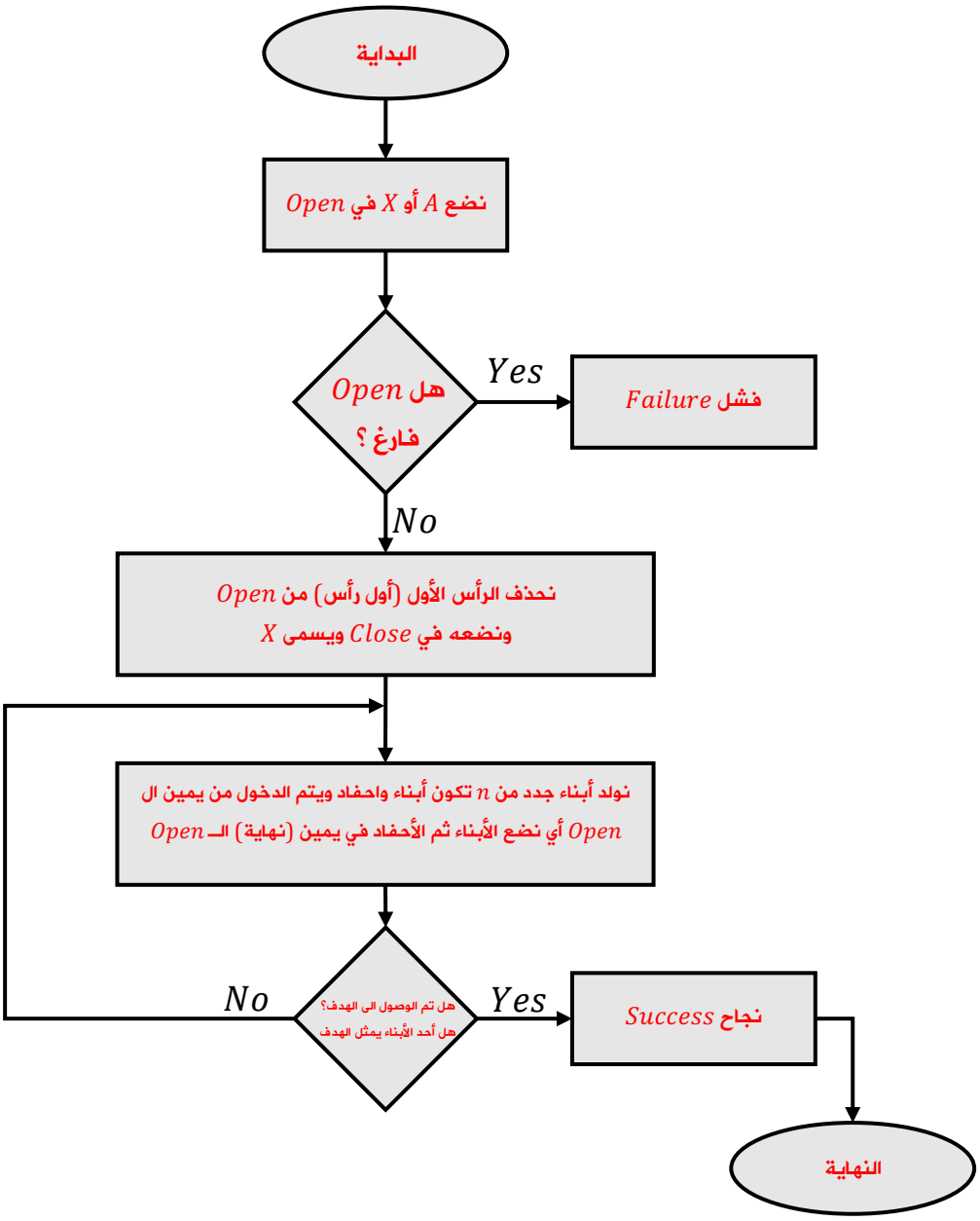
6- نبدأ بإزالة $(n, open)$ n وتضاف n الى $(n, closed)$

7- يتم توسيع n وذلك عن طريق استبدال العقدة n في $Open$ لعقد الأبناء ومن جهة اليمين (أي استبدل تلك العقد التابعة على يمين الفتح)

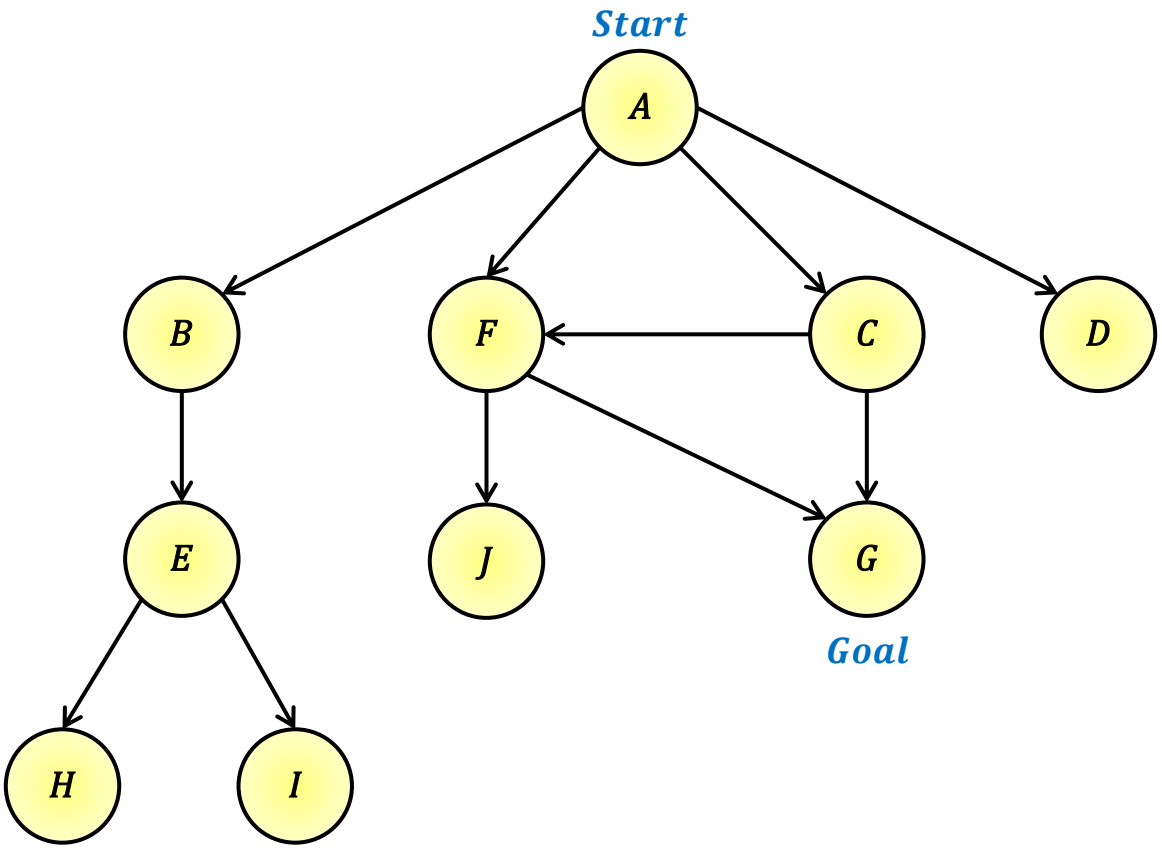
8- اذهب الى $Loop$ (الخطوة الثانية).

المخطط الانسيابي لخوارزمية البحث المستعرض (مهم جداً)

Flowchart of depth–first search algorithm

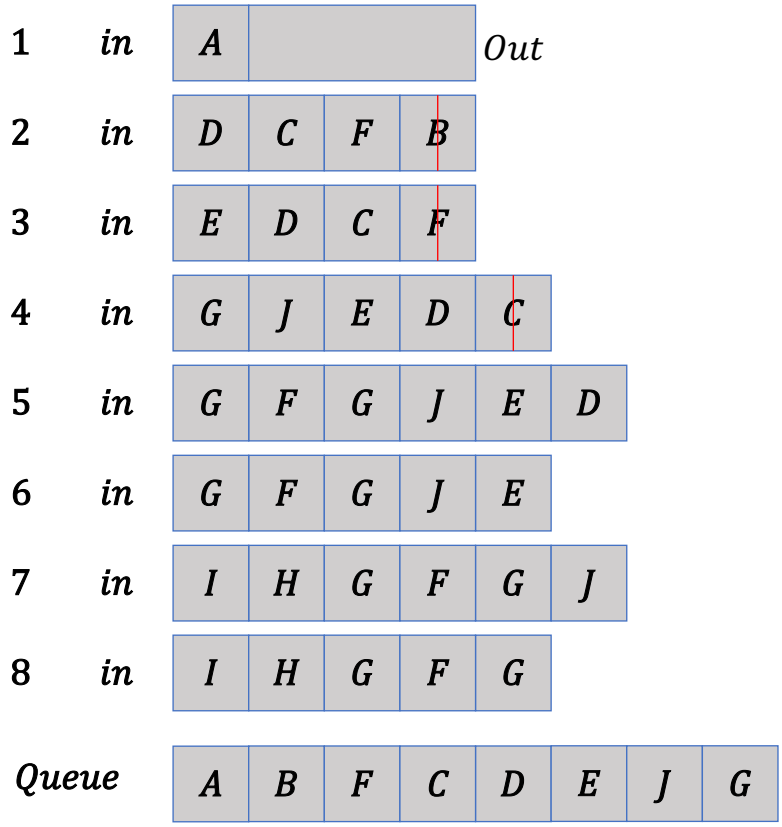


مثال: باستخدام خوارزمية البحث المستعرض جد طريقة الوصول الى (G) علماً أن الحالة الابتدائية هي (A).



Iteration	Open	Closed
1	$[A, 0]$ *or* A	[ ] *or* $\emptyset$
2	$[(B, A), (F, A), (C, A), (D, A)]$	$[(A, 0)]$
3	$[(F, A), (C, A), (D, A), (E, B)]$	$[(B, A), (A, 0)]$ *or* $B\ A$
4	$[(C, A), (D, A), (E, B), (J, E), (G, F)]$	$[(F, A), (B, A), (A, 0)]$
5	$[(D, A), (E, B), (J, F), (G, F), (F, C), (G, C)]$	$[(C, A), (F, A), (B, A), (A, 0)]$ *or* $C\ F\ B\ A$
6	$[(E, B), (J, F), (G, F), (F, C), (H, E), (I, E)]$	$[(D, A), (C, A), (F, A), (B, A), (A, 0)]$
7	$[(J, F), (G, F), (F, C), (G, C), (E, E), (I, E)]$	$[(E, B), (D, A), (C, A), (F, A), (B, A), (A, 0)]$
8	$[(G, F), (F, C), (G, C), (H, E), (I, E)]$	$[(I, F), (E, B), (D, A), (C, A), (F, A), (B, A), (A, 0)]$
9	G *is the Goal*	

كيفية تمثيل البيانات في *Queue* احسب مبدأ *FIFO* أو *LIFO*



تطبيق على الأرقام السرية

مثال: لدينا ثلاثة أرقام سرية معروفة التسلسل {1,2,3}.