البحث غير المطلع (uninformed Search) أو البحث الأعمى (Blind Search)

سوف نفترض أن الوكلاء الأذكياء قادرون على جعل معايير الأداء تأخذ أفضل القيم،وان الوكلاء قادرون على الوصول إلى هدفهم ويسعون للوصول إليه. أولا سوف نستعرض كيف ولماذا يستطيع الوكيل الحصول على تلك القدرة.

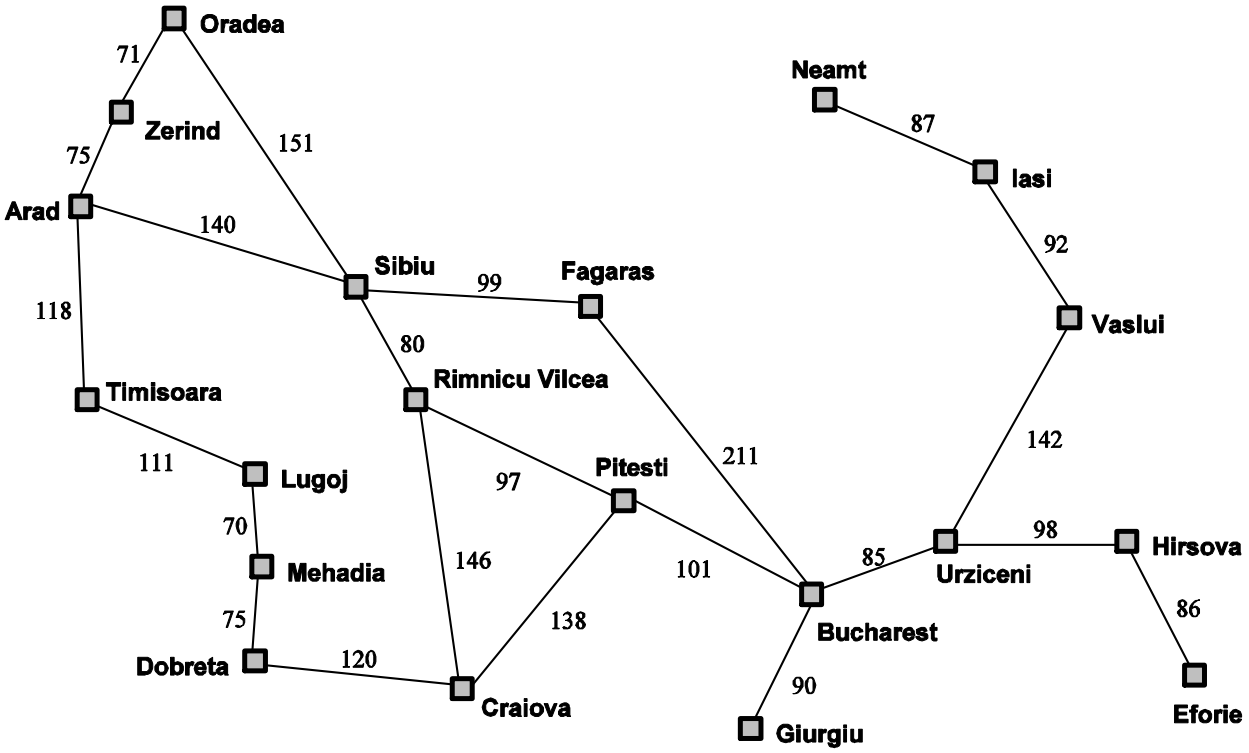


Figure1

ليكن لدينا وكيل معين موجود في مدينة Arad في رومانيا ويقضي عطلته كسائح ، أن معيار الأداء (**Performance Measure**) بالنسبة له يحوي العديد من العوامل: فهو يريد أن يتمتع بالشمس، وان يحسن مستواه في اللغة الرومانية، يتمتع بمباهج الحياة الليلية،يشاهد معالم المدينة... الخ.أضف إلى ذلك عند وكيلنا هذا تذكرة سفر غير قابلة للتأجيل للسفر منBucharest في اليوم التالي (انظر Figure1).إذا أن ما يسعى إليه وكيلنا (هدف Goal)هو الوصول إلى Bucharest في الوقت المناسب.بناء على هذا فإن كل الأفعال (**Courses Of Action**) التي لن تؤدي ذهاب الوكيل في الوقت المناسب إلىBucharest ستكون مرفوضة من قبل الوكيل. إن تحديد الهدف يؤدي إلى تنظيم سلوك الوكيل و تقليل الأمور التي سيختار الوكيل من بينها خطوته التالية، وهكذا فأن أول ما يجب أن يقوم به الوكيل هو**صياغة الهدف (Goal Formulation) .** يعتمد تحديد الهدف على الوضع الحالي وعلى معيار الأداء

سوف نعتبر الهدف مجموعة من الحالات التي يعتبر الوصول إليها تحقيقا للهدف، بحيث تكمن مهمة الوكيل في تحديد سلسلة الأفعال (**Sequence Of Actions**) التي ستقوده إلى حالة هدفية (Goal State).قبل أن ينجز مهمته يجب على الوكيل أن يحدد ما هي الأحداث التي سيقوم بها وما هي الحالات التي سيدرسها كي يصل إلى الهدف. يطلق على ذلك صياغة المشكلة (**Problem Formulation**).

وهكذا فإن وكيلنا قرر أن يصل إلى Bucharest راكبا سيارة منطلقا من Arad . واضح من الخريطة أن هناك 3 مدن أخرى تتصل ب Arad وهي Zerind و Timisoara و Sibiu حيث أن الوصول إلى من هذه المدن لا يعتبر هدفا وصولا للهدف لذلك فالوكيل الذي لا يعرف خريطة

رومانيا لن يعرف أي طريق يجب أن يسلك ، بعبارة أخرى الوكيل لا يعرف ما هو أفضل حدث لأنه لا يملك معرفة كافية عن الحالة التي سيؤول إليها وضعه إثر تنفيذ أي حدث. وإذا لم يستطع الوكيل الحصول على أي معلومات إضافية فسيكون في مأزق وأفضل ما يمكنه فعله هو اتخاذ أي اتجاه بشكل عشوائي.

سنفترض أن وكيلنا يملك خريطة رومانيا (سواء في ورقه أو في ذاكرته). تؤمن الخريطة للوكيل معلومات عن الحالات التي سيؤول إليها والأحداث التي يمكن أن يقوم بها.يستطيع الوكيل استخدام هذه المعلومات لإيجاد سلسة الأحداث التي تمكنه من الوصول إلى الهدف. نطلق على عملية إيجاد هذه السلسلة **البحث (Search)**.إن أي خوارزمية بحث تأخذ كمدخلات مشكلة معينة وتعيد كخرج **الحل (Solution)** على شكل سلسلة من الأحداث يؤدي **تطبيق** هذه الأحداث إلى الوصول إلى الهدف. ببساطة يتصرف الوكيل(صياغة-ايجاد-تطبيق)

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

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

في الشفرة الزائفة السابقة تقوم الدالة **U**PDATE-**S**TATE بتحديد الحالة الراهنة ومن ثم يُحدَد الهدف بواسطة الدالة FORMULATE-GOAL يلي ذلك البحث الذي يُجرى بواسطة العملية SEARCH والتي تعيد سلسلة الحل seq .بعد ذلك تخبرنا الدالة FIRST بأول حدث يجب القيام وتعيده إلى action وتغير سلسلة الأحداث بحيث تنتزع أول حدث. لاحظ أن وكيلنا هنا يتجاهل التغيرات الناتجة عن تصرفه هو، لأنه يفترض أن الحدث التالي من سلسلة الأحداث قابل للتطبيق دائما بغض النظر عن تأثير تطبيق أي حدث منها على البيئة المحيطة. الجدير بالذكر أن البيئة المفترضة من أجل الوكيل الذي قام بحل المشكلة في الشفرة الزائفة السابقة يمكن وصفها كما يلي:

بيئة ساكنة (static) لان صياغة الهدف والمشكلة والحل بدون الالتفات إلى تغيرات قد تحدث في البيئة؛ بيئة واضحة (observable) لان الحالة الابتدائية واضحة.كوننا نستطيع أن نذكر جميع البدائل المختلفة (عددها محدود) التي يمكن أن يمر بها الوكيل يعني أن البيئة متقطعة (discrete). وأخيرا واهم صفة أن البيئة محددة لأنه يمكن معرفة الحالة التي سيكون فيها الوكيل بعد إجراء أي **حدث ولا يأخذ بعين الاعتبار أي تغيير مفاجئ. علاوة على ذلك فالوكيل يتجاهل التغييرات التي تحدث بسبب أفعاله (يتصرف مغمض العينين بعد تحديد سلسلة الحل ولا يستشعر بيئته).**

Well-defined problems and solutions

تحديد المشاكل و الحلول

إن المشاكل تكون محددة بشكل واضح بواسطة أربعة عناصر :

- **الحالة الابتدائية** (The **initial state**) وهي الحالة التي يكون فيها الوكيل عند بدء العمل وفي المثال السابق فإن الحالة الابتدائية هي وقوعه في مدينة Arad ، (In(Arad.
- وصف لمجموعة الإحداث أو ردود الأفعال (**Actions**) المتاحة للوكيل والنتائج التي يمكن الحصول عليها عند تطبيق احد هذه الأحداث يمكن أن نسميها **دالة تحديد الوريث successor function** (ألا يذكرنا هذا بال Automata) مثلا

{<Go(Sibiu) , In(Sibiu)> ,
<Go(Timisoara),In(Timisoara)>,
<Go(Zerind),In(Zerind)>}

في الحقيقة إن الحالة الابتدائية ودالة الانتقال تشكلان معا فضاء الحالات لهذه المشكلة (**state space**) وهو مجموعة كل الحالات التي يمكن أن نصل إليها من الحالة الابتدائية. إن فضاء الحالات عبارة عن بيان Graph عقده (Vertices Or Nodes)هي الحالات و أقواسه (Edge) هي الأحداث وسنسمي مجموعة سلسلة الحالات المرتبطة مع بعض بأحدث معينة ب الطريق (Path)

- **اختبار الهدف** (The **goal test**) : إن توفر هذا العنصر يعني تحديد الوصول إلى الهدف من عدمه. قد يكون الهدف حالة واحدة أو مجموعة من الحالات، ويعتبر الوصول إلى أي واحدة منها وصولا إلى الهدف . في مثالنا السابق كانت الحالة-الهدف هي (**Bucharest**)In .
- **تكلفة المسار (path cost)** هي دالة تقوم بحساب تكلفة كل طريق (من الطرق التي نجدها أثناء البحث) وتعطي لها قيمة عددية، ويختار الوكيل الطريق الذي يناسب معايير الأداء لديه. يجدر بنا أن نشير إلى أن قيمة الخطوة (Step Cost) هي تكلفة الانتقال من الحالة x إلى الحالة y بإجراء الحدث a ونعبر عن ذلك بالرمز $c(x, a, y)$. في الشكل السابق نرى تكلفة الانتقال كل مرحلة. أن **حل المسالة** Solution هو طريق يصل بين الحالة الابتدائية والحالة النهائية (حالة هدفية). لاحظ في الخريطة السابقة نرى التكلفة تمثل المسافات بين المدن. في الحقيقة إن التكلفة هي من يحدد جودة الحل ولذا فالحل الأمثل (**optimal solution**) هو حل يأخذ اقل تكلفة من بين كل الحلول المتوفرة.

Formulating problems

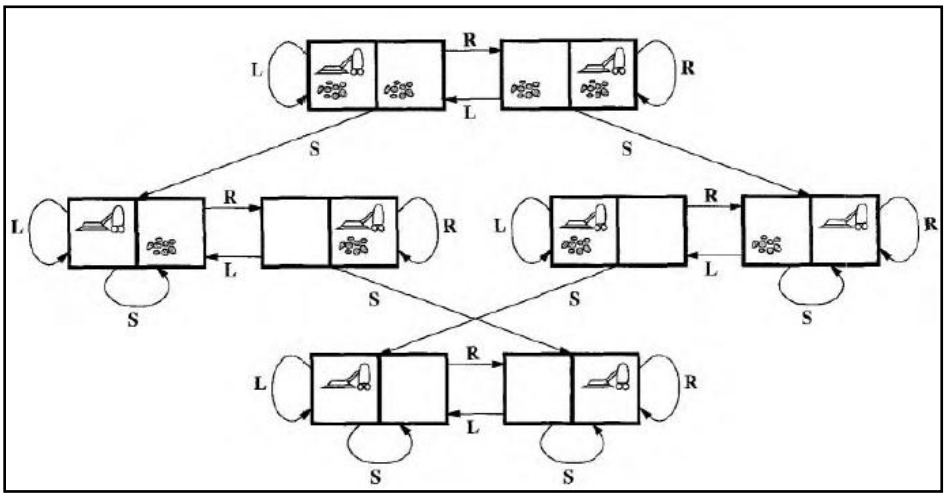
كنا قد وصفنا المشكلة في البند السابق بإستخدام الحالات الابتدائية ودالة الانتقال (دالة الوريث**successor function**) واختبار الهدف وتكلفة المسار، ولكننا لم نضع في حسابنا الظروف الخارجية التي قد يمر بها الوكيل فمثلا في مثال السائح في رومانيا لم نبالي بالمناظر على جانبي الطريق ولا حالة الطريق ولا الطقس و لا الكثير من الأمور التي لا تؤثر على سير العمل. في الحقيقة أن عملية التخلص من التفاصيل التي لا تؤثر على جوهر المشكلة تسمى تجريدا (**abstraction**).

لا يكون التجريد في **الظروف البيئة المحيطة** بالوكيل فقط وإنما يكون أيضا في **أحداث وتصرفات** الوكيل؛ فنحن لا نتحدث في المثال السابق عن تعبئة الوقود في الطريق ولا صعود وهبوط الركاب فقط نهتم بتغير موقع الوكيل بالنسبة للخريطة . نقول عن تجريد انه مفيد (*useful*) إذا كان تنفيذ كل حدث من أحداث (action) الحل يصبح أسهل في النموذج مما هو عليه في المسالة الأصلية

أمثلة على تحديد المشاكل

سنتناول بعض الأمثلة المبسطة (**Toy problems**) لشرح المفاهيم السابقة ولكن قبل ذلك نود نريد أن نوضح أن الأمثلة المبسطة هي أمثلة توضع لعرض أو اختبار طرق الحل المختلفة (**various problem-solving methods**).

مثال 1



في مثال المكنسة الكهربائية الذي تناولناه سابقا نلاحظ أن

1) **الحالات** : يمكن أن يكون الوكيل الذكي (المكنسة الكهربية هنا) في احد مربعين كل واحد منهما قد يكون نظيفا وقد يكون غير نظيف وعليه فان هناك 2^2 x 2 من الحالات المختلفة (حالة الغرفة الأولىxحالة الغرفة الثانيةx موقع المكنسة الكهربائية).يجدر بنا أن نعرف أن عالما يحتوي على n من الغرف سيحتوي على $n \cdot 2^n$ من الحالات.
2) تصلح أي حالة من الحالات الثمان **كحالة ابتدائية.**
3) إن الشكل 2 يوضح علاقة كل حدث (Right, Left, Suck) من أحداث الوكيل الذكي بالحالة التي سينتقل إليها بسبب ذلك الحدث
4) **بلوغ الهدف** : سنقول أننا وصلنا إلى الهدف ، إذا وجدنا أن كلي المربعين نظيفان.
5) وسنتفق على أن تكلفة كل مرحلة تساوي 1.

مثال 2

في هذا المثال سنتناول (The 8-puzzle) والشكل Figure 3 يوضح ماهية المشكلة.

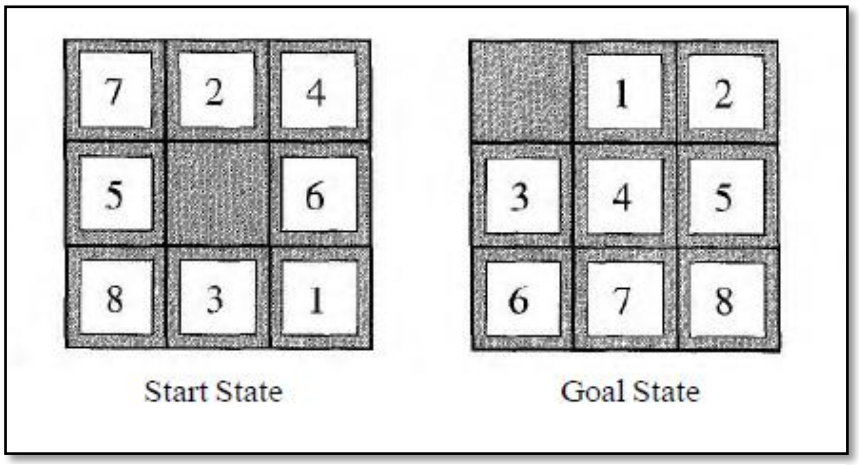


Figure2

تتألف **The 8-Puzzle** من رقعة 3×3 تحتوي على مربعات مرقمة من 1 إلى 8 ومربع خالي .فقط المربعات المجاورة للمربع الخالي هي التي تستطيع إلى هذه المنطقة مخلفة وراءها مربعا خاليا (انظر الشكلFigure2).

1) **الحالات**: يمكن أن نعرف الحالات بتعريفنا موقعا معينا لكل مربع من المربعات الثمانية المرقمة على الرقعة.
2) **الحالة الابتدائية** : أي تحديد من للمربعات الثمانية المرقمة يمكن أن يعتبر بداية مقبولة.
3) **دالة الوريث**: هي دالة تحدد الحالة التي سيؤول إليها الوضع بعد تطبيق احد الأحداث الممكنة من الوضع الحالي وهي (Up, Right, Left, Down).
4) بلوغ الهدف: لابد من وجود وسيلة تقارن الحالة الراهنة مع الحالة الهدفية (قد تكون هناك حالات هدفية غير تلك الموضحة في Figure2) لتحديد بلوغ الهدف من عدمه.في حالتنا هذه سنقارن كل مربع في الحالة الراهنة مع المربع المناظر في الحالة الهدفية.
5) **تكلفة المسار**: سنعتبر تكلفة الانتقال من حالة إلى أخرى مساوية للواحد، لذا فإن تكلفة المسار تساوي عدد الخطوات التي قمنا بها.

مثال 3

في هذا المثال سنتناول **8-Queens Problem** وهي مسالة وضع 8 وزراء في رقعة شطرنج. تتميز هذه المشكلة بان حالاتها تمتع بصياغة تزايدية (incremental formulation). نعني بالصياغة التزايدية زيادة العناصر اللازمة لوصف حالة ابتداء من حالة فارغة. من اجل مسألة **8-Queens Problem** هذا يعني أن كل حدث سيؤدي إلى زيادة عدد الوزراء في الرقعة بمقدار 1. **يمكن أن نستخدم صياغة الحالة التامة**

(Complete-State Formulation) وفيها نضع 8 وزراء بحيث يشغل كل وزير صفا وعمودا مختلفا ونحركهم حتى نصل إلى حالة لا يهاجم فيها أي وزير وزيراً آخر. من اجل الصيغة التزايدية للمشكلة سنقدم الوصف التالي:

1) **الحالات**: أي ترتيب لأي عدد من الوزراء على الرقعة من0 إلى 8 .
2) **الحالة الابتدائية**: رقعة فارغة.
3) **دالة الوريث**: وضع وزير على خلية فارغة من الرقعة.
4) **بلوغ الهدف:** سنقول إننا بلغنا الهدف إذا امتلكنا8 وزراء لا يهاجم بعضهم بعضا.

ان هذه الصياغة تجعلنا نقوم بفحص عدد كبير من الحالات التي يظهر للعيان عدم ملائمتها لذلك سنقوم بتغيير البندين التاليين:

1) **الحالات**:كل الحالات التي يحتوي فيها كل صف وكل عمودا وزيرا.
2) **دالة الوريث**:وضع الوزير على خلية فارغة وغير مهاجمة من وزير آخر.

البحث عن حلول SEA.RCHING FOR SOLUTIONS

بصياغتنا للمشاكل ينبغي حلها.نستطيع الوصول إلى الحل بالبحث داخل فضاء الحالات (State Space).ان تفعيل دالة الوريث مع الحالة الابتدائية يولد لنا ما يسمى بشجرة البحث (Search Tree) . شجرة البحث هي مخطط بياني موجه يبين المسار الذي سلكناه للوصول إلى حالة معينة. نسمي كل عقدة من عقدة الشجرة بعقد بحث (Search node). وتمثل كل عقدة من عقد الشجرة حالة من حالات فضاء البحث، إضافة إلى مؤشر إلى الحالة السابقة التي أوصلتنا إلى هذه الحالة، وتكلفة الوصول إلى هذه الحالة انطلاقا من الحالة السابقة ،والحدث الذي أدى تطبيقه في الحالة السابقة كي نصل إلى هذه الحالة والعمق الذي تقع فيه العقدة. لاحظ انه يمكن أن نصل إلى نفس الحالة من أكثر من مسار هذا يعني وجود عقدتين تحتويان على نفس الحالة ، ولكن الحالات السابقة لهذه الحالة مختلفة .هذا يعني ان شجرة البحث قد تستمر في التوسع إلى مالا نهاية في حين إن فضاء الحالات منتهي.

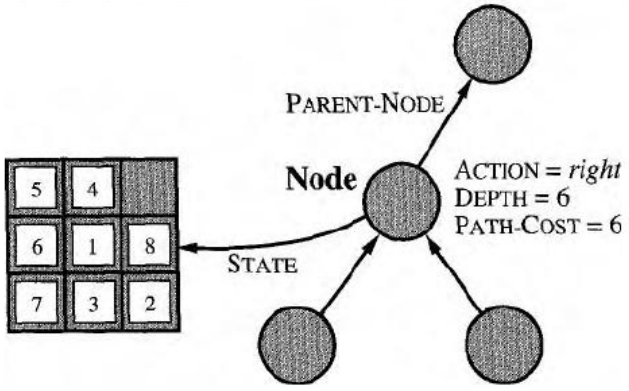


Figure3

أن تطبيق دالة الوريث على عقدة معينة يطلق عليه توسيع العقدة **(Expanding)** ونتيجة لهذه العملية يحدث توليد (**Generating**) أبناء من هذه العقدة. ففي مثال السائح نتيجة لتوسيع العقدة **Arad** نتجت العقد **Zerind** و **Timisoara** و **Sibiu** (أنظر الشكل Figure1) وسنرى ما هي العقدة التالية التي سنختارها لمواصلة البحث. إذا فرضنا أننا اخترنا العقدة **Sibiu** وطبقنا دالة الوريث عليها فسنحصل على **Arad** و**Fagaras** و **Oradea** و **RimnicuVilcea** (دالة الوريث هنا تمثل المدن المتصلة بالمدينة الحالية). قد نختار عقدة من بين هذه الأربع الجديدة أو عقدة من الثلاث السابقات. أن الطريقة التي نحدد بها اختيار العقدة التالية تسمى إستراتيجية البحث.

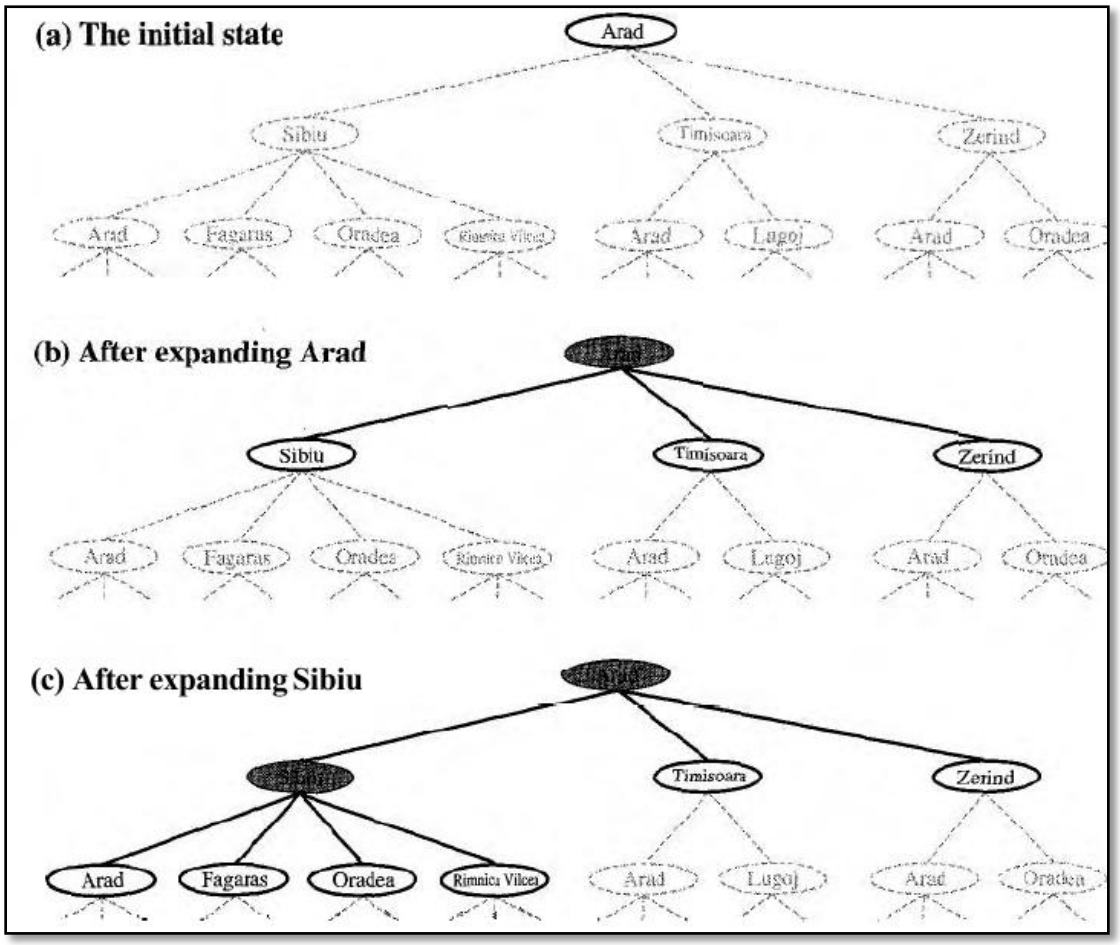


Figure4

لاحظ ظهور العقدة Arad من جديد ، ولو قمنا باختيار العقدة Arad ثم قمنا بتوسيعها فستتولد نفس العقد التي أنشئت سابقا ومن ثم استمرار نمو الشجرة إلى مالا نهاية. وهذه شفرة زائفة تبين البحث داخل شجرة

```
function TREE-SEARCH(problem, strategy) returns a solution, or failure
  initialize the search tree using the initial state of problem
  ioop do
    if there are no candidates for expansion then return failure
    choose a leaf node for expansion according to strategy
    if the node contains a goal state then return the corresponding solution
    else expand the node and add the resulting nodes to the search tree
```

تقول الشفرة السابقة مايلي:

1) تهيئة الشجرة بتحديد الحالة الابتدائية (جذر الشجرة).
2) استمر في عمل التالي
   a. إذا لم يكن هناك عقد لتوسيعها معناها أن الخوارزمية فشلت في الوصول إلى هدف أعدا النتيجة (فشل)
   b. اختر عقدة طرفيه كي نوسعها (الاختيار حسب الإستراتيجية)
   c. إذا شكلت العقدة هدفا إذا أعد الحل الملائم لهذا الهدف
   d. إذا لم يتحقق الشرط في الخطوة c وسع العقدة التي اخترتها وضم العقد الناتجة إلى شجرة البحث

تجنب الحالات المتكررة Avoiding Repeated States

تحدثنا إلى الآن عن البحث ولم نذكر كيف يمكن تجنب الحالات المتكررة (كمار رأينا في الفقرة السابقة توليد العقدةArad مرة أخرى).إن فضاء الحالات لبعض المشاكل لا يسمح بتكرار الحالات، لأن فضاء الحالات أصلا يمثل شجرة وهذا وجود طريق وحيد إلى كل حالة. ولما كان فضاء الحالات محدودا، فستكون شجرة البحث محدودة أيضا.مثال على هذا مشكلة الوزراء الثمانية، التي تقول صياغتها أن كل وزير جديد يوضع في أقصى اليسار في أول عمود خالي من وزراء سابقين.ولو كانت صياغة المشكلة بحيث يسمح لكل وزير جديد أن يوضع في أي عمود لكان من الممكن الوصول إلى كل حالة بواسطة $n!$ من المسارات من أجل n من الوزراء.

في الواقع لا مفر من تكرار الحالات عند حل بعض المشكلات و ينتمي إلى هذا النوع من تلك المشاكل التي تكون فيها **الأحداث قابلة للعكس**، مثل مسائل البحث عن المسار(السائح في رومانيا) أو مسألة 8-puzzle .إن شجرة البحث لهذه المشاكل لانهائية (نولد الكثير من الحالات أكثر من مرة).الجدير الذكر أن تكرار حالة معينة يعني الوصول إليها من مسارات مختلفة. أن الخوارزميات التي لا تتذكر تاريخها معرضة لتكرار هذا التاريخ. تجنبا لهذا الشيء نزود الخوارزميات ببني تحتفظ العقد التي وسّعناها يطلق عليها (**Closed List**) بينما العقد التي وجدنا ولكنها لم توسع بعد تُحفظ في قائمة تسمى (**Open List**). وعليه فإن **Tree-Search** سيعدل كالتالي:

1. **Initialize: Set Open={s},Closed={}**
2. **Fail: IF Open={} , Then Terminate With Failure**
3. **Select: Select State n From Open (By Strategy) and Save n in Closed**
4. **Terminate: if n ϵ G Terminate With Success**
5. **Expand: Generate The Successors of n using O.**
   For each Successor m Insert m in Open Only if m does not belong to **Open** and m **does not** belong to **Closed**
6. **Go To Step 2.**

حيث s هي الحالة الابتدائية ، O هي دالة الوريث ، G مجموعة الحالات التي يعني الوصول إليها وصولا إلى الهدف.

1. التهيئة : نجعل القائمة Open تحتوي فقط على الحالة الابتدائية، ونجعل Closed فارغة
2. إخفاق: إذا وجدنا أن القائمة Open خالية تنهي الخوارزمية عملها معلنة إخفاقها في إيجاد حل
3. اختيار : نختار عقدة n من القائمة Open (الاختيار حسب الإستراتيجية) ونضعها في Closed
4. التوقف بنجاح : إذا كان n ينتمي إلى G توقف الخوارزمية عملها معلنة نجاحها في الوصول إلى حل
5. توسيع العقدة : نولد كل أبناء العقدة n مستخدمة دالة الوريث

من المعلوم أن نتيجة تشغيل أي خوارزمية بحث هو الإخفاق في العثور على حل أو العثور على حل.سوف نقيم أداء كل خوارزمية بحث باستخدام المعايير التالية:

- **التمام أو الشمول (Completeness)** : هل تضمن الخوارزمية العثور على حل إذا كان هذا الحل موجودا
- **أمثلية الحل (Optimality)**: هل تجد الخوارزمية الحل الأمثل دائما
- **التعقيد الزمني (Time Complexity)** : الزمن اللازم كي تجد الخوارزمية الحل
- **التعقيد المكاني (Space Complexity)** :كم نحتاج من الذاكرة كي ننفذ البحث.

يجدر بنا أن نذكر كميات نستخدمها للتعبير التعقيدين الزمني والمكاني وهي كما يلي: معامل التفريع **(b-branching factor)** وهو أقصى عدد من الأبناء يمكن أن تملكه كل عقدة. **(d-depth of shallowest goal node)** عمق اقرب عقدة هدفية إلى الجذر أو أكثر الأهداف ضحالة. أخيرا m أقصى عمق ممكن أن نصل إليه في شجرة البحث.

سنرى في الفقرات التالية العديد من استراتيجيات البحث الأعمى.

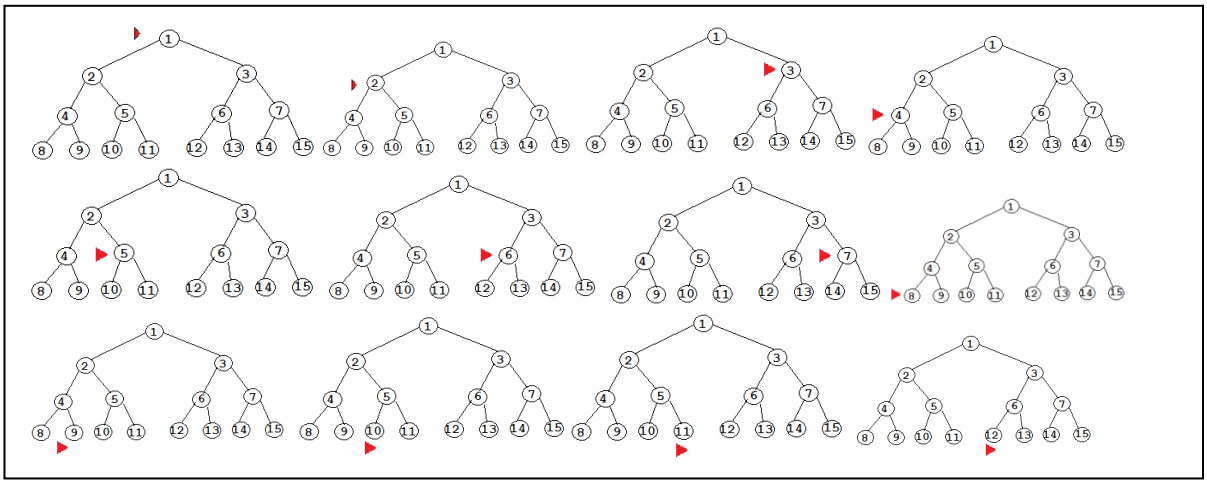


Figure5

البحث بالعرض Breadth-first search

البحث بالعرض إستراتيجية بسيطة للبحث حيث نقوم بتوسيع الجذر أولا ونضم أبناءه إلى شجرة البحث ثم نوسع الأبناء، كل على حدة ونضيف الأحفاد إلى شجرة البحث، فإذا انتهينا من توسيع الأبناء نوسع الأحفاد فإذا انتهينا نوسع أبناء الأحفاد وهكذا (أنظر Figure 5).وبشكل عام قبل الانتقال إلى عقد مستوى معين يجب توسيع كل العقد في المستوى الذي يسبقه.**لاحظ أن الاختيار من القائمة Open يحدث حسب المبدأ الداخل الأول يخرج أولا (First In First Out FIFO)** هذا يعني أننا سنختار العقدة الأولى من Open أما العقد الجديدة فتضاف إلى مؤخرة القائمة.

انظر كيف سنقوم بالبحث باستخدام القائمتين **Open, Closed** من اجل الشجرة في (Figure 5) وسنعتبر الهدف هو العقدة **12**

Closed	Open
1	1
1,2	2,3
1,2,3	3,4,5
1,2,3,4	4,5,6,7
1,2,3,4,5	5,6,7,8,9
1,2,3,4,5,6	6,7,8,9,10,11
1,2,3,4,5,6,7	7,8,9,10,11,12,13
1,2,3,4,5,6,7,8	8,9,10,11,12,13,14,15
1,2,3,4,5,6,7,8,9	9,10,11,12,13,14,15
1,2,3,4,5,6,7,8,9,10	10,11,12,13,14,15
1,2,3,4,5,6,7,8,9,10,11	11,12,13,14,15
1,2,3,4,5,6,7,8,9,10,11,12	12,13,14,15

سنقوم بتقييم البحث بالعرض باستخدام المعايير الأربعة سابقة الذكر: من الواضح أن هذه الخوارزمية**تامة (شاملة** complete) ، فلو كانت عقدة هدفية موجودة على عمق معين d فإن الخوارزمية ستسمح بالوصول إلى هذه العقدة، عندما تنتهي من فحص كل العقد في الطبقة التي تعلوها مباشرة. إن البحث بالعرض يؤدي إلى **حل امثل** إذا كانت تكلفات الانتقال من عقدة إلى أخرى موحدة (أو متزايدة كلما اتجهنا إلى العمق) بين جميع العقدة.يجدر بنا أن اختيار العقدة الهدفية الأقرب إلى الجذر لا تمثل دائما حلا امثلا ومع ذلك**فالبحث بالعرض يقدم حلا امثلا** معظم الأحيان.

سنبين كيف نحسب التعقيد الزمني كما يلي: لاحظ أننا لا ننتقل إلى أي مستوى حتى نمر بكل العقد في المستوى الذي يسبقه ، بناءً على هذا سنبدأ من الجذر ثم إلى المستوى الذي يليه (المستوى الأول) الذي يحتوي على b عقدة (لان معامل التفريع من اجل عقدة الجذر يساوي) تحتوي كل عقدة منها على b عقدة ابن لهذا فإن المستوى الثاني يحتوي على b^2 عقدة، تحتوي كل عقدة منها على b ابن ولهذا فإن المستوى الثالث يحتوي على b^3

فإذا كان عقدة هدفية واقعة على عمق d فإننا نحتاج في أسوأ الأحوال إلى المرور بعدد $b + b^2 + b^3 + \cdots + b^d = O(b^{d+1})$ من العقد. يجب أن نحتفظ بكل العقد التي مررنا بها لأنه قد تمثلا حلا أو عقدة أب لحل. وهكذا فإن التعقيد المكاني أيضا تعقيد من الدرجة$O(b^{d+1})$. أهم مشكلة في تطبيق البحث بالعرض هي توفير ذاكرة لإشباع الإنفاق الشره المستخدم في تخزين العقد.

البحث بالتكلفة الموحدة Uniform-cost search

إن البحث بالعرض يعطي حلا أمثل إذا كانت تكلفة الانتقال من أي عقدة إلى عقدة متصلة بها موحدة على مستوى شجرة البحث. بقليل من التعديل يمكن أن نجد حلا امثلا مهما كانت تكلفة الانتقال من أي عقدة إلى عقدة . ن **إن البحث بالتكلفة الموحدة يعني أننا سنقوم باختيار العقدة ذات التكلفة الأقل**.لاحظ أن تساوي تكلفات الانتقال من أي عقدة إلى أي عقدة متصلة بها يجعل البحث بالتكلفة الموحدة يتحول إلى البحث بالعرض. عند البحث بالتكلفة لا ننظر إلى العمق الذي تقع عنده العقدة، بل إلى تكلفة الوصول إلى العقدة. إذا حدث وكانت تكلفة الانتقال من عقدة إلى أخرى صفرية فالخوارزمية مهددة بالوقوع في حلقة لانهائية.لهذا فالخوارزمية تامة(شاملة) إذا وفقط إذا كانت تكلفة الانتقال من أي عقدة إلى عقدة تكلفة أكبر من الصفر.هذا يضمن زيادة التكلفة عند الانتقال من عقدة إلى أخرى ضمن أي مسار.

هنا سنرى سردا لخطوات الخوارزمية :

1. **Initialize: Set Open={s},Closed={},Set C(s)=0**
2. **Fail: IF Open={} , Then Terminate With Failure**
3. **Select: Select The minimum Cost n From Open and Save n in Closed**
4. **Terminate: if n ϵ G Terminate With Success**
5. **Expand: Generate The Successors of n using O.**
   For each Successor m :
   IF $m \notin$ [Open U Closed] Then
   Set C(m)=C(n)+C(n,m) and Insert m in Open
   IF $m \in$ [Open U Closed] Then
   Set C(m)= min {C(m),C(n)+C(n,m)}
   IF C(m) has Decreased and $m \in Closed$ Then Move m to Open
   Insert m in Open Only if m does not belong to **Open** and m **does not** belong to **Closed**
6. **Go To Step 2.**

C(s) تعني تكلفة الانتقال إلى s (s هو الجذر أو نقطة البداية)
C(n) تعني تكلفة الوصول من البداية أو من الجذر إلى العقدة n
C(n,m) تعني تكلفة الانتقال من العقدة n إلى العقدة m

1. **التهيئة : نجعل القائمة Open تحتوي فقط على الحالة الابتدائية، ونجعل Closed فارغة ثم نجعل تكلفة الوصول إلى s تساوي**
2. **إخفاق: إذا وجدنا أن القائمة Open تنهي الخوارزمية عملها معلنة إخفاقها في إيجاد حل**
3. **اختيار : نختار عقدة n من القائمة Open ذات التكلفة الأقل ونضعها في Closed**
4. **التوقف بنجاح : إذا كان n ينتمي إلى G توقف الخوارزمية عملها معلنة نجاحها في الوصول إلى حل**
5. **توسيع العقدة : نولد كل أبناء العقدة n مستخدمة دالة الوريث (اللاحق)**
   من اجل أي ابن m من أبناء العقدة n
   إذا كان $m \notin$ [Open U Closed]
   نضع C(m)=C(n)+C(n,m) ثم نرسل الابن m إلى القائمة Open
   إذا كان $m \in$ [Open U Closed]
   نضع C(m) تأخذ صغرى القيمتين أما قيمتها السابقة C(m) أو C(n)+C(n,m)
   إذا كانت قيمة C(m) تغيرت (أصبحت اقل) وكانت $m \in Closed$ فإننا ننقلها من **Closed** إلى **Open**.
6. **عد إلى الخطوة 2**

مثال 4: جد الحل إذا علمت أن الهدف هو العقدة12

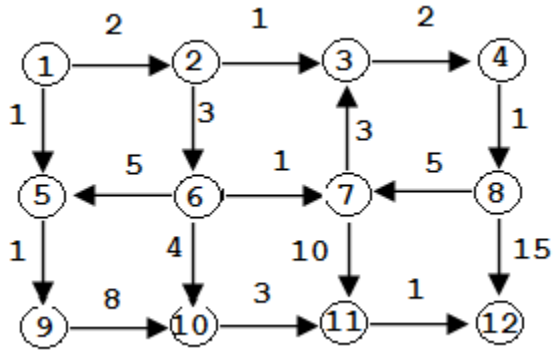


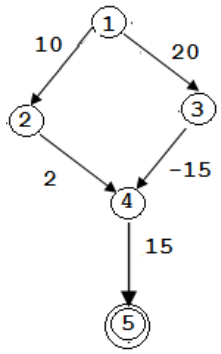
Figure6

Closed	Open
$1^{(0)}$	$1^{(0)}$
$1^{(0)}5^{(1)}$	$2^{(2)}5^{(1)}$
$1^{(0)}5^{(1)}2^{(2)}$	$2^{(2)}9^{(2)}$
$1^{(0)}5^{(1)}2^{(2)}9^{(2)}$	$9^{(2)}6^{(5)}3^{(3)}$
$1^{(0)}5^{(1)}2^{(2)}9^{(2)}3^{(3)}$	$6^{(5)}3^{(3)}10^{(10)}$
$1^{(0)}5^{(1)}2^{(2)}9^{(2)}3^{(3)}6^{(5)}$	$6^{(5)}10^{(10)}4^{(5)}$
$1^{(0)}5^{(1)}2^{(2)}9^{(2)}3^{(3)}6^{(5)}4^{(5)}$	$\cancel{10^{(10)}}4^{(5)}10^{(9)}7^{(6)}$
$1^{(0)}5^{(1)}2^{(2)}9^{(2)}3^{(3)}6^{(5)}4^{(5)}7^{(6)}$	$10^{(9)}7^{(6)}8^{(6)}$
$1^{(0)}5^{(1)}2^{(2)}9^{(2)}3^{(3)}6^{(5)}4^{(5)}7^{(6)}8^{(6)}$	$10^{(9)}8^{(6)}11^{(16)}$
$1^{(0)}5^{(1)}2^{(2)}9^{(2)}3^{(3)}6^{(5)}4^{(5)}7^{(6)}8^{(6)}10^{(9)}$	$10^{(9)}11^{(16)}12^{(23)}$
$1^{(0)}5^{(1)}2^{(2)}9^{(2)}3^{(3)}6^{(5)}4^{(5)}7^{(6)}8^{(6)}10^{(9)}11^{(12)}$	$\cancel{11^{(16)}}12^{(23)}11^{(12)}$
$1^{(0)}5^{(1)}2^{(2)}9^{(2)}3^{(3)}6^{(5)}4^{(5)}7^{(6)}8^{(6)}10^{(9)}11^{(12)}$	$\cancel{12^{(23)}}12^{(13)}$

المسار إلى الهدف 12-11-10-6-2-1 .أما سلسلة البحث عن الحل هي آخر سطر في العمود **Closed.**

لاحظ أننا غيرنا تكلفة العقد 11,12, 10 وذلك لأننا عثرنا على كل واحدة منها من مسارين مختلفين وكان المسار الثاني أحسن من الأول، لاحظ أننا عثرنا على العقدة 3 من مسار آخر (عن طريق العقدة 7) ولكننا تجاهلنا ذلك لان المسار الأول كان أفضل من الثاني. الجدير بالذكر أن العقدة لا تعود من Closed إلى Open إلا إذا كانت لدينا تكلفات سالبة (انظر المثال التالي).

مثال 5 : **: جد الحل إذا علمت أن الهدف هو العقدة5**



Closed	Open
$1^{(0)}$	$1^{(0)}$
$1^{(0)}2^{(10)}$	$2^{(10)}3^{(20)}$
$1^{(0)}2^{(10)}4^{(12)}$	$3^{(20)}4^{(12)}$
$1^{(0)}2^{(10)}4^{(12)}3^{(20)}$	$3^{(20)}5^{(27)}$
$1^{(0)}2^{(10)}4^{(12)}3^{(20)}4^{(5)}$	$4^{(5)}5^{(27)}$
$1^{(0)}2^{(10)}4^{(12)}3^{(20)}4^{(5)}5^{(20)}$	$5^{(27)}5^{(20)}$

ويكون المسار إلى الهدف هو 5-4-3-1. لاحظ عودة العقدة 4 من Closed إلى Open. يمكن ضمانة تمام (شمولية completeness) إذا كانت تكلفة الانتقال من عقدة إلى أخرى اكبر الصفر أما إذا كانت هناك تكلفات انتقال صفرية فالخوارزمية مهددة بشبح التورط في حلقات لانهائية.كون تكلفة الانتقال موجبة يضمن زيادة التكلفة عند المرور بالمسار.وهذا يعني أن الخوارزمية توسع العقد تصاعديا حسب التكلفة وهكذا فإن الحل الذي سنحصل عليه هو حل أمثل دائما. الخوارزمية تختار العقدة التالية حسب التكلفة وليس حسب العمق لذلك من الصعب التعبير عن التعقيد الزمني والمكاني باستخدام d و b (العمق d-depth وكذلك معامل التفريع b-branching factor) لذلك سنفترض أن تكلفة أي انتقال تكلف على الأقل كمية موجبةε ، فإذا كانت تكلفة الحل الأمثل هي C^* . إذا ستقوم الخوارزمية ب $\left[C^*/\varepsilon\right]$ انتقال ويكون التعقيد هو $O(b^{\left[C^*/\varepsilon\right]+1})$ وإذا كانت التكلفات متساوية فإن الخوارزمية تؤول إلى البحث بالعرض المذكور سابقا.

البحث بالعمق Depth-first search

البحث بالعمق يبدأ بالعقد الأكثر عمقا أولا حتى يصل إلى عقدة طرفية لا تملك أبناءً، فيتوقف عندها التعمق ويعود حتى يصل إلى اقرب عقدة لديها أبناء فيتعمق في أبناءها وهكذا.

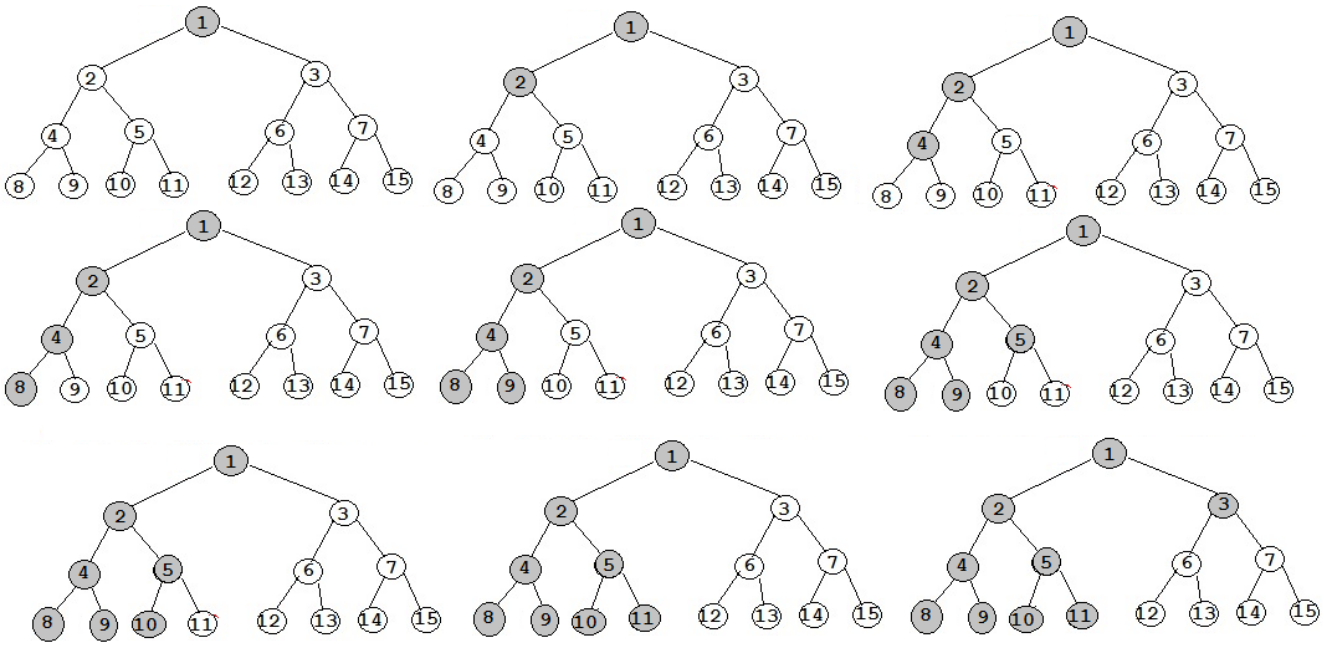


Figure7

لاحظ أننا اخترنا توسيع العقدة 4 قبل أن نختار توسيع العقدة 3 التي تقع على عمقٍ اقل،ثم اخترنا توسيع العقدة 8 (التي لم نجد أبناءً لها) قبل توسيع العقدة 5 ، ثم ارتفعنا فوجدنا أن اقرب عقدة لم توسع هي 9 فوسعناها (ولم نجد لها أبناءً لذلك انطلقنا لأعلى) ثم وسعنا العقدة 5 فوجدنا لها أبناء فانطلقنا إلى العمق ..الخ.

أن البحث بالعمق يكلف ذاكرة متواضعة مقارنة بالبحث بالعرض فهو عند أقصى عمق m وفي شجرة معامل تفريعها b يحتفظ فقط بالمسار وبأشقاء كل عقدة في المسار (طبعا الأشقاء لم يُوسعوا لذلك لا حاجة للاحتفاظ بالأبناء لأنهم لا جود لهم) أي أننا نحتفظ ب bm من العقد إضافة إلى الجذر أي ب bm+1 من العقد يعني تعقيد مكاني قيمته (O(bm،انظر الشكل Figure 8 في احد صيغ البحث بالعمق والتي يطلق عليها Backtracking Search أو التراجع لا نحتاج إلى الاحتفاظ بالأشقاء، فقط نحتاج إلى حفظ المسار أي m+1 أي تعقيد مكاني قيمته (O(m.
العيب الأساسي للبحث بالعمق هو أن الخوارزمية قد تختار مسار خاطئا أثناء البحث فتسير في العمق إلى ما لانهاية بينما الهدف يقع قريبا من الجذر في الجانب الآخر من الشجرة وهذا يعني أن الخوارزمية ليست تامة(شاملة not complete). عيب آخر البحث بالعمق (انظرالشكل Figure 7) لو أن لدينا عقدتان هدفيتان الأولى 3,11 فإن الخوارزمية ستعيد العقدة 11 دائما مع أن تكلفة الوصول إلى لعقدة 3 اقل بكثير وعلى هذا فالخوارزمية لا تقدم حلولا مثلى.وفي أسوأ الحالات تحتاج الخوارزمية إلى المرور بـ$O(b^m)$ لاحظ أن m عادة ما تكون أكبر بكثير من d.

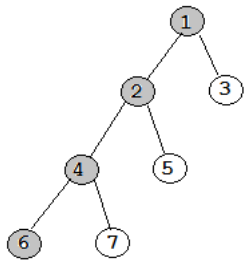


Figure8

مثال 6 : انظر إلى الشكل Figure 7 اوجد الحل من أجل العقدة 13

Closed	Open
1	1
1,2	2,3
1,2,4	4,5,3
1,2,4,8	8,9,5,3
1,2,4,8,9	9,5,3
1,2,4,8,9,5	5,3
1,2,4,8,9,5,10	10,11,3
1,2,4,8,9,5,10,11	11,3
1,2,4,8,9,5,10,11,3	3
1,2,4,8,9,5,10,11,3,6	6,7
1,2,4,8,9,5,10,11,3,6,12	12,13,7
1,2,4,8,9,5,10,11,3,6,12,13	13,7

وهكذا فإن الحل هو : 13-6-3-1. **ولا نحتاج للاحتفاظ بالقائمة Closed**

البحث بالعمق المحدد Depth-limited search

إن المشكلة التي يعاني منها البحث بالعمق (وهي التعمق في البحث إلى ما لانهاية) يمكن حلها بوضع حد أقصى للعمق ℓ، هذا يعني انه عند العمق ℓ نعامل العقد وكأنها لا تملك أبناء، وتتوقف الخوارزمية عن التعمق (إذا لم تجدا الهدف) مما يجبرها على الصعود والبحث عن الهدف في أعماق اقل من ℓ. مع أن الأسلوب الأخير يضمن الوصول إلى الهدف ولكن هناك عيب خطير يجعل الخوارزمية **لا تضمن الوصول إلى الهدف** (غير تامة) وهو أن العمق المحدد قد يكون اقل من العمق اللازم للوصول إلى الهدف $(\ell < d)$.

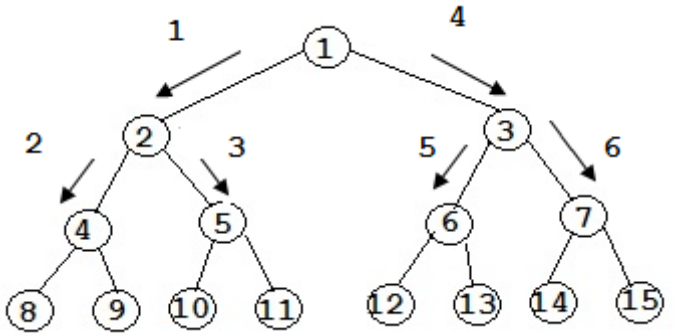


Figure9

$\ell = d$ لذلك فأقصى عمق ممكن أن نصل إليه هو2 – لاحظ أننا تجاهلنا المستوى الثالث

يجدر بنا أن أنذكر أن البحث بالعمق هو حالة خاصة من بالبحث العمق المحدد وذلك عندما $l = \infty$. ومقارنة بالبحث بالعمق فإن التعقيد الزمني $O(b^l)$ ويكون التعقيد المكاني $O(bl)$. يمكن أحيانا تعيين قيمة ℓ من فهمنا للمشكلة فمثلا في مشكلة السائح في رومانيا (التي سبق ذكرها) قد يكون أقصى عمق نصل إليه هو 19 لان عدد المدن هو 20 وبدراسة أفضل للخريطة سنجد انه من أي مدينة إلى مدينة نحتاج فقط إلى 9 مراحل. يطلق على هذا العدد **قطر(diameter)** فضاء الحالات (أقصى عدد المراحل يجب اجتيازه للانتقال من حالة إلى أخرى) .

يمكن تمثيل الخوارزمية السابقة باستخدام التعاود (recursive) كما هو مبين في الشفرة التالية:

```
Function Depth-Limited-Search(problem, limit) return Solution or Failure or Cutoff

   Return Recursive-DLS(Make-Node(Initial-State[problem]),problem, limit)

Function Recursive-DLS(node,) problem , limit) return Solution or Failure or Cutoff

 Cutoff_occurred ← false

If Goal-Test[problem](State[node]) then return Solution (node)

  else if Depth[node]=limit then return cutoff

      else for each successor in Expand(node, problem) do

          {

        Result ← Recursive-DLS(successor, problem , limit)

        If  Result= Cutoff then Cutoff_occurred ← true

          Else if Result ≠ Failure then return result

       }

 If Cutoff_occurred then return Cutoff else return Failure
```

مثال 7 : قم بإيجاد الحل في الشكل Figure9 إذا علمت أن الهدف 13 وأقصى عمق هو 2

Closed	Open
1	1
1,2	2,3
1,2,4	4,5,3
1,2,4,5	5,3
1,2,4,5,3	3
1,2,4, 5,3,6	6,7
1,2,4, 5,3,6,7	7

لم تجد الخوارزمية حلا وعليه تعلن إخفاقها في إيجاده

البحث بتعمق تكراري Iterative deepening depth-first search

يمثل البحث بتعمق تكراري إستراتيجية عامة تستخدم غالبا مع بالبحث بالعمق لتحديد أفضل حد أقصى للعمق.يتم هذا عن طريق زيادة العمق الأقصى تدريجيا إلى أن نصل إلى هدف.يستمر هذا العمق حتى نصل إلى d (العمق اللازم للوصول إلى اقرب هدف إلى الجذر).يجمع البحث بتعمق تكراري حسنات البحثين بالعرض وبالعمق.فهو كالبحث بالعمق يعطينا استهلاكا ممتازا للذاكرة $O(bd)$.وكما في البحث بالعرض الخوارزمية تامة (شاملة complete) إذا كان معامل التفريع محدودا و تعطي الحل الأمثل إذا كانت تكلفات الانتقال تزايدية (أو على الأقل متساوية).يبين الشكل Figure 10 خطوات عمل الخوارزمية إلى العمق 4 على شجرة ثنائية (معامل التفريع 2 لكل عقدة إبنان كحد أقصى).

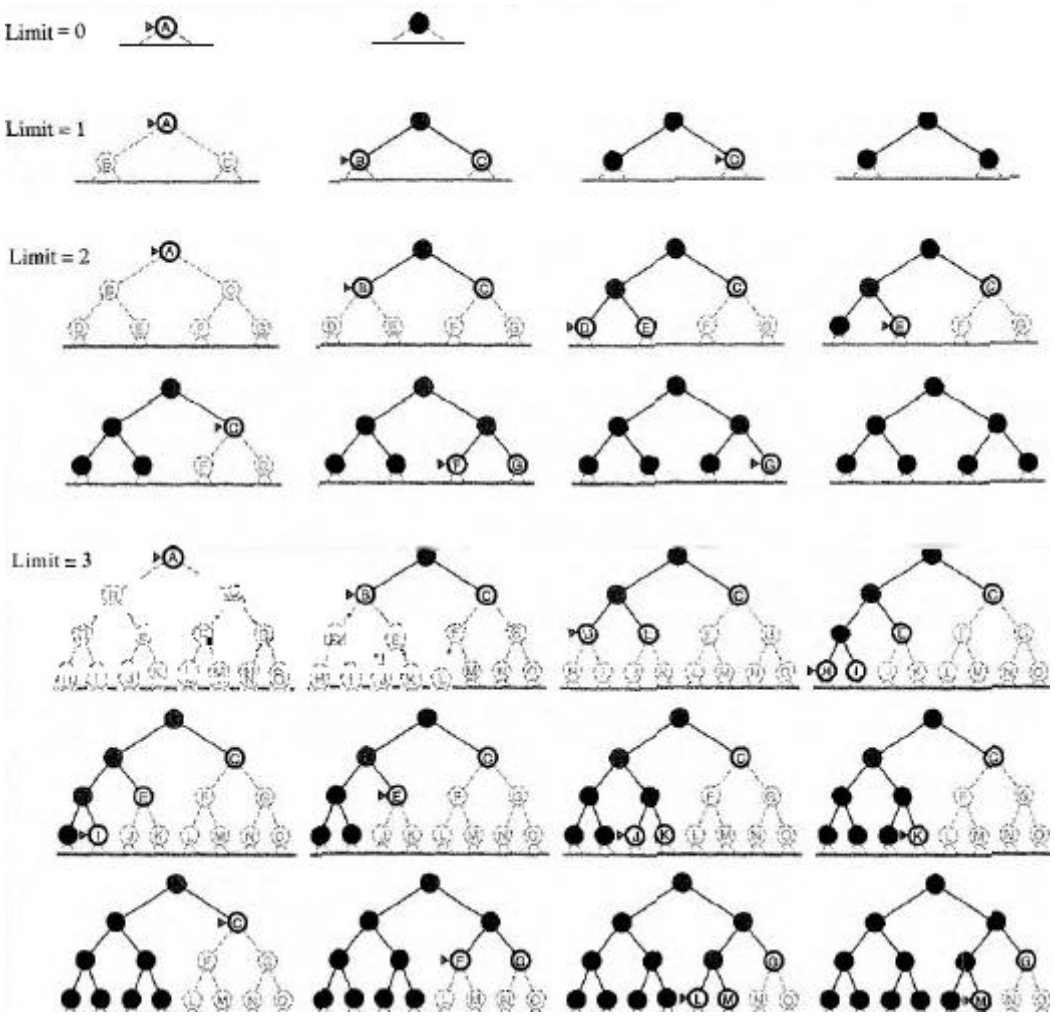


Figure10

يبدو لأول وهلة أن البحث بالتعمق التكراري مكلف لأننا نفحص نفس العقدة أكثر من مرة ولكن في الحقيقة أن التكرار ليس مكلفا جدا، ففي شجرة ذات معامل تفريع ثابت للعقدة ، تقع اغلب العقد في المستويات السفلى لذلك يمكن تجاهل تكرار فحص العقد في المستويات العليا طالما أن معظم العقد يقع أسفل الشجرة (لاحظ في الشكل Figure 10 أن المستوى الثاني يملك 4 عقد بينما مجموع المستوى الصفري والأول هو 3 عقد وفي المستوى الثالث نجد 8 عقد بينما مجموع العقد في الصفري والأول والثاني هو 7 عقد)، كما أن المستويات تتكرر بشكل مختلف.

$$N(Iterative\ deepening\ depth-first\ search) =$$

$$b^1 + (b^1 + b^2) + (b^1 + b^2 + b^3) + \cdots + (b^1 + b^2 + .. + b^d) =$$

$$= b^1 d + b^2 (d-1) + \cdots + b^d (1)$$

وهذا التعقيد الزمني يمكن مقارنته مع التعقيد الزمني للبحث بالعرض انظر:

$$N(\text{Breadth} - \text{first search}) = b^1 + b^2 + .. + b^d$$

الجدير انه في البحث بالعرض قد نقوم بإضافة عقد من المستوي d+1 (رغم أن الهدف يقع في العمق d). ولكن هذا لا يحدث أبدا في البحث التكراري لأننا نفحص العمق (بواسطة الإجراء Depth-Limited-Search) قبل إنشاء أبناء جدد، مما يعني حقيقة أن البحث التكراري أسرع من بالعرض بغض النظر عن الفحص المتكرر للعقد في البحث التكراري. الشفرة التالية تبين عمل البحث التكراري

Function Iterative-Deepening-Search**(** problem**) return solution, or failure**

inputs: problem, a problem

$for\ \ depth \leftarrow 0\ to \infty\ do$

$result \leftarrow \ \text{Depth} - \text{Limited} - \text{Search}(\text{problem}, depth)$

$if\ result \neq Cutoff\ \boldsymbol{then\ return}\ result$

عموما يمكن القول أن البحث بتعمق تكراري هو **البحث المفضل** عندما نبحث بحثا أعمى إذا كان فضاء البحث كبيرا وعمق الهدف غير معلوم. كذلك مثل البحث بالعرض فإن البحث بالتعمق التكراري لا ينتقل إلى فحص عقد المستوى التالي إلا إذا فحص كل العقد في المستويات السابقة.

مثال 8 : بين كيف سنصل إلى الحل في الشجرة الموضحة في الشكل Figure9 باستخدام البحث بالتعمق التكراري إذا كان الهدف هو 15:

Closed	Open
1	1
1	1
1,2	2,3
1,2,3	3
1	1
1,2	2,3
1,2,4	4,5,3

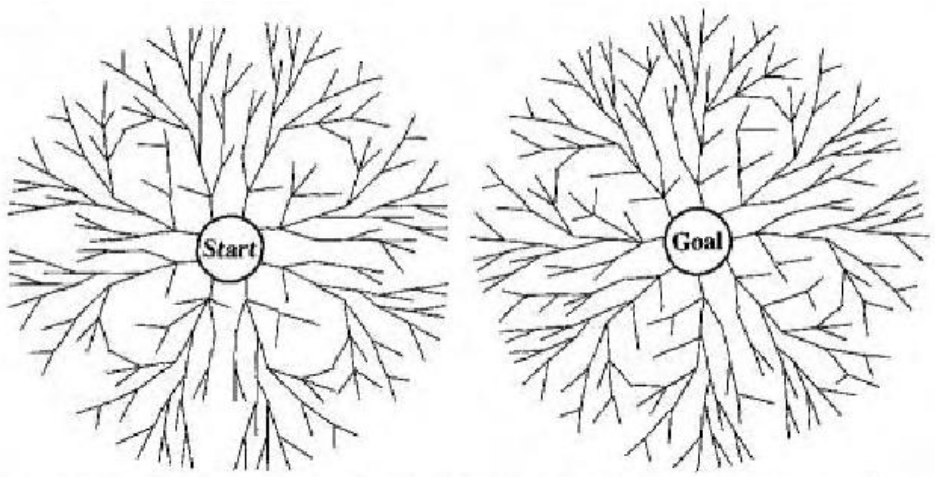
Closed	Open
1,2,4,5	5,3
1,2,4,5,3	3
1,2,4,5,3,6	6,7
1,2,4,5,3,6,7	7
1	1
1,2	2,3
1,2,4	4,5,3
1,2,4,8	8,9,5,3
1,2,4,8,9	9,5,3
1,2,4,8,9,5	5,3
1,2,4,8,9,5,10	10,11,3
1,2,4,8,9,5,10,11	11,3
1,2,4,8,9,5,10,11,3	3
1,2,4,8,9,5,10,11,3,6	6,7
1,2,4,8,9,5,10,11,3,6,12	12,13,7
1,2,4,8,9,5,10,11,3,6,12,13	13,7
1,2,4,8,9,5,10,11,3,6,12,13,7	7
1,2,4,8,9,5,10,11,3,6,12,13,7,14	14,15
1,2,4,8,9,5,10,11,3,6,12,13,7,14,15	15

من نافلة القول أننا لا نحتفظ بمحتويات القائمة Closed.

البحث ثنائي الاتجاه Bidirectional search

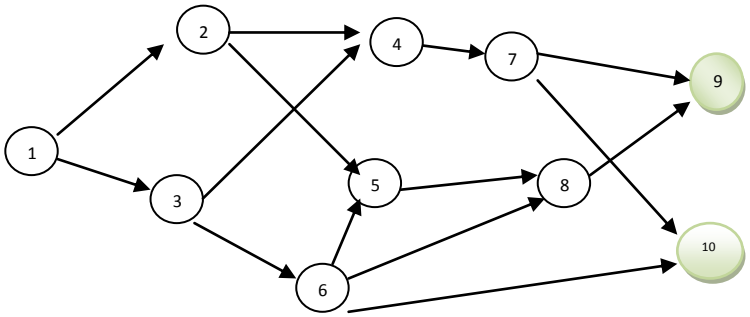
أن الفكرة الأساسية الكامنة وراء البحث هي انه يمكن إجراء بحثين معا (بالاتجاه الأمامي من الحالة الابتدائية حتى الهدف، والعكس من الهدف وحتى البداية). فإذا كان الهدف على عمق d والتقى البحثان الأمامي والخلفي في المنتصف فإن التعقيد الزمني هو $b^{d/2} + b^{d/2}$ وهذه النتيجة أفضل بكثير من b^d و كما هو موضح من الشكل فإن مساحة دائرتين صغيرتين اقل من مساحة دائرة كبيرة مركزها هو الحالة الابتدائية وتقع الحالة النهائية داخل محيط الدائرة.

Figure11

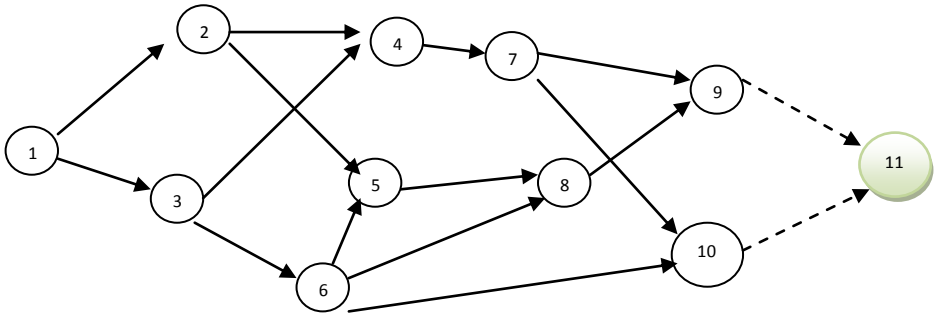


ينفذ البحث ثنائي الاتجاه عن طريق تكوين عقد بواسطة البحث في الاتجاهين، ثم وجود آلية تفحص فيما إذا كانت العقدة الجديدة موجودة كطرف (fringe) في الجهة الأخرى أم غير موجودة قبل أن توسيعها (نستخدم البحث بالعرض لإجراء البحث بالاتجاهين). في حالة الحصول على جواب إيجابي فهذا يعني الوصول إلى الهدف.إن اختبار وجود عقدة ناتجة من احد البحثين في الجهة الأخرى (نتائج البحث الآخر) تجري عن طريق

جداول تجزئة (Hash Table).لذلك فعملية البحث تكلفا زمنا ثابتا (التعقيد الزمني الثابت يمكن تجاهله!لماذا؟) وهكذا فإن التعقيد الزمني يساوي $b^{d/2}$.فيما يخص التعقيد المكاني فيجب الاحتفاظ بأحد الشجرتين على الأقل لإختبار انتماء عقد البحث الثاني لهذه الشجرة، وهكذا فإن التعقيد المكاني يساوي أيضا $b^{d/2}$. هذا الشره في إنفاق الذاكرة هو أكبر نقاط ضعف البحث ثنائي الاتجاه. الخوارزمية تامة (شاملة (Complete وتعطي الحل الأمثل Optimal إذا كانت تكلفات الانتقال موحدة وكانت عملية البحث الاتجاهين تستخدم البحث بالعرض أما طرق البحث الأخرى فتعاني من عدم التمام أو غياب الحلول المثلي أو من الاثنين معا.بسبب هذا التحسين في زمن الخوارزمية فإن البحث ثنائي الاتجاه يبدو جذابا ولكن المشكلة هي كيف ننظم بحثا في الاتجاه المعاكس (أي من الهدف إلى البداية).سوف نعرف أسلاف predecessors العقدة n على أنها تلك العقد التي تؤدي إلى توليد العقدة n بواسطة الدالة Pred(n). واضح أن البحث ثنائي الاتجاه يتطلب ن تكون الدالة Pred فعالة وابسط الأوضاع هو ذلك الوضع الذي تكون فيه الأحداث اللازمة للانتقال من عقدة إلى أخرى قابلة للعكس (مثلا التحريك لأعلى أو لأسفل في 8-puzzle).إذا كان فضاء البحث يحتوي على أكثر من هدف يمكن إنشاء حالة وهمية وجعل جميع العقد الهدفية إسلافا لها (بعبارة أخرى نعامل جميع العقد الهدفية كأنها عقدة هدفية وحيدة). انظر الشكل بالاسفل



فضاء عينة يحتوي على أكثر من هدف



فضاء عينة يحتوي على أكثر من هدف

Criterion	Breadth-First	Uniform-Cost	Depth-First	Depth-Limited	Iterative Deepening	Bidirectional (if applicable)
Complete?	Yes[a]	Yes[a,b]	No	No	Yes[a]	Yes[a,d]
Time	$O(b^{d+1})$	$O(b^{1+\lfloor C^*/\epsilon \rfloor})$	$O(b^m)$	$O(b^\ell)$	$O(b^d)$	$O(b^{d/2})$
Space	$O(b^{d+1})$	$O(b^{1+\lfloor C^*/\epsilon \rfloor})$	$O(bm)$	$O(b\ell)$	$O(bd)$	$O(b^{d/2})$
Optimal?	Yes[c]	Yes	No	No	Yes[c]	Yes[c,d]

Evaluation of search strategies. b - is the branching factor; d - is the depth of the shallowest solution; m is the maximum depth of the search tree; l is the depth limit. Superscript caveats are as follows: a complete if b is finite; b complete if step costs ≥ ε for positive ε ; c optimal if step costs are all identical; d if both directions use breadth-first search