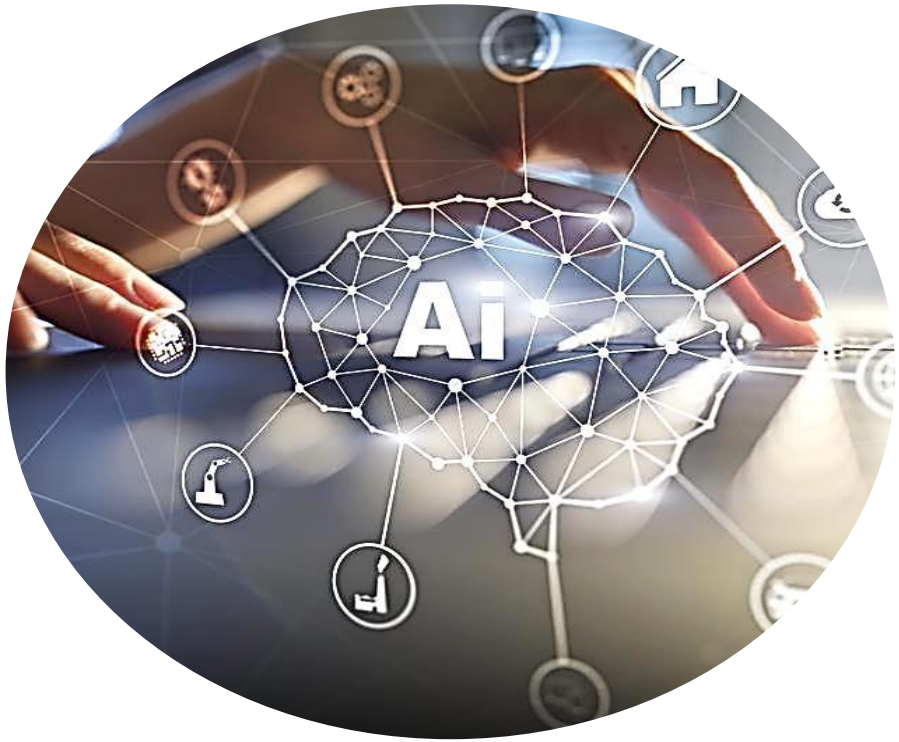
مقدمة في الذكاء الصنعي

الهندسة المعلوماتية



مدرس المقرر

د.بلال شيحا

المحاضرة الخامسة

تصنيفات البحث

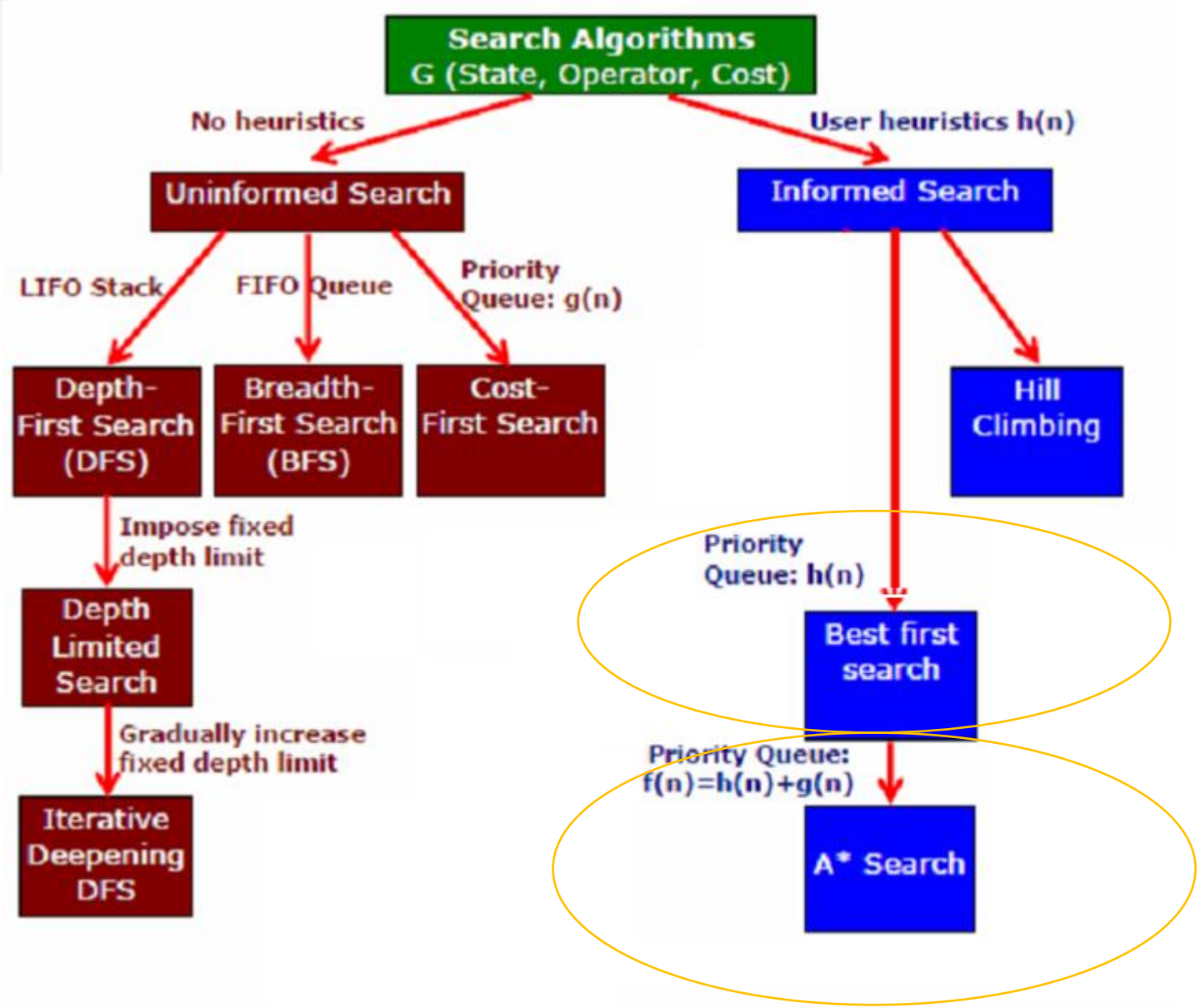
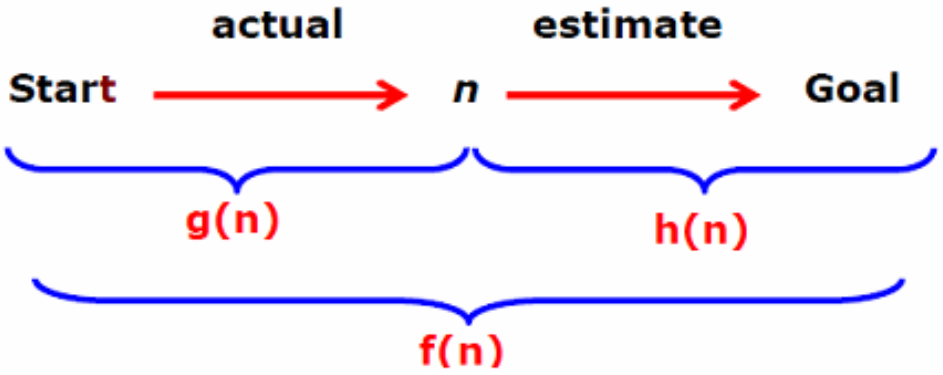


Fig. Different Search Algorithms

مقارنة

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



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

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

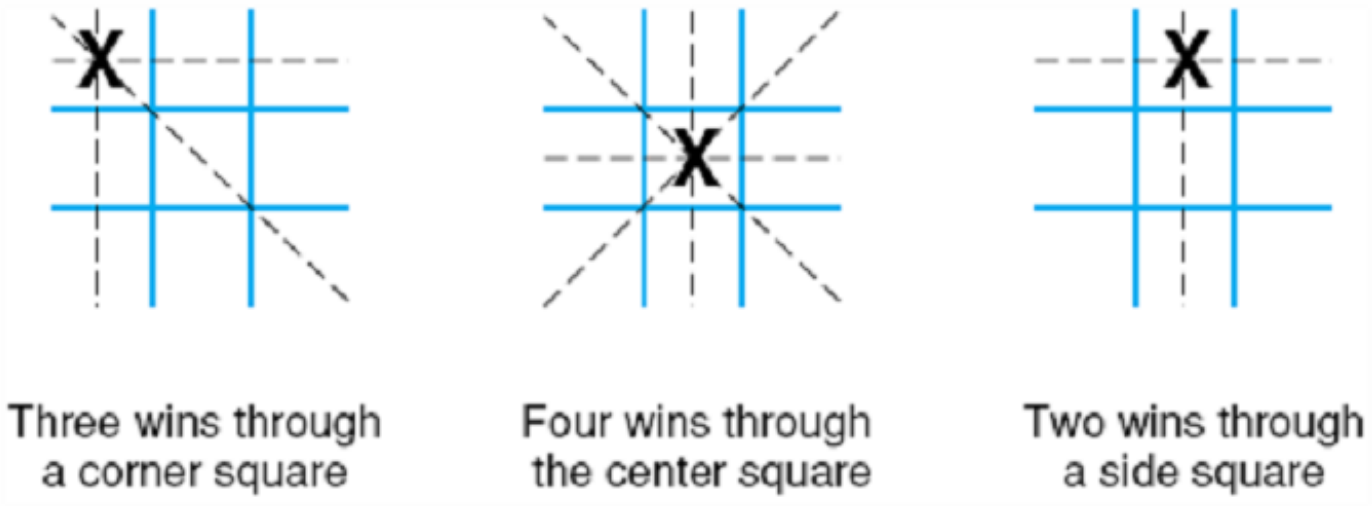
البحث بأفضل أول البسيط
(Hill climbing تسلق الهضبة)
greedy local search :

- يعتبر تسلق الهضبة أحد أبسط استراتيجيات البحث الاجتهادي. تعتمد هذه الاستراتيجية على توليد أبناء العقد و تقويم أبنائها، و انتقاء الابن الأفضل لتكرار عملية التوليد دون أخذ بقية العقد سواء كانت عقدة أب أو ابن أو عقدة أخوة أو غير ذلك بعين الاعتبار. و من الممكن الوصول إلى منطقة لا يمكن للمتسلق المتابعة بعدها وتدعى هذه النقطة قمة محلية Local Maxima و يتطلب من المتسلق في هذه الحالة الانتقال إلى طريق ممكن آخر، و يعتبر الوصول (الوقوع) في القمة المحلية أحد عيوب هذه الطريقة و بزيادة الخبرة يمكن تجنب الوصول إلى قمة محلية.
- إذ أن الخوارزمية الأولية لتسلق الهضبة لا تستطيع تذكر المراحل السابقة مما قد يؤدي إلى الحيرة و حتى إلى التوقف القسري عند الوصول إلى نقطة لا تسمح بالمتابعة لأنها تشكل تابع التقويم الأفضل، أي ليس هناك عقد ذات تابع تقويم أفضل.
- تعتبر لعبة Tic- Tac- Toe مثلا جيدا عن تسلق القمة (الهضبة)، إذ يتبع اللاعب بشكل أولي الطريق الذي يقربه من الربح دون التروي و النظر في الحالة التي سيحققها المنافس، إذ ليس المطلوب في هذه الحالة تقصي الأخطاء و الاستفادة منها.

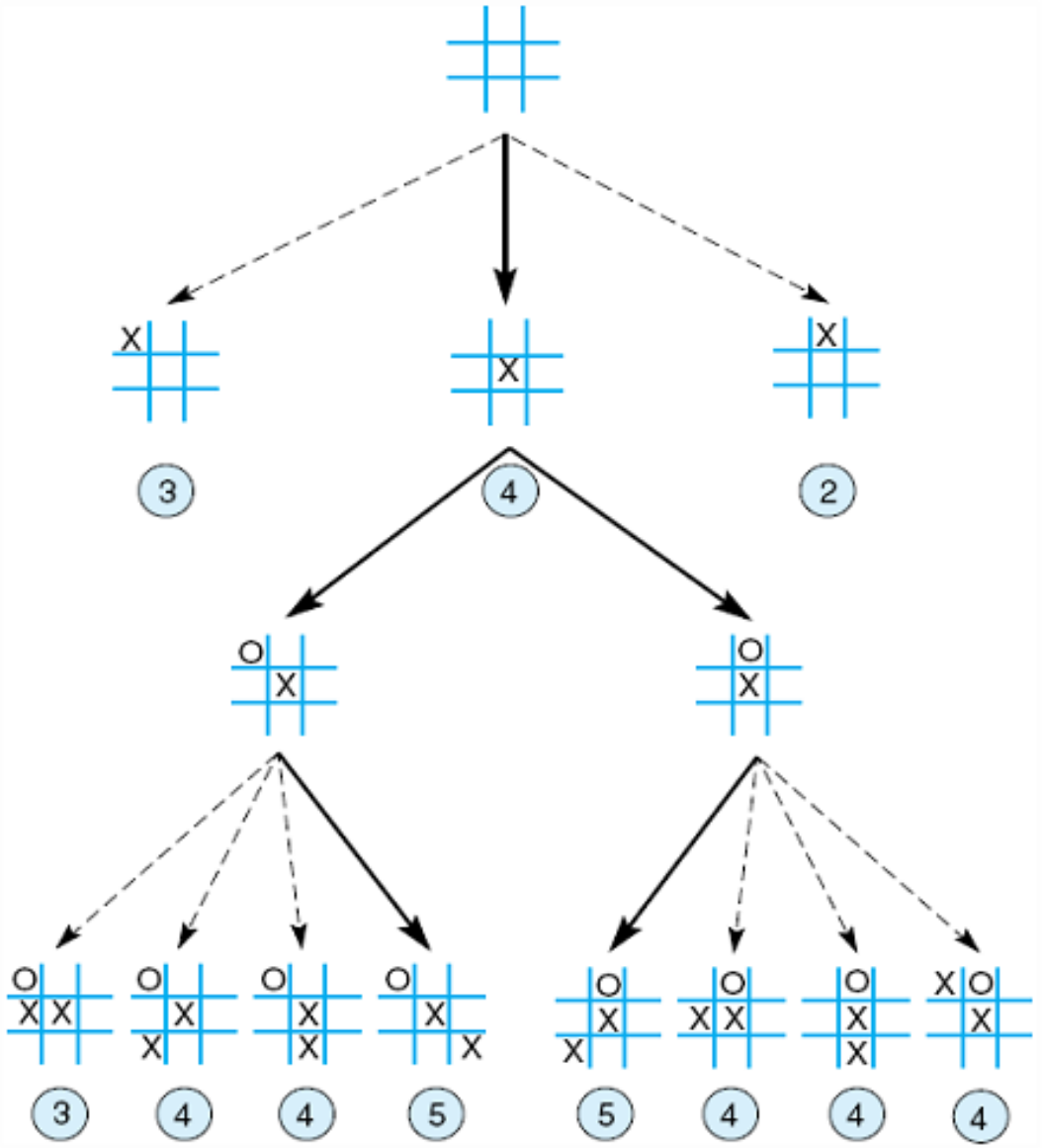
البحث بأفضل أول البسيط
(تسلق الهضبةHill climbing)
greedy local search :

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

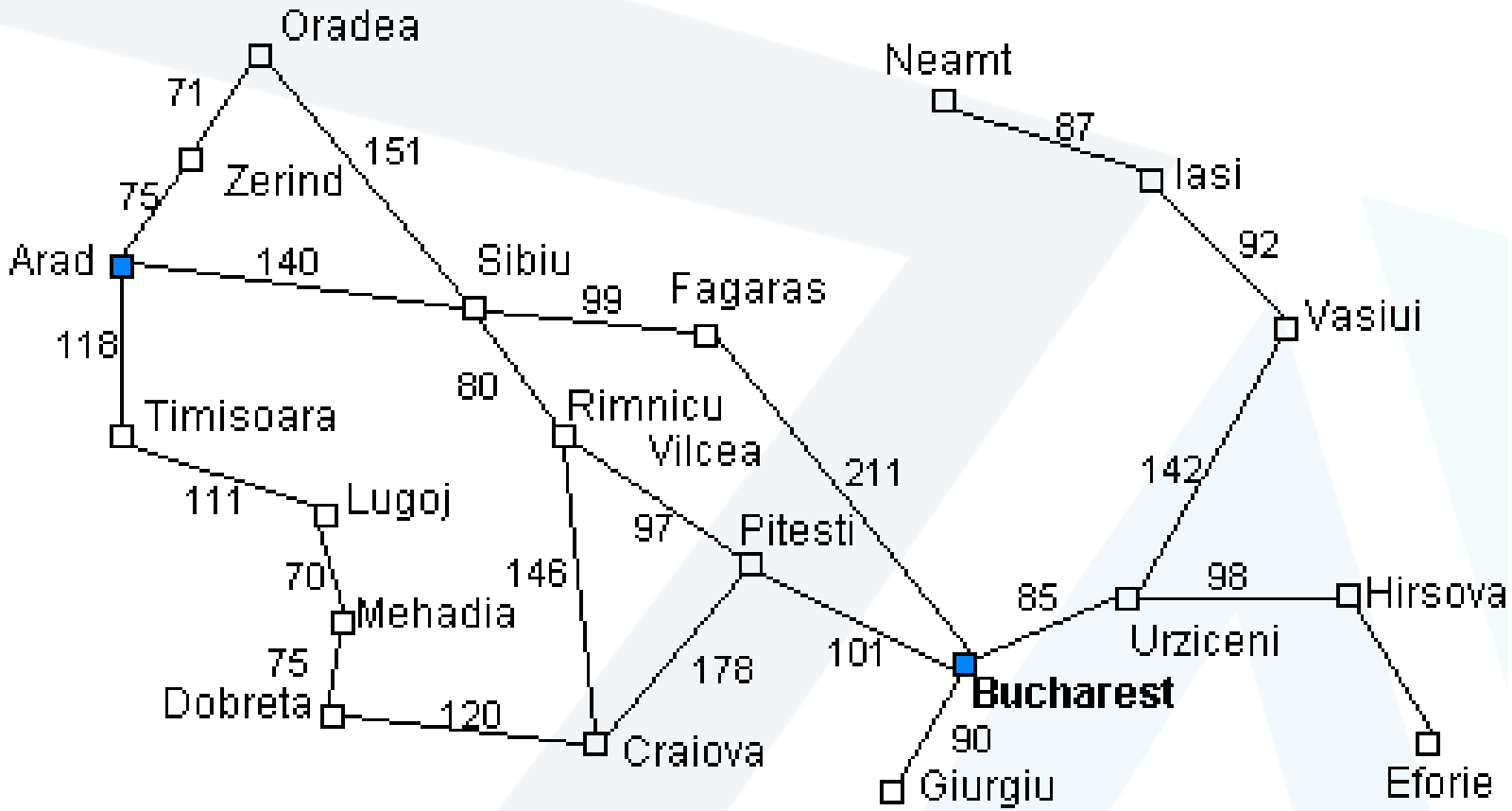
- تابع التقويم المتبع هو حساب عدد خطوط الربح و اختيار النقلة ذات عدد خطوط الربح الأكبر كما هو مبين في الشكل.



البحث بأفضل أول البسيط
(تسلق الهضبةHill climbing)
greedy local search :



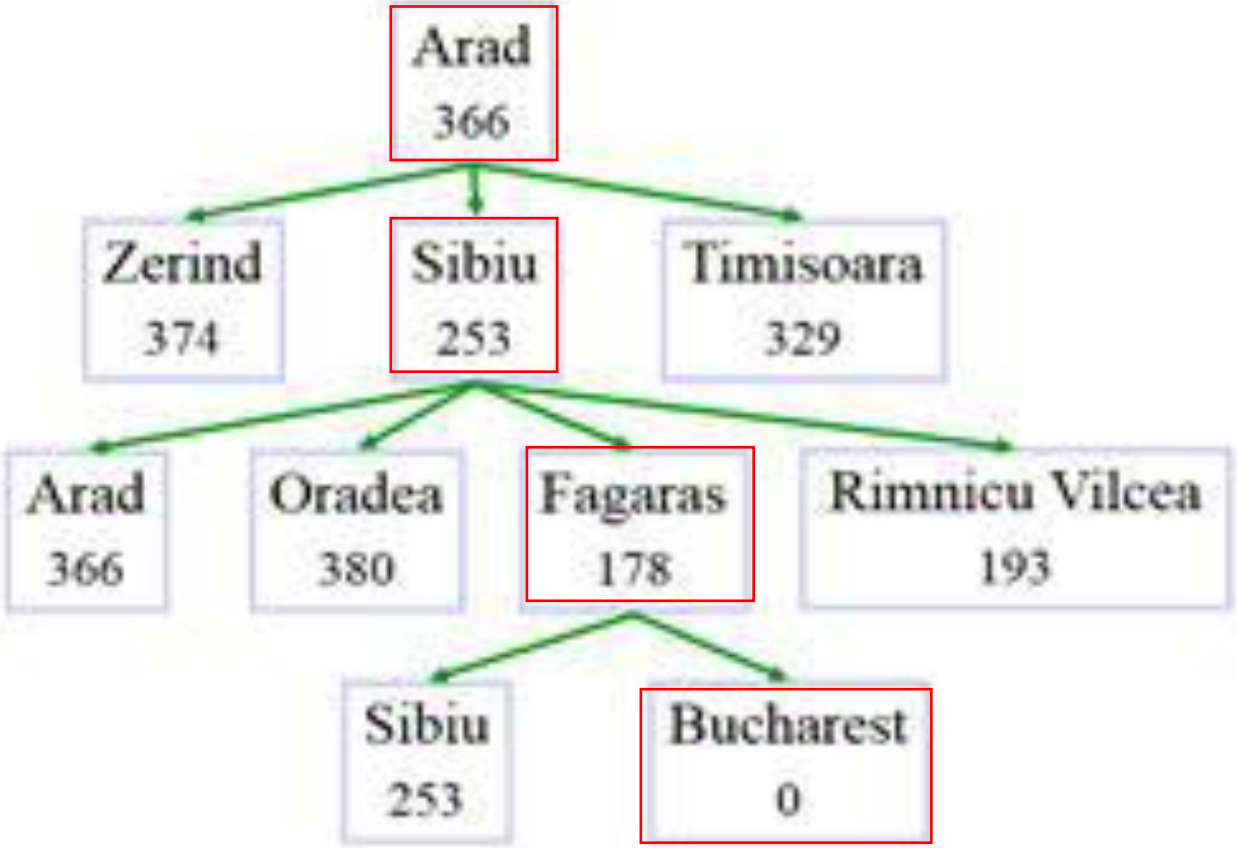
البحث بأفضل أول البسيط
(تسلق الهضبةHill climbing)
greedy local search :



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

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

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



البحث بأفضل أول البسيط
(Hill climbingتسلق الهضبة)
greedy local search :

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

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

1	2	3
4	5	6
7	8	

Goal

1	2	3
4	5	6
7		8

- 2-التجريبية الثانية

Manhattan Distance سنقوم باختيار

1	2	3
4	5	6
7		8

و هي عدد الحركات اللازمة
لإيصال كل مربع إلى مكانه
الصحيح وفق الهدف وستكون
قيمة التجريبية هي H(n)=1 للمربع التالي.

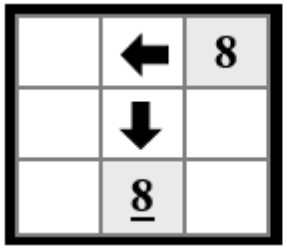
البحث بأفضل أول البسيط
(Hill climbingتسلق الهضبة)
greedy local search :

نلاحظ أنه كلاً من المربعات8 1 , 3 , ليست في مكانها الصحيح .وسنقوم بحساب عدد الحركات اللازمة لعودة كلاً من هذه المربعات لمكانها كما يلي:

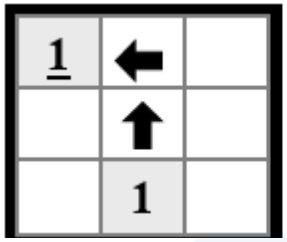
1	2	3
4	5	6
7	8	

Goal

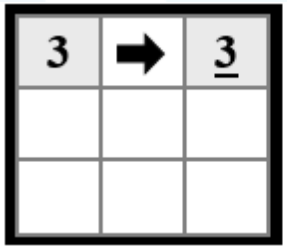
3	2	8
4	5	6
7	1	



3 spaces



3 spaces

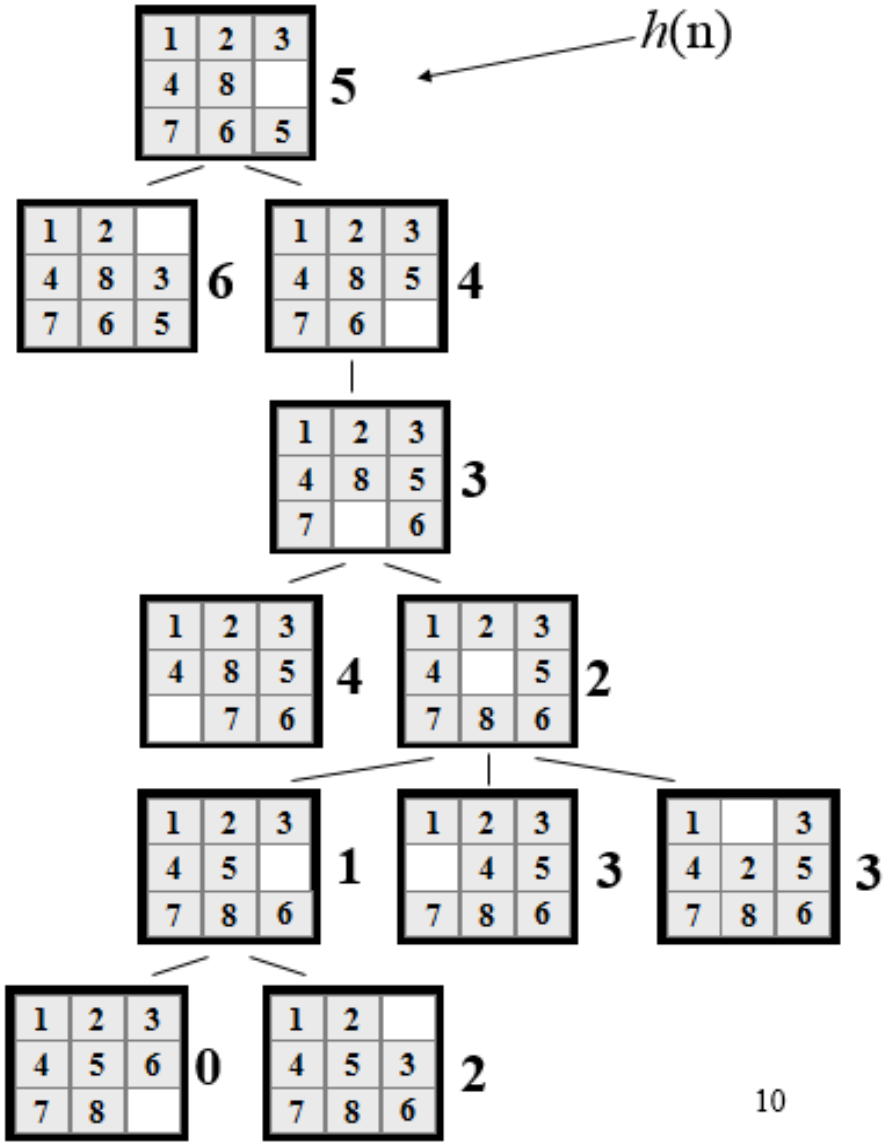


2 spaces

Total= 8 (للأشكال الثلاثة السابقة)

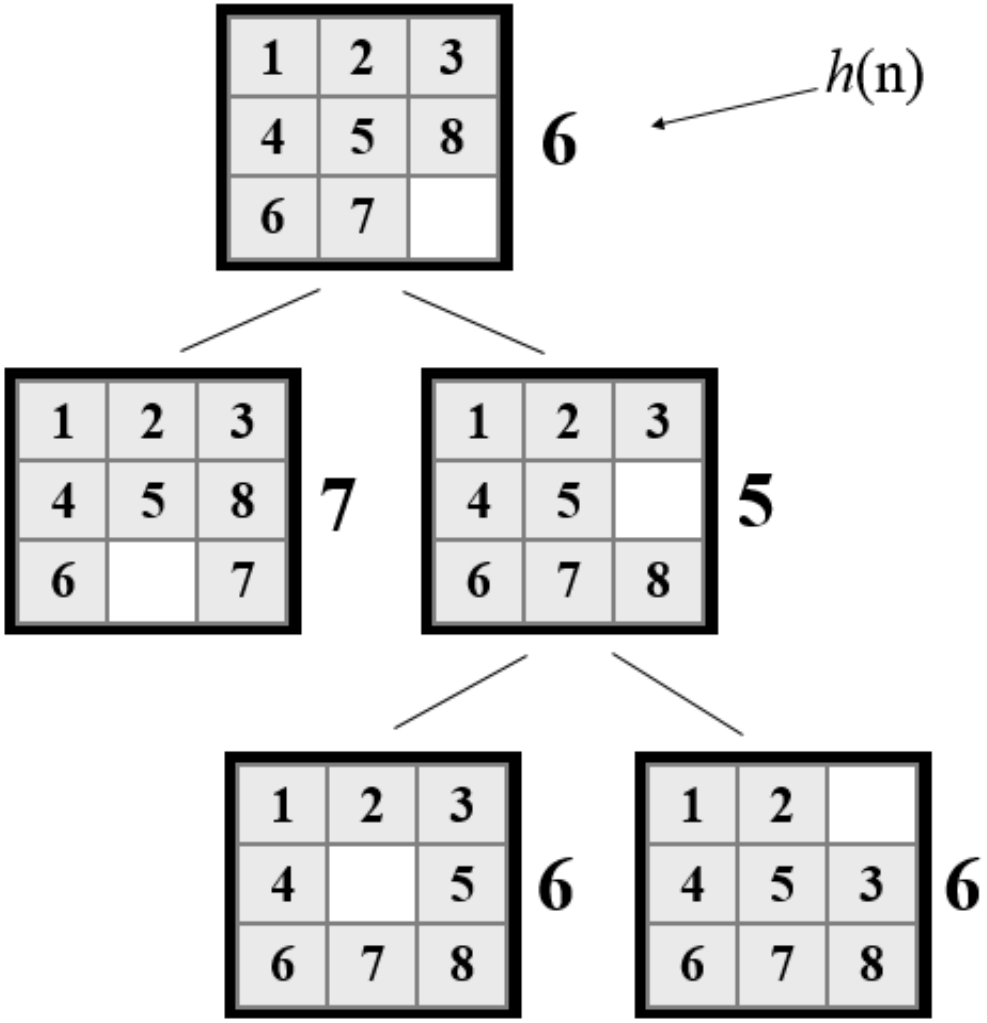
ونستطيع القول هنا أن التجريبية تفكر أنه بإمكاننا أن نصل للهدف من الخلال القيام ب 8 حركات فقط.

الحل



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

الحل



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

خوارزمية عامة للبحث في البيان
GRAPH SEARCH

١- أنشئ شجرة بحث T_r مكونة فقط من عقدة البداية n_o .

ضع n_o في قائمة مرتبة تسمى OPEN .

٢- أنشئ قائمة تسمى CLOSED تكون فارغة في البداية .

٣- إذا كانت OPEN فارغة، اخرُج بالنتيجة: إخفاق.

٤- اختر العقدة الأولى في OPEN، و احذفها من OPEN ، و ضعها في CLOSED . سم هذه العقدة n .

٥- إذا كانت n عقدة هدف، اخرج بنجاح مع الحل الذي حصلت عليه برسم مسار راجع على طول الأقواس في T_r من n إلى n_o (يتم انشاء الأقواس في الخطوة ٦).

٦- وسع العقدة n بتوليد مجموعة M من الخلَف. أرسِ M باعتبارها خلفا للعقدة n في بإنشاء الأقواس T_r من n إلى كل عنصر من M . ضع هذه العناصر من M في OPEN.

٧- أعد ترتيب القائمة OPEN إما وفق بعض المخططات الاعتباطية أو وفق استحقاق تجريبي.

٨- عد إلى الخطوة ٣ .

خوارزمية عامة للبحث في البيان
GRAPH SEARCH

يمكن استعمال الخوارزمية السابقة لتنفيذ بحث الأفضل أولا ، أو البحث عرضا أولا، أو البحث عمقا أولا. في البحث عرضا أولا نضع العقدة الجديدة في نهاية OPEN (الداخل أولا يخرج أولا أو FIFO) و لا يعاد ترتيب العقد. في البحث بطريقة العمق أولا نضع العقدة الجديدة في بداية OPEN (الداخل أخيرا يخرج أولا أو LIFO). في بحث الأفضل أولا (يسمى أيضا البحث التجريبي)، يعاد ترتيب OPEN وفق استحقاق تجريبي للعقد.

البحث بأفضل أول Best-first Search

و سنقدم فيما يلي البحث بأفضل أول بما يمكننا من تجاوز مسألة القمة المحلية Local Maxima

البحث بأفضل أول Best-first Search

نستخدم في هذه الحالة قائمتي OPEN و CLOSED كما هو عليه الحال في طرق البحث العشوائي. نضيف إلى فكرة اختيار الحل الأمثل تذكر الحالات السابقة التي تمكننا من الانتقال إلى الحالة الأفضل المتوفرة بين الحالات التي دخلت مرحلة OPEN كما يظهر من الخوارزمية التالية:

Best-first Search البحث بأفضل أول

```
function best_first_search;

begin
   open := [Start];                                              % initialize
   closed := [ ];
   while open ≠ [ ] do                                           % states remain
      begin
         remove the leftmost state from open, call it X;
         if X = goal then return the path from Start to X
         else begin
                  generate children of X;
                  for each child of X do
                  case
                        the child is not on open or closed:
                           begin
                                 assign the child a heuristic value;
                                 add the child to open
                           end;
                        the child is already on open:
                           if the child was reached by a shorter path
                           then give the state on open the shorter path
                        the child is already on closed:
                           if the child was reached by a shorter path then
                              begin
                                 remove the state from closed;
                                 add the child to open
                              end;
                  end;                                           % case
                  put X on closed;
                  re-order states on open by heuristic merit (best leftmost)
               end;
return FAIL                                                      % open is empty
end.
```

best-first search(Greedy)
البحث بأفضل أول (الجشع)

- **تعتمد استراتيجيات البحث المعرفي على أسلوب البحث الجشع**
- **يمكننا أن نعرف البحث الجشع Greedy search أنه البحث الذي يحقق العلاقة $f(n) = h(n)$.**

تعريف تابع حدسي يخمن كل مسلك واعد h(n)

انطلاقا من عقدة ابتدائية سبر كل المسالك لاختيار المسلك الواعد

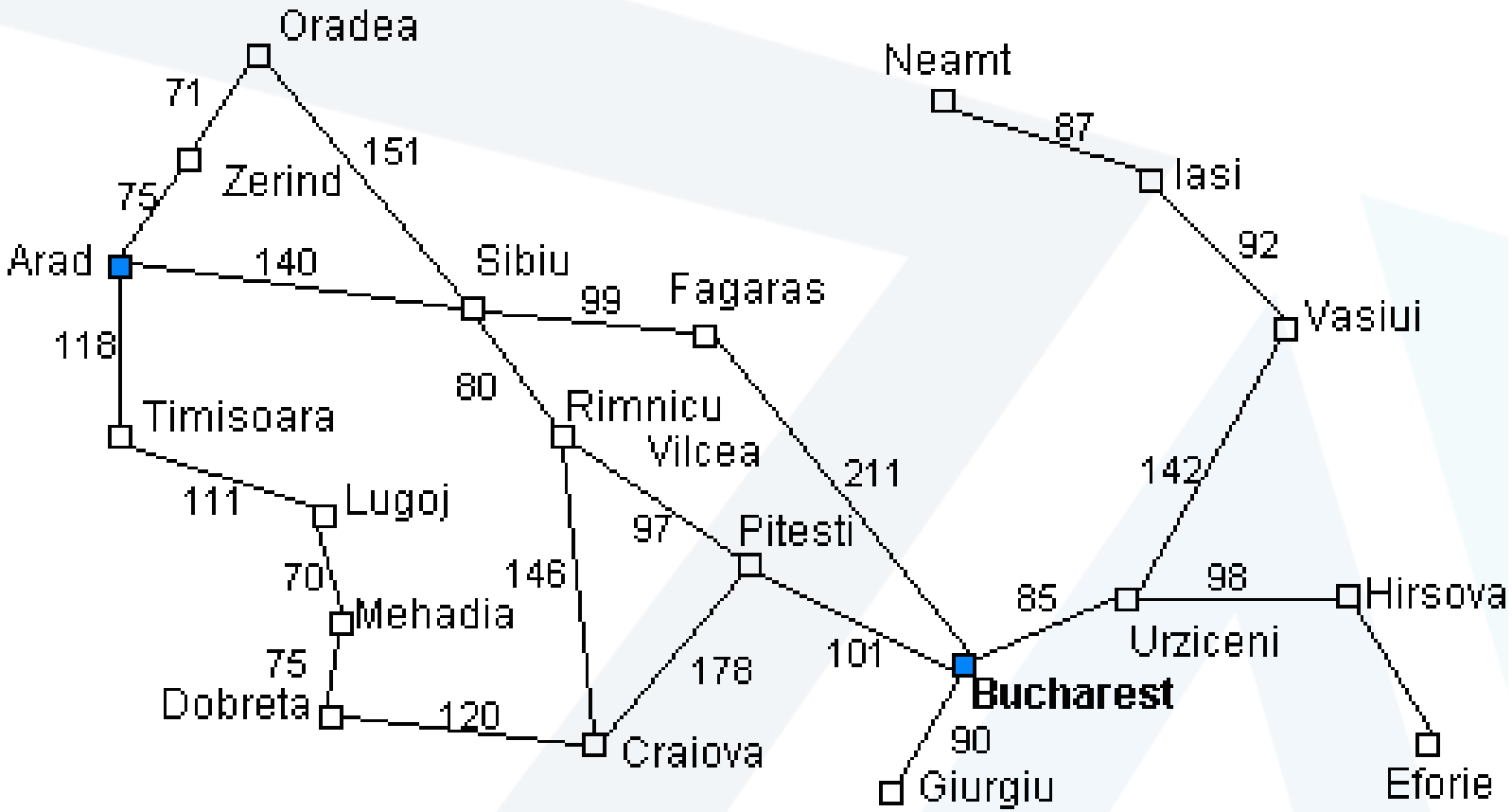
العقدة التي يبدو أنها الأكثر قربا من الهدف يكون لها أفضلية الاستكشاف

Greedy best-first search expands the node that appears to be closest to goal

Minimize the cost to reach a goal

في مثال السائح في رومانيا و بفرض أن الكلفة التقديرية للتجريبية (الكلفة المتوقعة) هي المسافة التي تصل كل مدينة بالهدف عبر خط مستقيم (خط النظر).

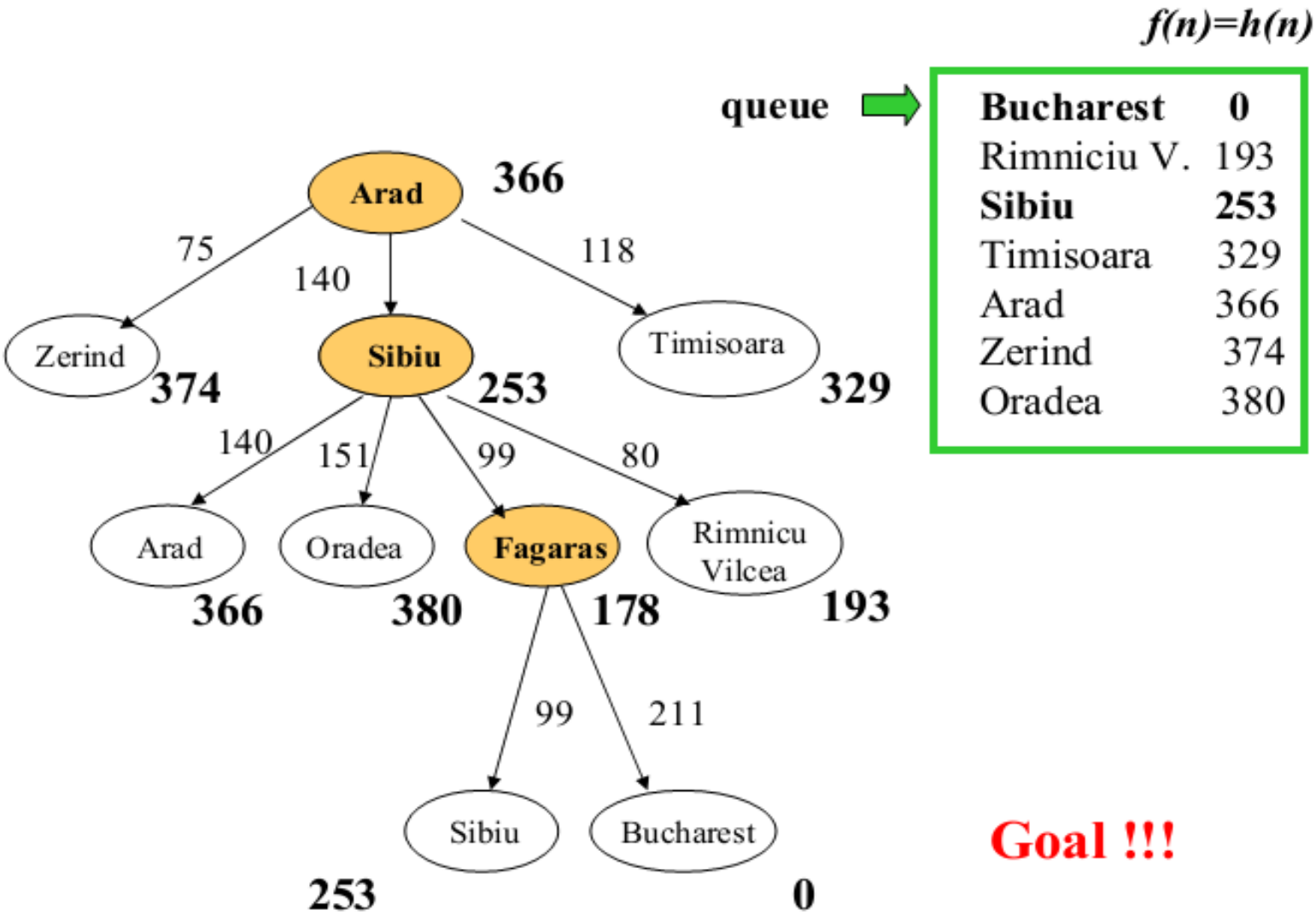
best-first search(Greedy)
البحث بأفضل أول (الجشع)



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

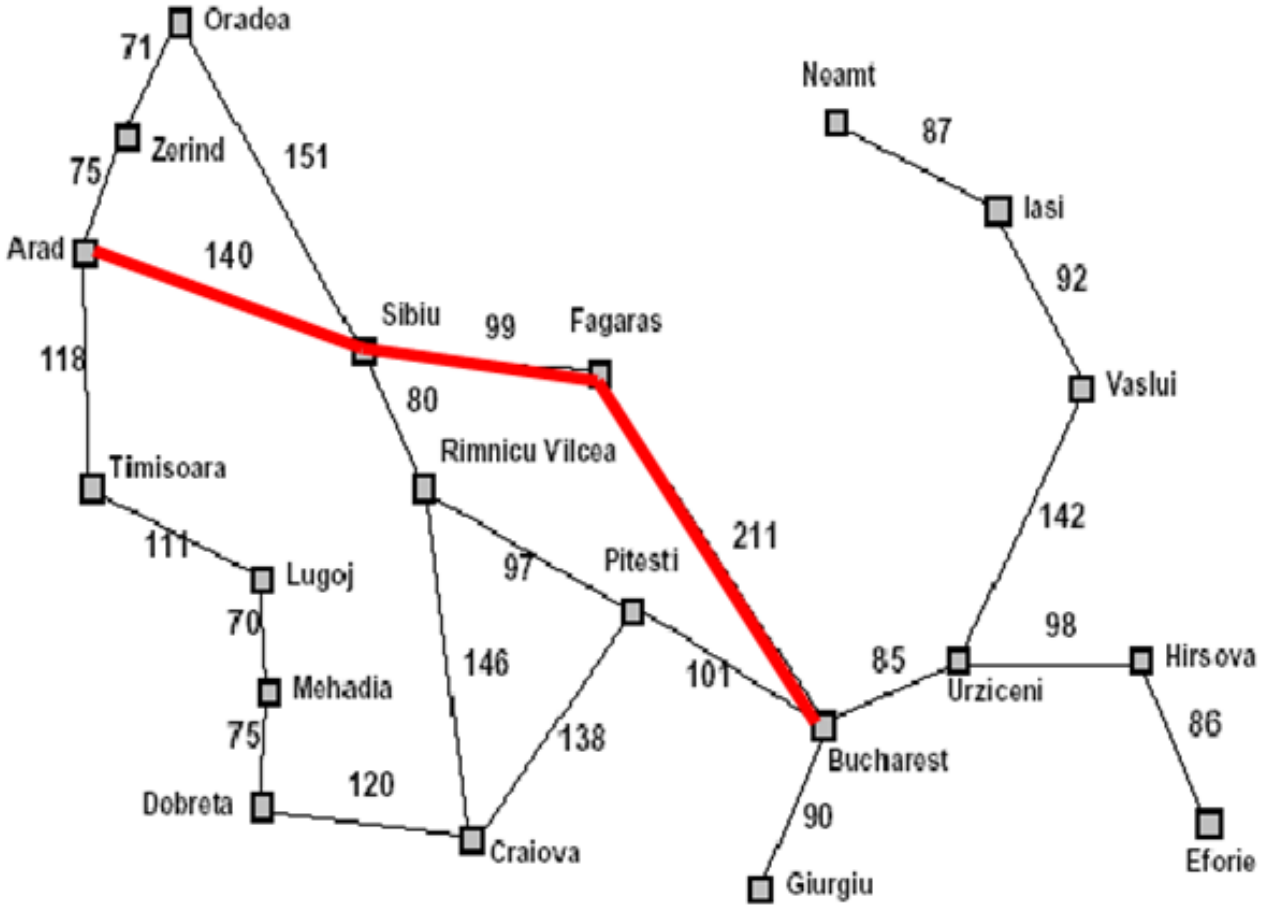
best-first search(Greedy)
البحث بأفضل أول (الجشع)

Greedy search



best-first search(Greedy)
البحث بأفضل أول (الجشع)

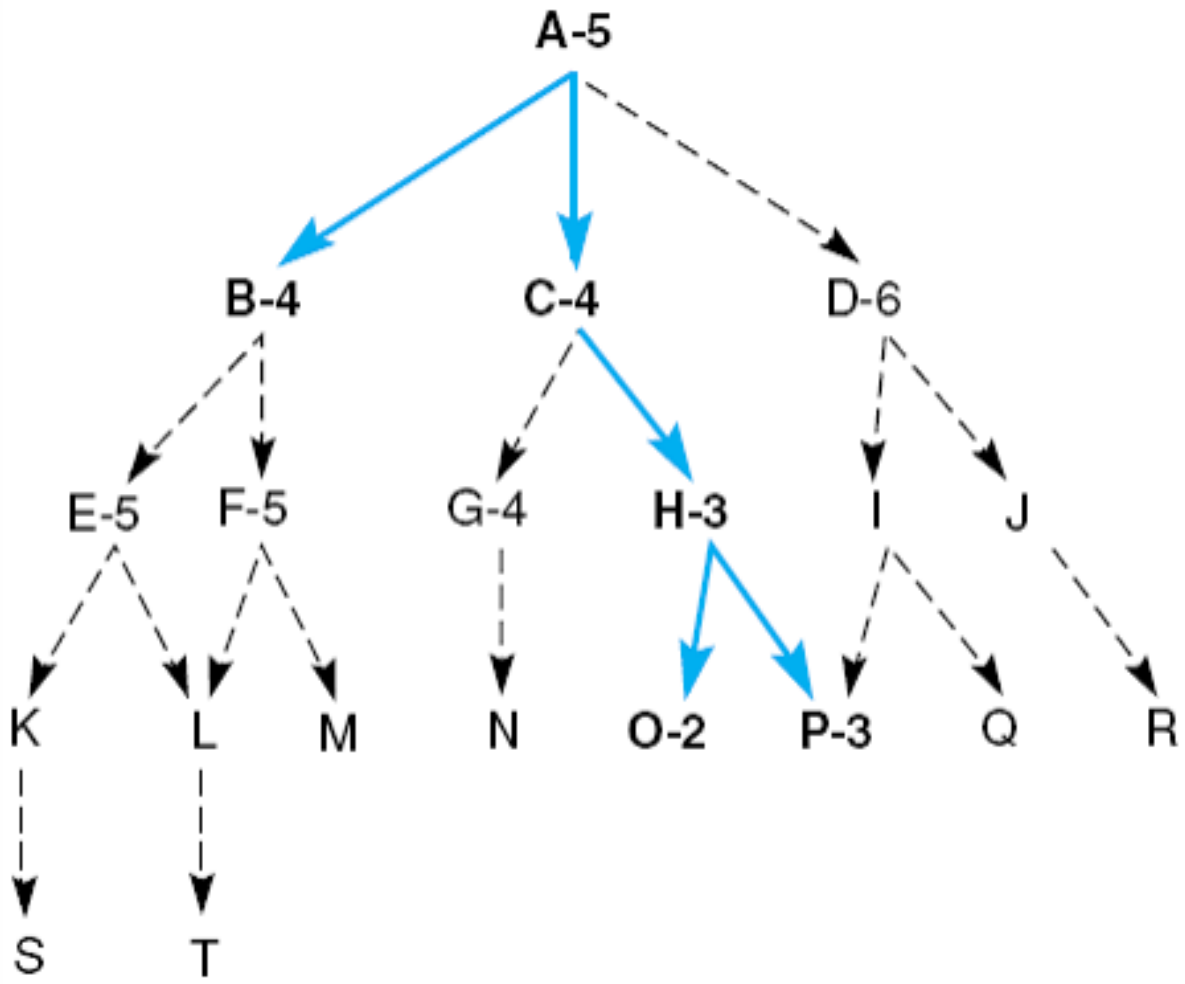
Example: traveler problem with straight-line distance information



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 result**

best-first search(Greedy)
البحث بأفضل أول (الجشع)



فضاء حالة افتراضي

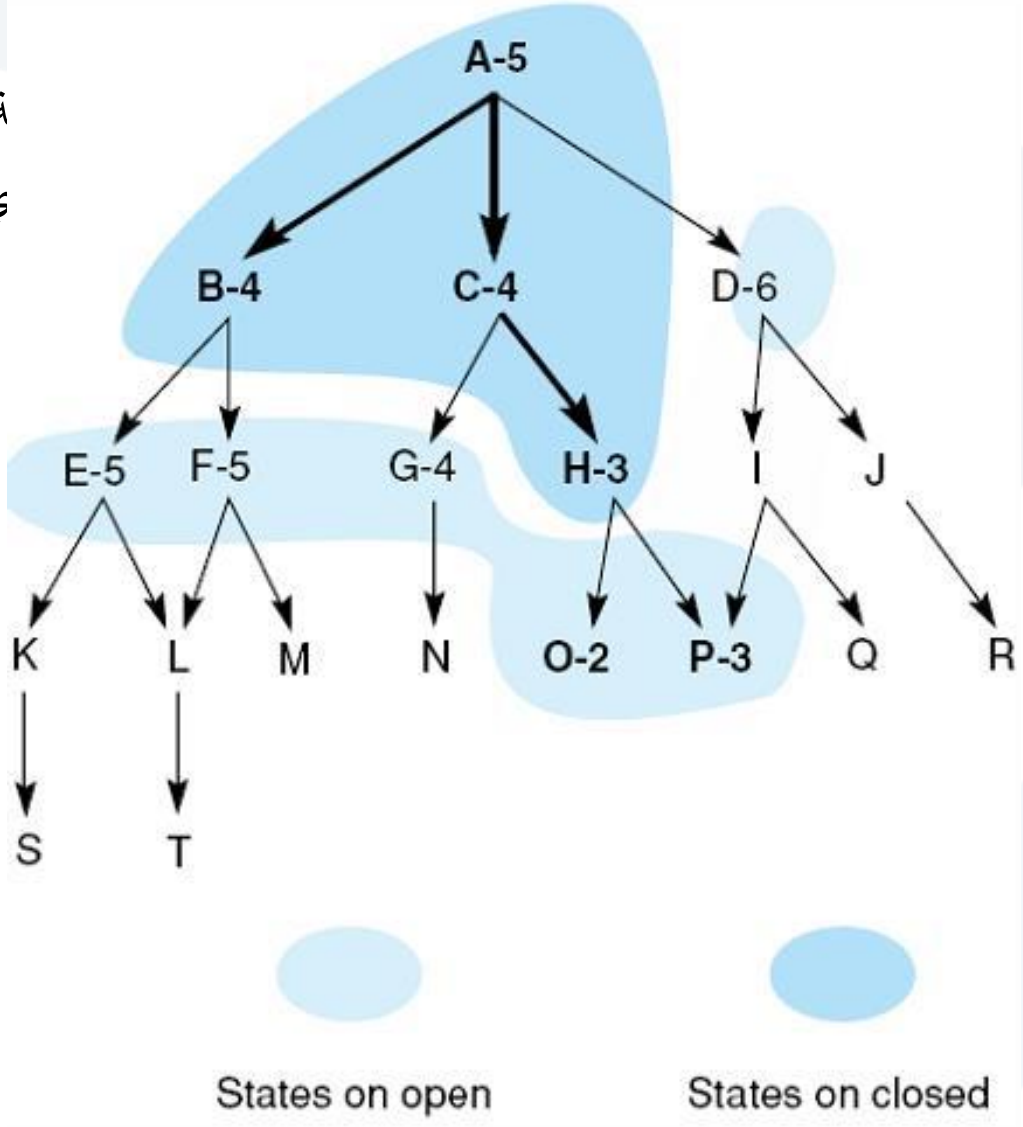
سنقوم ببحثه

best-first search(Greedy)
البحث بأفضل أول (الجشع)

نتائج تقصي تطبيق الخوارزمية

1. **open = [A5]; closed = [ ]**
2. **evaluate A5; open = [B4,C4,D6]; closed = [A5]**
3. **evaluate B4; open = [C4,E5,F5,D6]; closed = [B4,A5]**
4. **evaluate C4; open = [H3,G4,E5,F5,D6]; closed = [C4,B4,A5]**
5. **evaluate H3; open = [O2,P3,G4,E5,F5,D6]; closed = [H3,C4,B4,A5]**
6. **evaluate O2; open = [P3,G4,E5,F5,D6]; closed = [O2,H3,C4,B4,A5]**
7. **evaluate P3; the solution is found!**

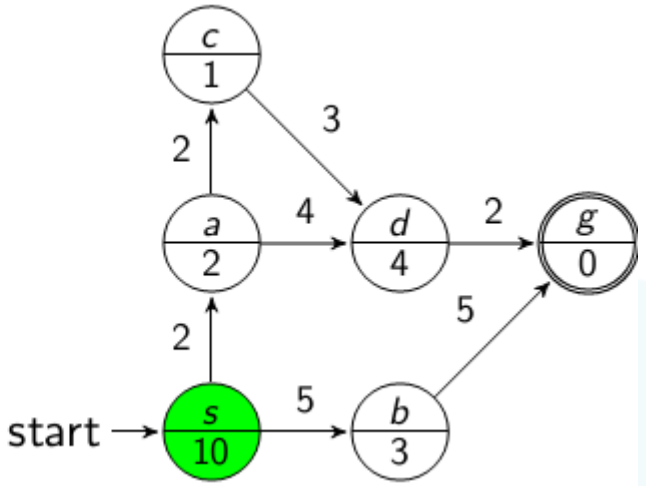
best-first search(Greedy)
البحث بأفضل أول (الجشع)



best-first search(Greedy)
البحث بأفضل أول (الجشع)

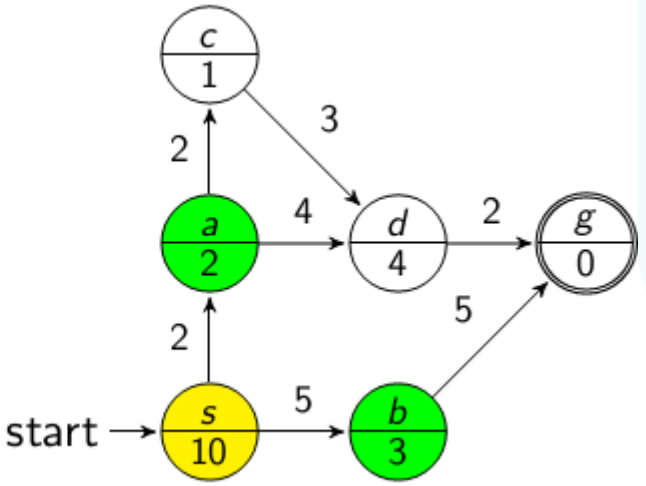
Q:

path	cost	h
$\langle s \rangle$	0	10



Q:

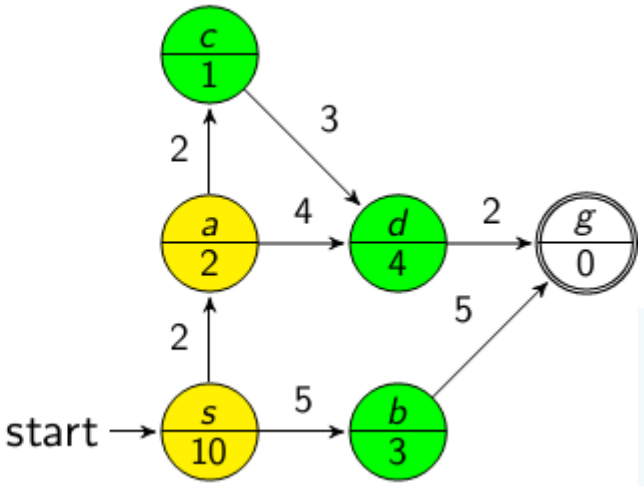
path	cost	h
$\langle a, s \rangle$	2	2
$\langle b, s \rangle$	5	3



best-first search(Greedy)
البحث بأفضل أول (الجشع)

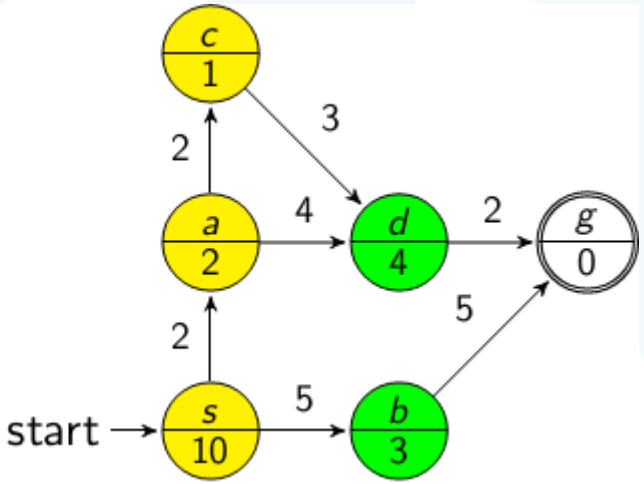
Q:

path	cost	h
$\langle c, a, s \rangle$	4	1
$\langle b, s \rangle$	5	3
$\langle d, a, s \rangle$	6	4



Q:

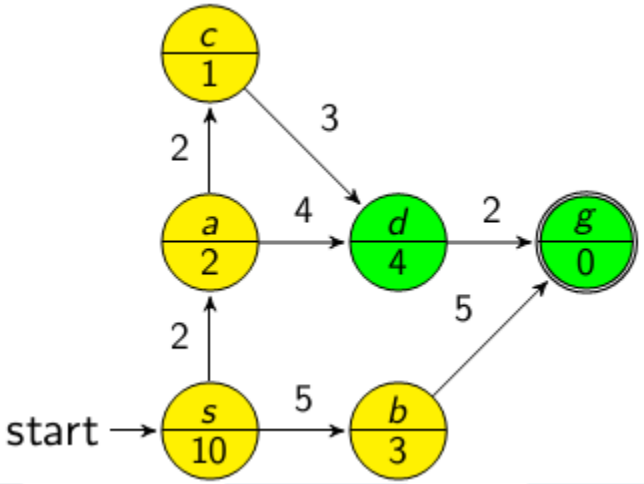
path	cost	h
$\langle b, s \rangle$	5	3
$\langle d, a, s \rangle$	6	4
$\langle d, c, a, s \rangle$	7	4



best-first search(Greedy)
البحث بأفضل أول (الجشع)

Q:

path	cost	h
$\langle g, b, s \rangle$	10	0
$\langle d, a, s \rangle$	6	4
$\langle d, c, a, s \rangle$	7	4



- Greedy search is not complete and not optimal, but is often fast and efficient, depending on the heuristic function *h*

الخورزمية A^*

- سوف نخصص خوارزمية GRAGH SEARCH لخوارزمية بحث الأفضل أولا التي تعيد ترتيب العقد (في الخطوة ٧) في OPEN وفق القيم التصاعدية للدالة f، سوف نسمي هذا الإصدار من GRAGH SEARCH : خوارزمية A^* .
- لتكن h(n)= التكلفة الحالية للمسار الأقل تكلفة بين العقدة n و عقدة الهدف (من بين جميع عقد الأهداف الممكنة ومن بين جميع المسارات الممكنة من n إلى هذه العقد).
- و لتكن g(n) = تكلفة المسار الأقل تكلفة بين عقدة البداية n_o و العقدة n.

الخورزمية A^*

- حينئذ تصبح f(n) = h(n) + g(n) تكلفة المسار الأقل تكلفة من n_o إلى عقدة هدف من بين جميع المسارات (المقيدة) التي تمر عبر العقدة n. لاحظ أن f(n_o) = h(n_o) هي عندئذ تكلفة المسار (غير المقيد) الأقل تكلفة من n_o إلى عقدة هدف.
- من أجل كل عقدة n لتكن $f(n)$ «العامل التجريبي» بحيث يكون $h(n)$ تقديرا ما ، و لتكن $g(n)$ «عامل العمق» تكلفة المسار الأقل تكلفة الذي حصلنا عليه بخوارزمية A^* لغاية العقدة n .
- نستخدم في خوارزمية A^*: $f = h + g$.
- لاحظ أن هذه الخوارزمية تعطي $h = 0$ مع خوارزمية البحث ذا التكلفة الموحدة.

الخورزمية A^*

- و لمنع الخوارزمية من الدخول في حلقات أثناء بحثها عن مسار الهدف نعدل خوارزمية GRAGH SEARCH و نقوم باستبدال الخطوة ٦ لتصبح:

- ٦- وسع العقدة n بتوليد مجموعة M من الخلَف بحيث لا يكون أحدها سلفا سابقا للعقدة n في T_r .

ارسِ M باعتبارها خلفا للعقدة n في T_r و ذلك بإنشاء الأقواس من n إلى كل عنصر من M . ضع هذه العناصر من M في OPEN.

الخورزمية A^*

- ١- أنشئ بيان بحث G مكونا فقط من عقدة البداية n_o . ضع n_oفي قائمة تسمى OPEN .
- ٢- أنشئ قائمة تسمى CLOSED تكون فارغة في البداية .
- ٣- إذا كانت OPEN فارغة، اخرُج بالنتيجة: إخفاق.
- ٤- اختر العقدة الأولى في OPEN، و احذفها من OPEN ، و ضعها في CLOSED . سم هذه العقدة n .
- ٥- إذا كانت n عقدة هدف، اخرج بنجاح مع الحل الذي حصلت عليه برسم مسار راجع على طول المؤشرات من n إلى n_o رجوعا إلى G(تبني المؤشرات شجرة بحث و هي تنشأ في الخطوة ٧).
- ٦- وسع العقدة n بتوليد مجموعة M من خلَفها التي ليست الآن من أسلاف n في G. أرسِ عناصر M باعتبارها خلفا للعقدة n في G.

الخورزمية A^*

- ٧- أنشئ مؤشرا إلى n انطلاقا من كل عنصر من M غير موجودة بعد في *أي غير موجودة بعد في ((G* OPEN و لا في CLOSED*))*.

أضف هذه العناصر من M إلى OPEN .

و من أجل كل عنصر m من M ، انتمى سابقا إلى OPEN أو إلى CLOSED.

أعد توجيه مؤشره إلى n إذا كان أفضل مسار إلى m وجدناه إلى الآن هو عبر n.

و من أجل كل عنصر من M انتمى سابقا إلى CLOSED ، أعد توجيه جميع مؤشرات خلفه في G بحيث تؤشر رجوعا على طول أفضل المسارات التي وجدناها إلى ان تصل إلى هذه الخلف.

- ٨- أعد ترتيب القائمة OPEN وفق قيم f التصاعدية. (يجري حلّ روابط قيم f الصغرى لمصلحة أعمق عقدة في شجرة البحث)
- ٩- عد إلى الخطوة ٣ .

A* Search Algorithm

```
OPEN = {InitialState}; CLOSE = nil; SUCCESS = False
While OPEN != {} And SUCCESS=False do
Begin
    u = Get_First_Node_From_OPEN(OPEN)
    OPEN = Queue(OPEN)
    If GoalState (u) Then   SUCCESS=True
    Else
    Begin
        Add(u, CLOSE)
        For each v in Children(u) do
        Begin
            If  v Not in OPEN and v not in CLOSE Then
                Add (v, OPEN)
                Father(v)=u
                Sort(OPEN)              % f ASC then g DECS
            Else If v in OPEN and g(v)>g(u)+k(u,v) Then
                g(v)=g(u)+k(u,v)
                f(v)=g(v)+h(v)
                Father(v)=u
            Else If v in CLOSE and g(v)>g(u)+k(u,v) Then
                g(v)=g(u)+k(u,v)
                f(v)=g(v)+h(v)
                Father(v)=u
        End
    End
End
If  OPEN={} Then FAIL
Else
    GenerateSolutionPath(u)
```

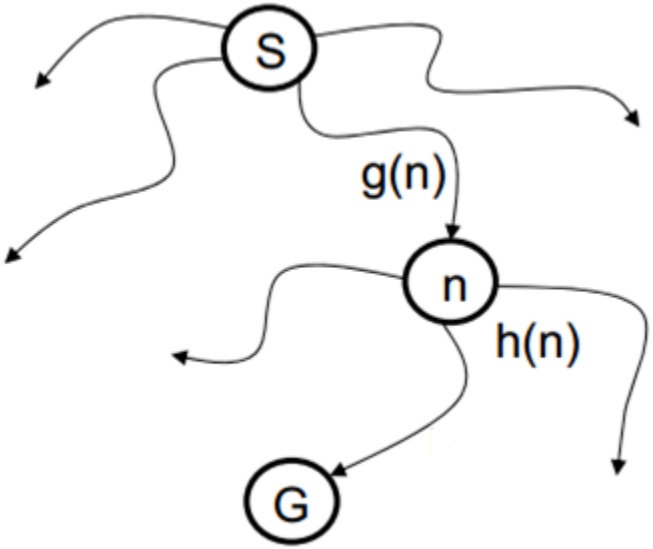
الخورزمية A^*

الفكرة الاستراتيجية لهذه الخوارزمية هي تجنب التوسع في ممرات كلفتها أكبر إلى هذه العقدة، و أن

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

حيث g(n) هو التكلفة انطلاقا من الحالة الابتدائية إلى العقدة n

و h(n) هو تخمين التكلفة من العقدة n إلى الهدف (التابع الحدسي)



Minimize the total path cost

ملاحظات

1-يمكننا استخدام الأسلوب الذي اتبعناه في خوارزميات البحث العمياء وهو أن نضع العقد التي اكتشفناها ولم نزرها ضمن بنية معينة.

في Depth- First كنا نستخدم Stack (LIFO)

في Breadth- First كنا نستخدم Queue (FIFO)

في Uniform Cost كنا نستخدم Priority Queue حسب Cost

أما في خوارزمية A* سنعتمد على Priority Queue وستكون الأفضلية حسب قيمة F(n)

2-في حال كانت قيمة التجريبية H=0 عندها ستتحول الخوارزمية إلى Uniform Costوبالتالي سنصل لحل أمثلي ولكن بعد أن نقوم بسبر كل الحلول.

3-اذا كانت H تساوي الكلفة الحقيقية سيكون الأداء سريع جداً ولن نقوم بسبر فضاء الحلول كاملاً.

4-حسن اختيار التجريبية هو العامل الأساسي بالأداء والهدف هو تسريع الوصول إلى للهدف دون سبر كل فضاء الحلول.

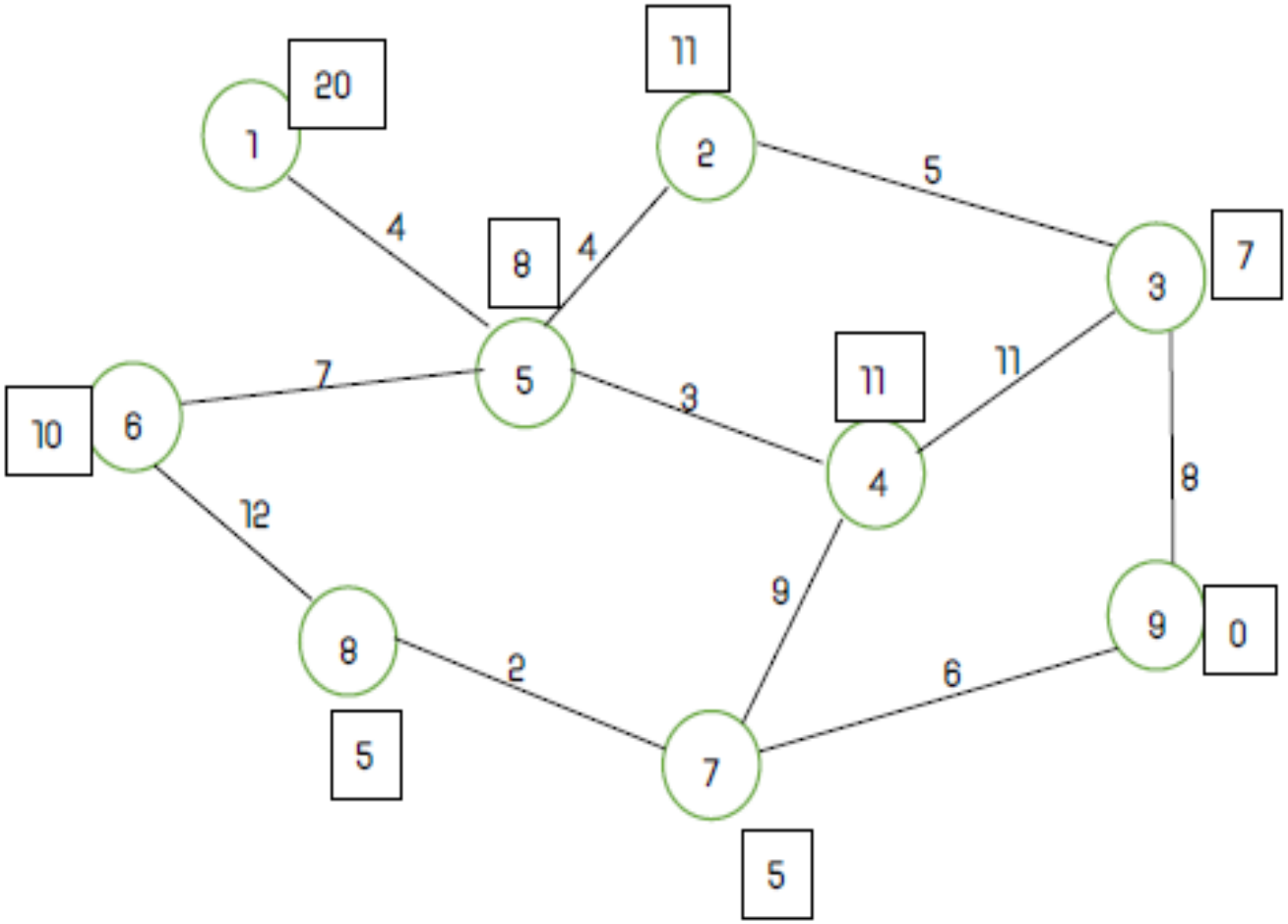
ملاحظات

- في Open نضع العقد المولدة والتي علينا زيارتها.
- وفي Close نضع العقد التي تمت زيارتها والتي ولدنا أبناءها أي العقد المطورة.

- ملاحظة:
- اذا تساوت قيم التابع F لأكثر من عقدة سنقوم باختيار العقدة الأُب إلى الهدف ذات أصغر قيمة للدالة H

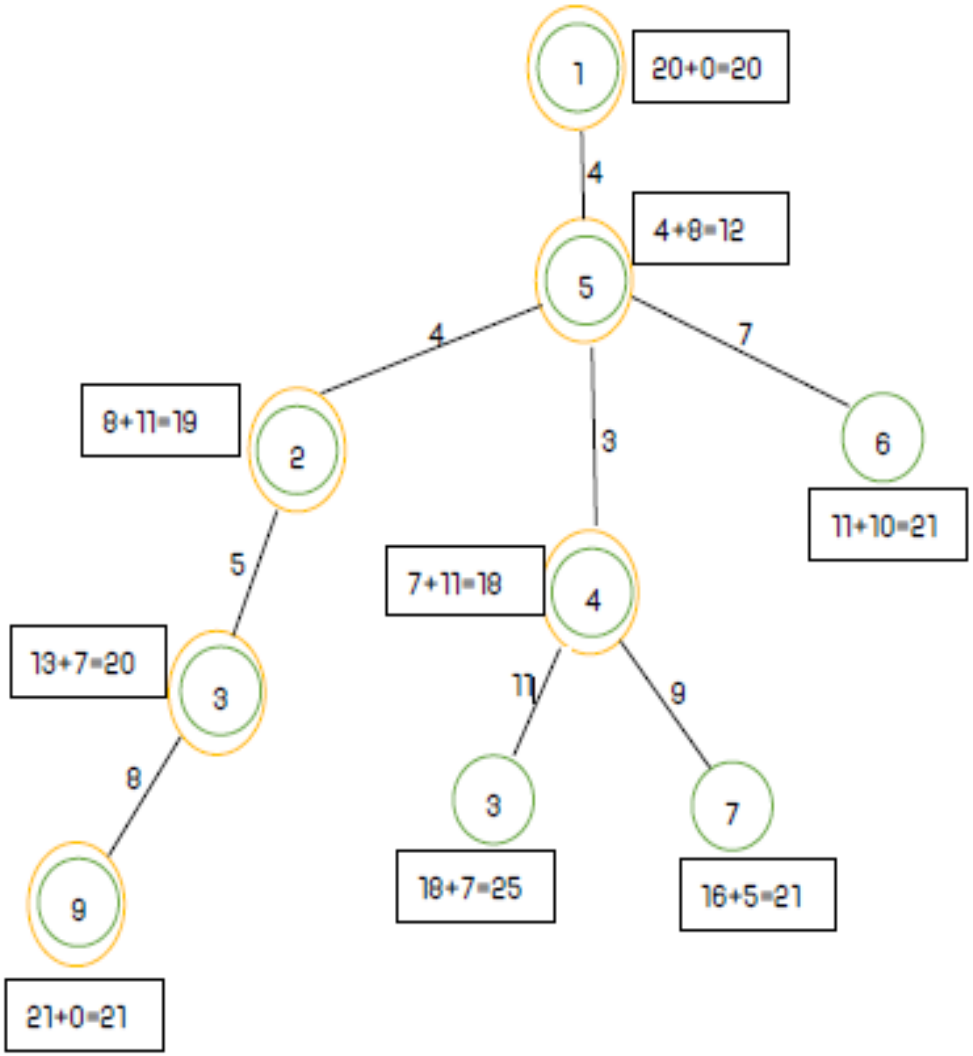
مثال (١)

- ما هو عدد العقد المولدة والعقد المطورة وما ترتيب العقد المطورة في المثال التالي:



مثال (١)

- الأرقام الموجودة ضمن المربعات هي قيم التجريبيات.
- سيكون الحل على الشكل التالي:

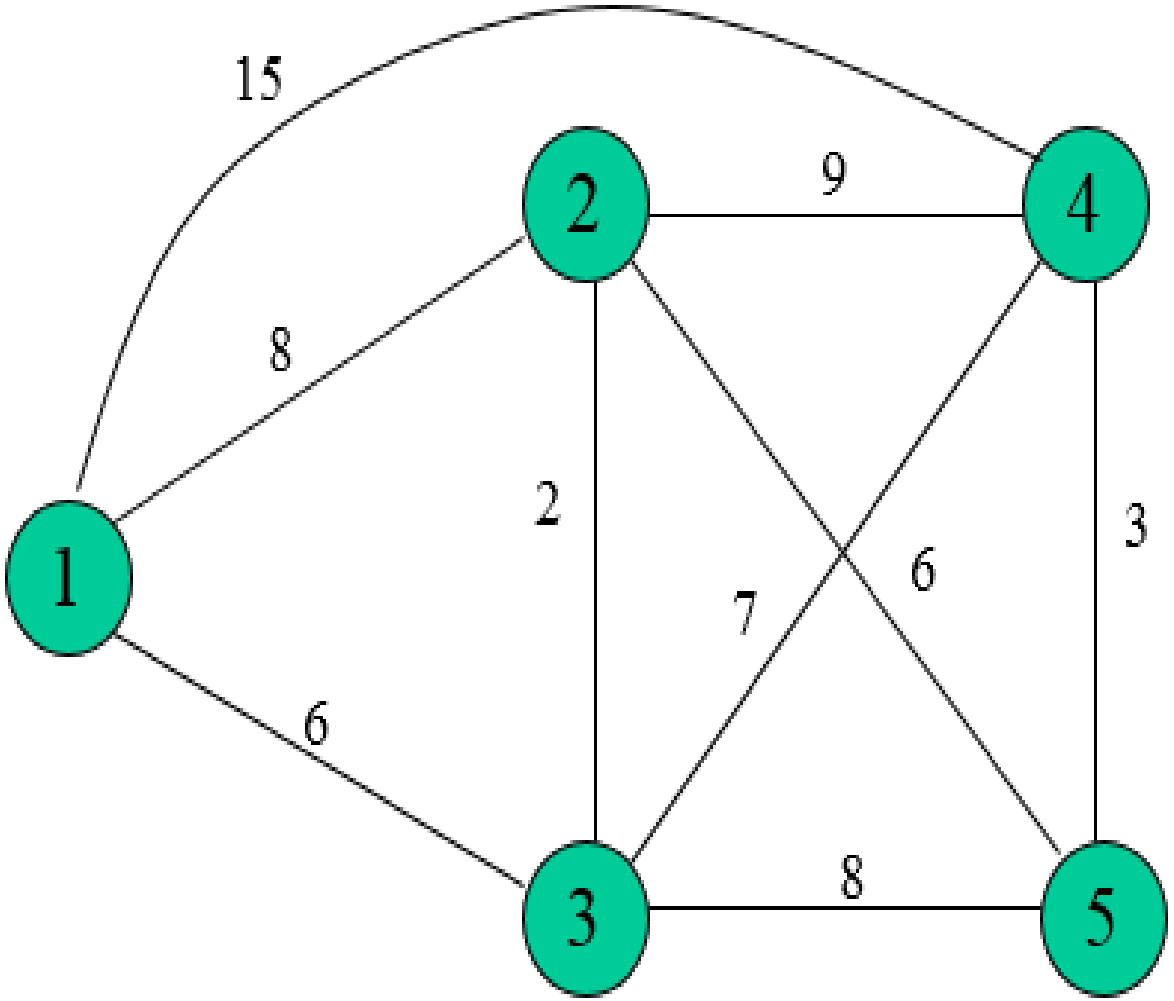


مثال (١)

- ملاحظات عن الحل:
-
- ١-عند العقدة 4 تراجعنا إلى الوراء بسبب وجود أوراق آخرى ذات كلفة أقل.
- 2-عند العقدة 3 كان بإمكاننا أن نصل للعقدة 4 ولكننا قمنا بإهمالها لأننا استطعنا الوصول إليها من طريق آخر وبكلفة أقل.
- 3-وبما أننا نقوم بوضع العقد ضمن Queue مرتبة سيكون الهدف من ضمنها ولا نستطيع التوقف عن توليد وتطوير عقد الشجرة إلى أن يكون الهدف في رأس Queue (أي عقدة الهدف هي العقدة ذات الكلفة الأقل.)
- وسيكون ترتيب العقد المطورة هو:
- 1→5 → 4 → 2 → 3 → 9
- .وعددها هو 6 عقد مطورة أما العقد المولدة فعددها هو9

مثال (٢)

- ليكن لدينا البيان التالي:



مثال (٢)

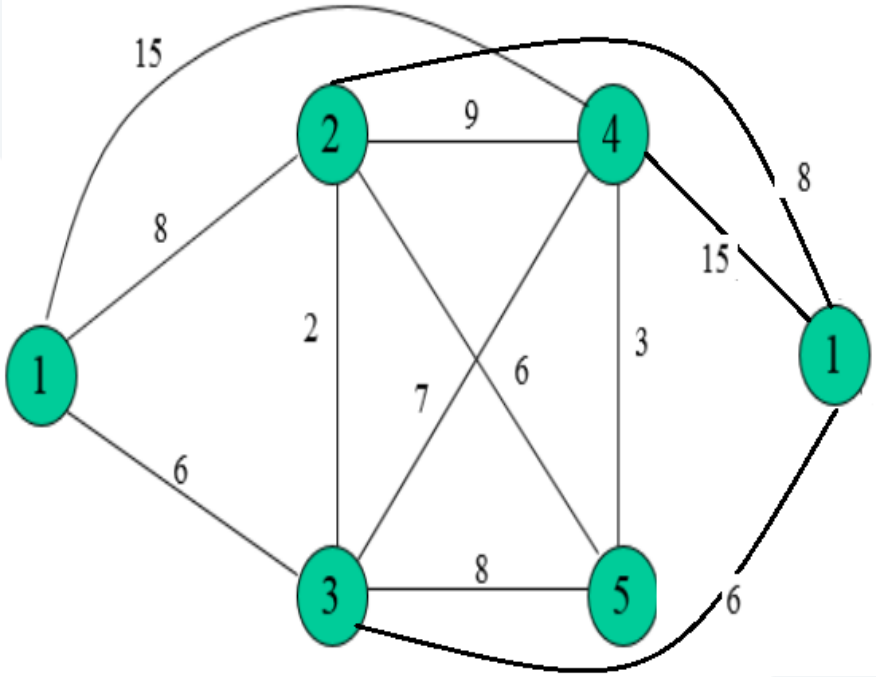
- يطلب تنفيذ البحث في البيان السابق باتباع خوارزميه ***A***
- على أن يتم الانطلاق من المدينة الأولى و العودة إليها بالمرور على جميع المدن و غير مسموح زيارة مدينة مرتين.

- سنتبع مع هذا البيان تجريبية:

أقصر الطرق الداخلة للعقد المتبقية

- وقبل البدء بالحل يجب أن نلاحظ أن هذا البيان كامل أي لا يوجد لدينا طرق مسدودة . وفي حالة البيان غير الكامل سيكون لدينا طرق مسدودة وضرورة في التراجع عن بعض الحلول.

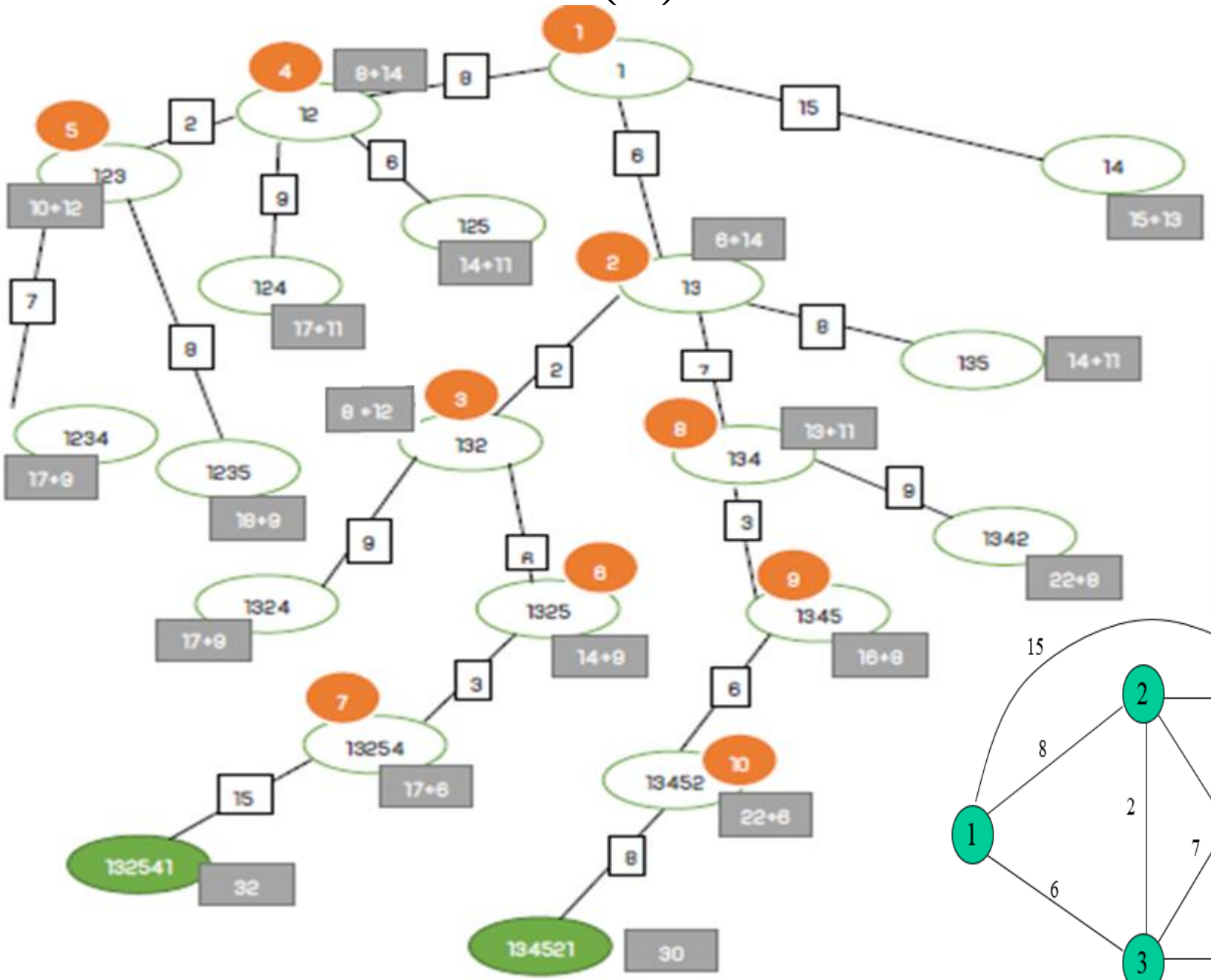
مثال (٢)



الحل:

- نحدد أقصر طريق داخل على كل مدينة.
- العقدة 1←6
- العقدة 2 ←2
- العقدة 3 ←2
- العقدة 4 ←3
- العقدة 5 ←3
- العقد (12) 1+3+4+5 ←14=6+2+3+3
- العقدة(123) 1+4+5 ←12=6+3+3
- العقدة (1325) 1+4 ←9=6+3
- و هكذا.....

مثال (٢)


1
1
4
8+14
8
12
2
5
123
10+12
9
124
17+11
6
125
14+11
15
14
15+13
6
2
8+14
13
7
8
1234
17+9
1235
18+9
2
8 +12
3
132
9
1324
17+9
6
1325
14+9
3
7
13254
17+6
15
132541
32
8
134521
30
7
8+14
8
135
14+11
2
7
8
13+11
134
3
9
1342
22+8
9
1345
16+8
6
10
13452
22+8
8

15
9
8
2
7
6
3
6
8
1
2
3
4
5

مثال (٢)

ملاحظات عن الحل:

←الدوائر البيضاء تعبر عن المدن التي تمت زيارتها حتى الآن فمثلا العقدة 123 تعبر عن تسلسل الزيارة من المدينة الأولى ,فالثانية , انتهاء بالمدينة الثالثة .

← كل الدوائر التي بجانب العقد" الدوائر البرتقالية " تدل على تسلسل زيارة العقد أي مثلاً العقدة 13452 هي العقدة العاشرة التي تمت زيارتها ومعالجتها.

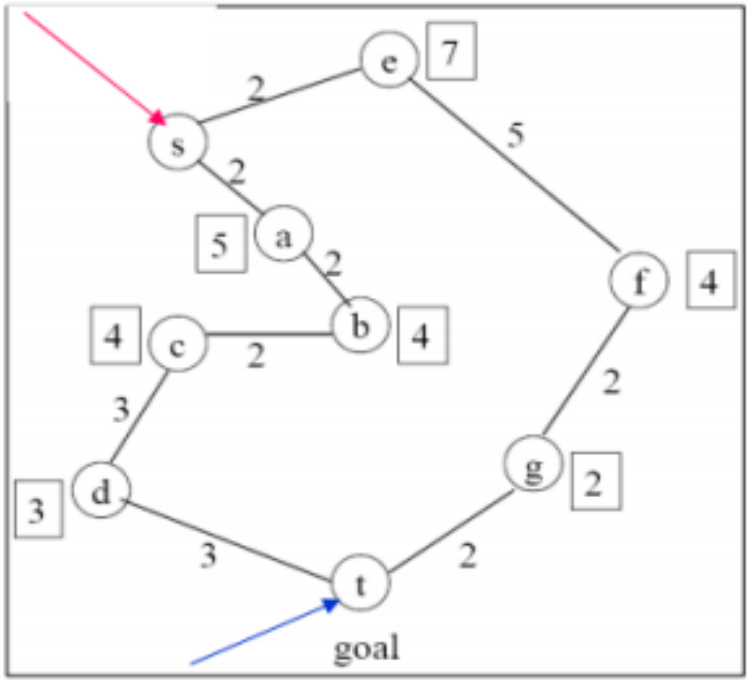
←العقد ذات اللون الأخضر هي عقد تمثل حلاً نهائياً ولكن ليس بالضرورة أمثلي.

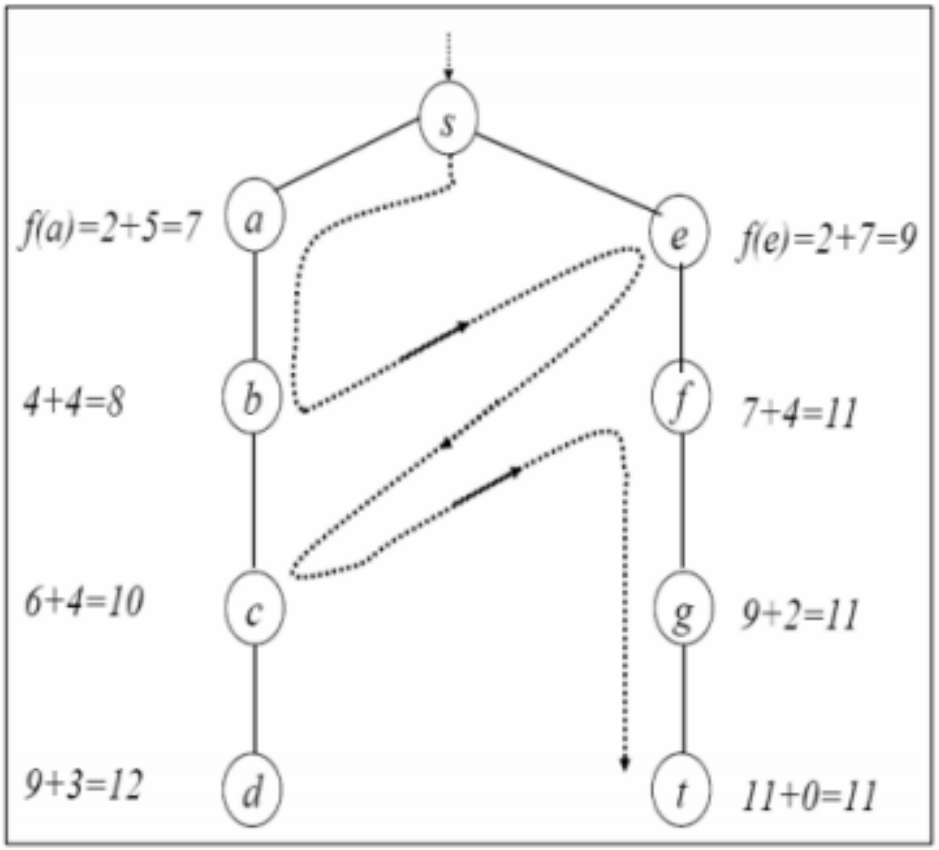
← المستطيلات الرمادية بجانب كل عقد تمثل قيم التجريبيات التي تم استخدامها فمثلاً عند العقدة 132 نجد أن الرقم 8 يعبر الكلفة الحقيقية و التي هي 6+2 حيث 6 هي تكلفة الوصول من العقدة 1 إلى العقدة 2 و إن الرقم 2 هو تكلفة الوصول من العقدة 3 إلى العقدة 2. أما بالنسبة للرقم 12 فيدل على مجموع أقصر الطرق إلى العقد المتبقية و التي تم تحديدها قبل البدء بالحل.

مثال (٢)

- ملاحظات عن الحل:
- ←في كل مرحلة من المراحل نقوم بمعالجة الورقة التي تكون التكلفة عندها أقل ما يمكن , وهذا ما يفسر انتقالنا من عقدة إلى أخرى فمثلاً عندما عالجنا العقدة 123 وجدنا أنه تكلفة الورقتين الناتجتين عنها كبيرة وأقل تكلفة بين الأوراق هي تكلفة العقدة. 1325
- ← عندما عالجنا العقدة 13254 وجدنا أن الورقة الناتجة عنها تعبر عن حل ولكن مع وجود احتمال أن يكون هنالك حل أفضل من الحل الذي توصلنا له سنعود لمعالجة الأوراق ذات الكلفة الأقل وهي العقدة. 134
- ←نلاحظ أن أوراق الشجرة غير كاملة طبعاً يجب أن نستمر في انتقاء الورقة الأصغر كلفة ومعالجتها إلى أن تصبح إحدى الأوراق التي تحوي الحل هي الورقة ذات الكلفة الأدنى .... وبالتالي علينا إكمال الشجرة في الأعلى←

مثال (۳)





$f(c) = g(c) + h(c) = 6 + 4 = 10$

$f(e) = g(e) + h(e) = 2 + 7 = 9$

$f(e) < f(c)$

$f(f) = 7 + 4 = 11$

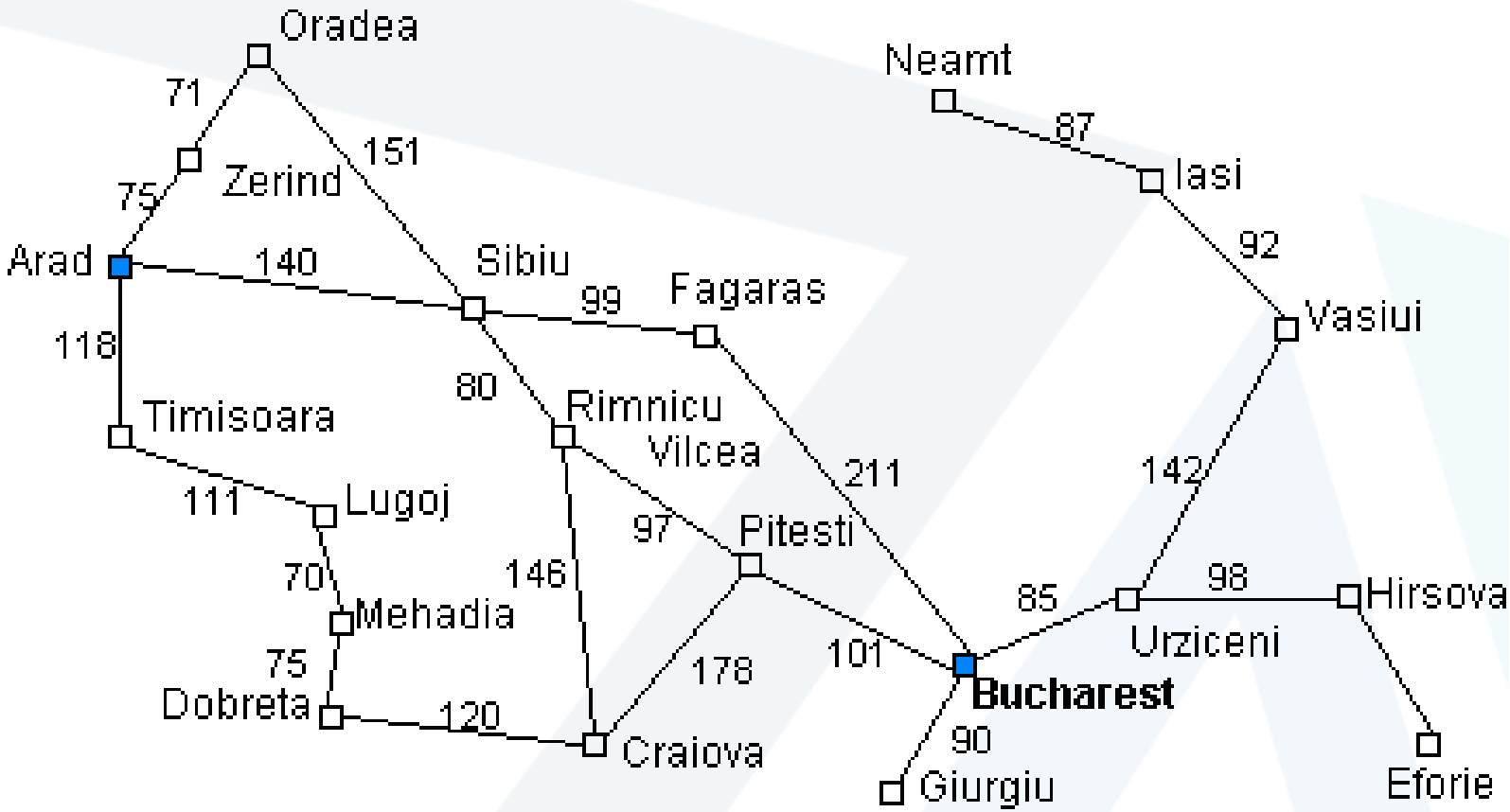
$f(c) = 10$

$f(c) < f(f)$

$f(d) = 12 > f(f)$

$f(g) = 11 < f(d)$

مثال (٤)
الخوارزمية A^*

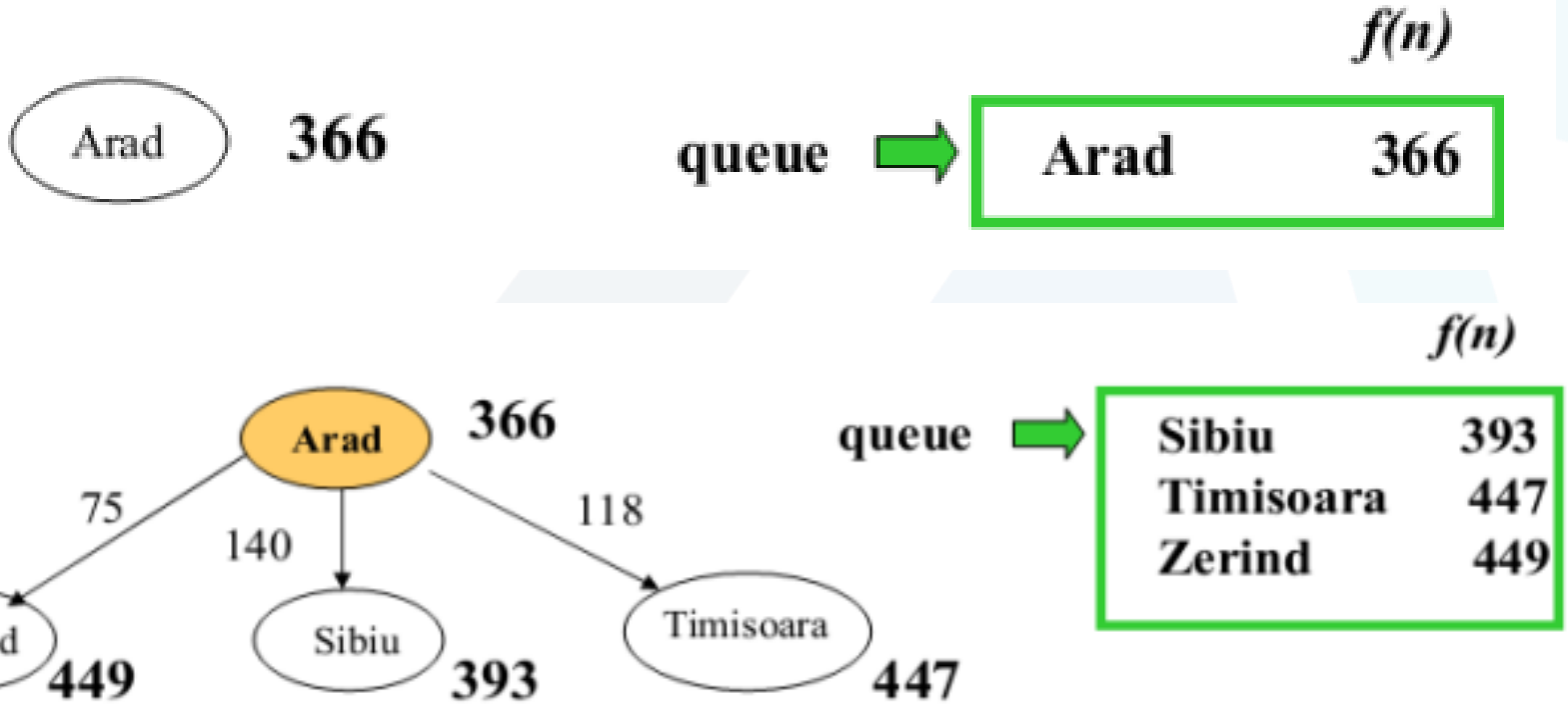


Straight-line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

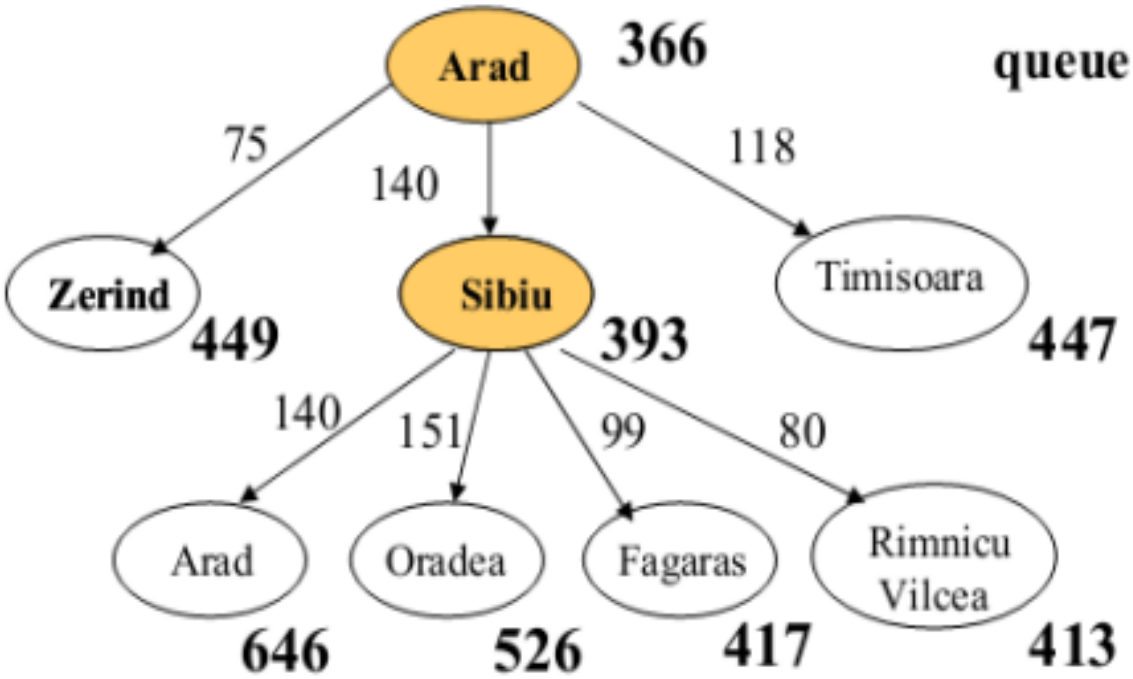
مثال (٤)

admissibility القبول

نعتبر أن الاجتهاد جديراً بالقبول إذا لم تتجاوز الكلفة الحقيقية الكلفة التقديرية للوصول إلى الهدف.



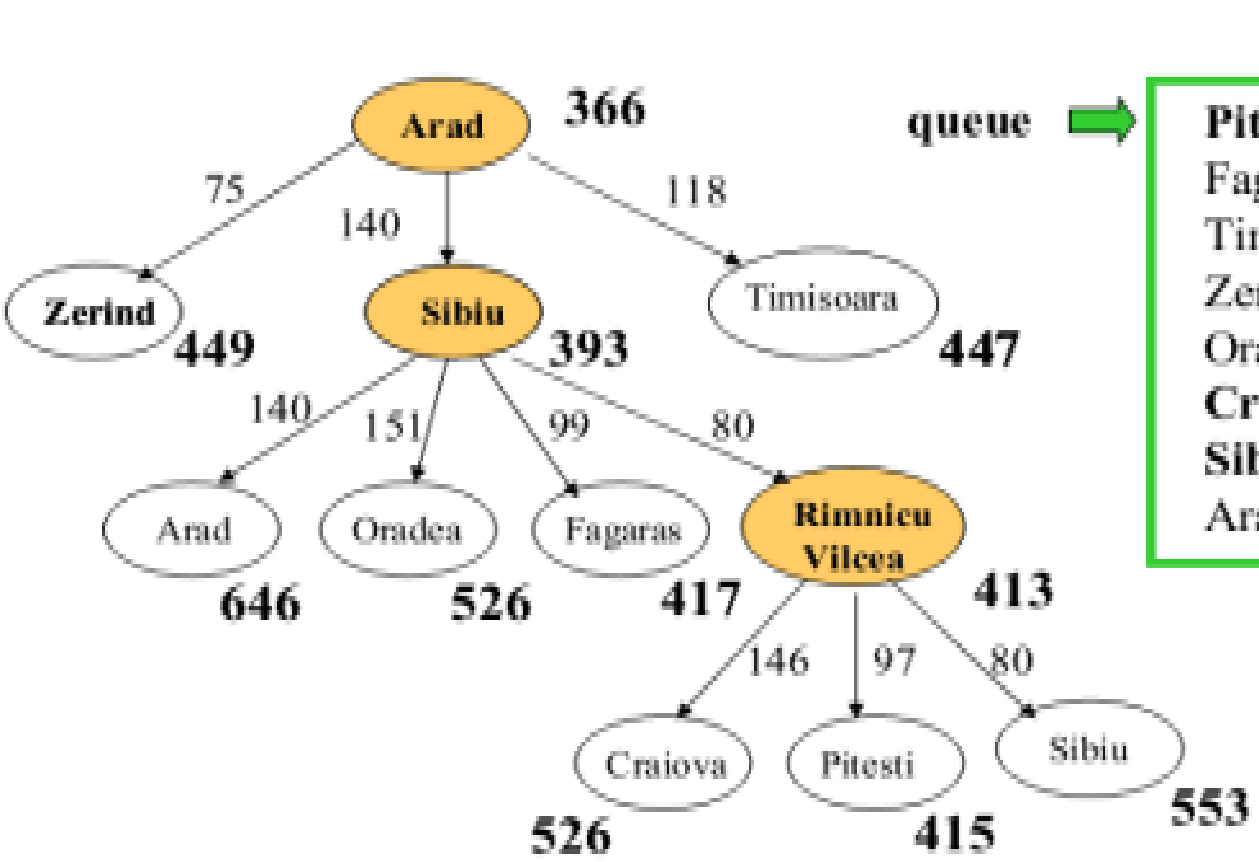
مثال (٤)



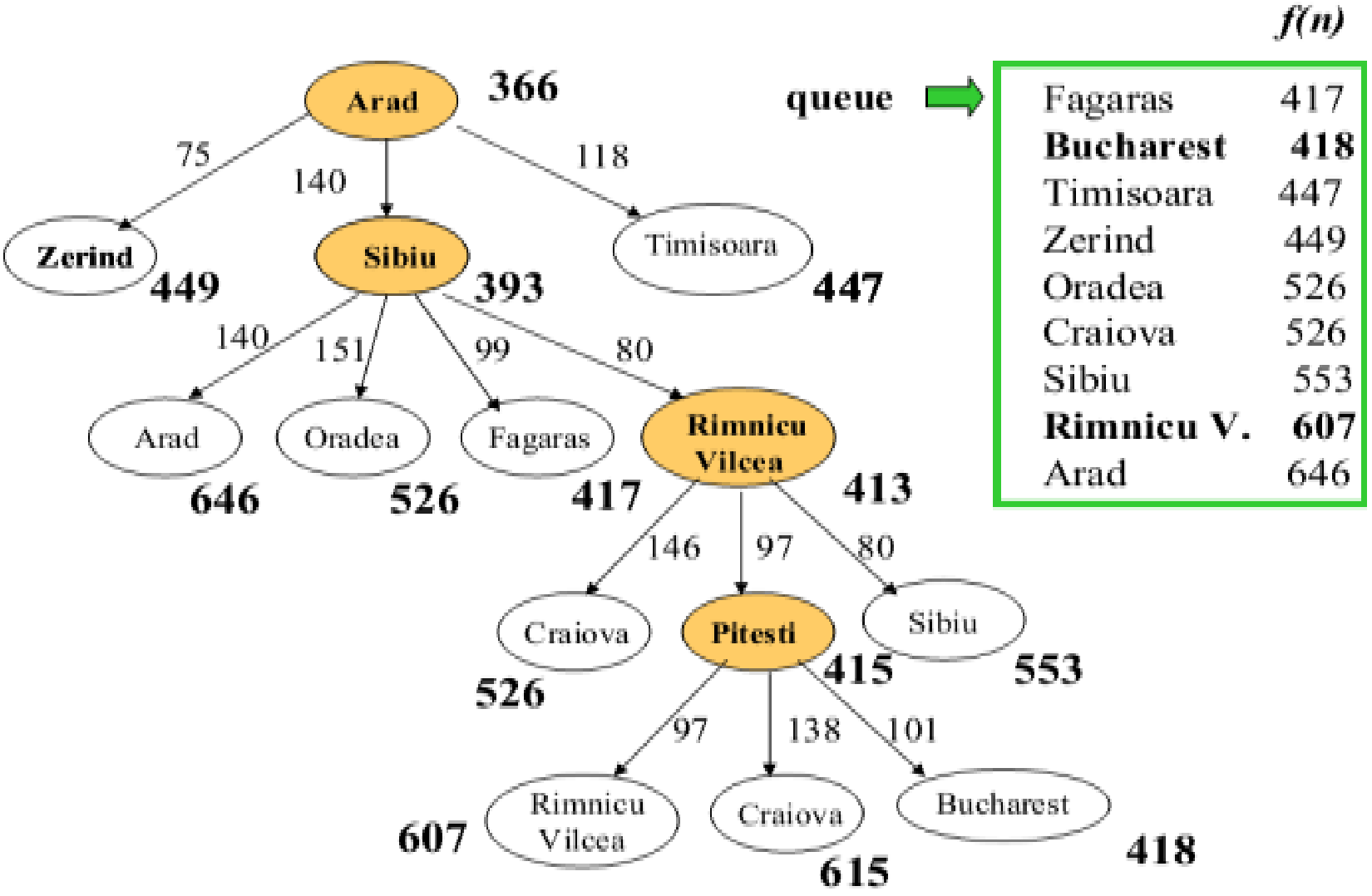
queue

	f(n)
Rimnicu V.	**413**
Fagaras	**417**
Timisoara	447
Zerind	449
Oradea	**526**
Arad	**646**

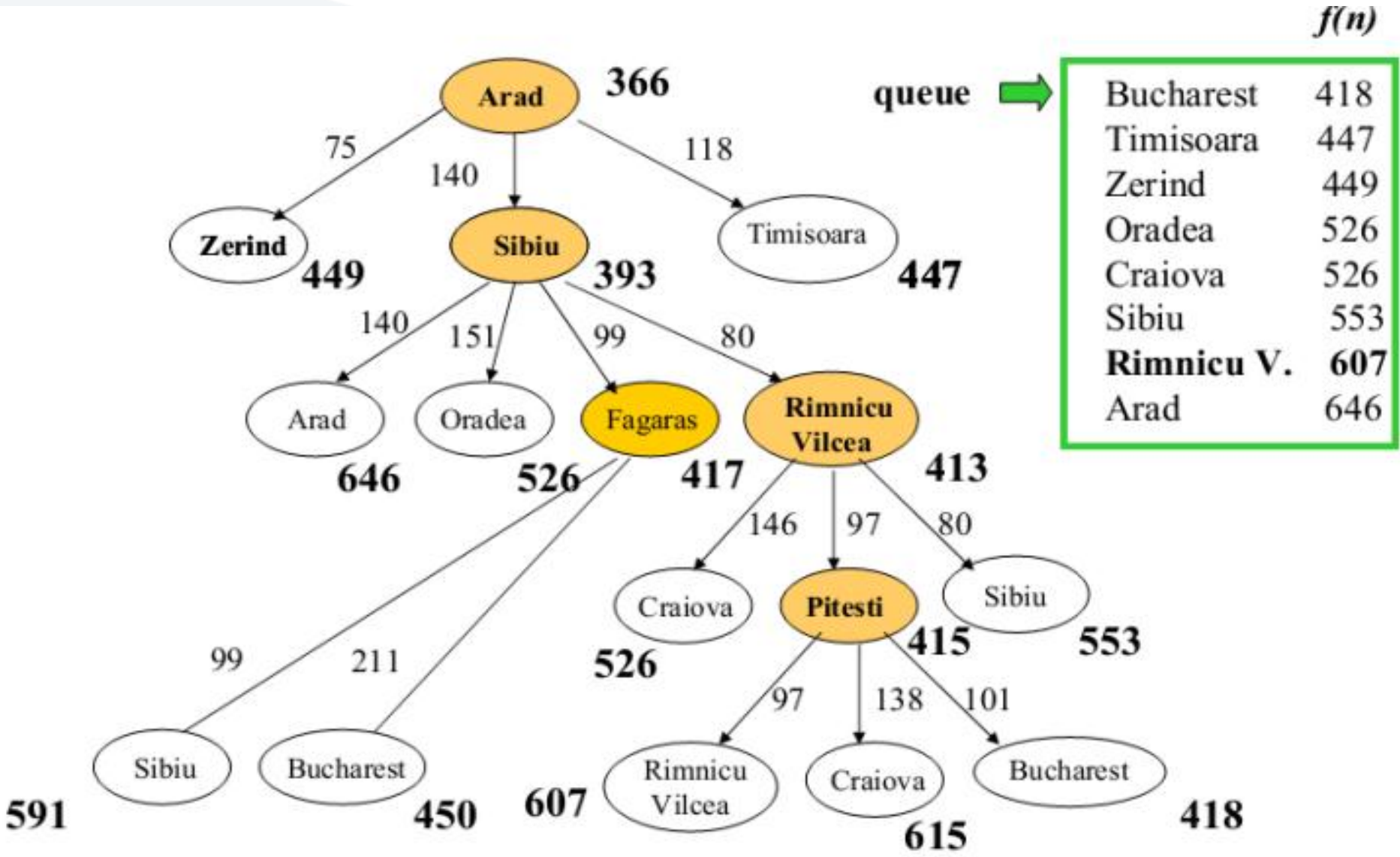
مثال (٤)


f(n)
Arad
366
queue
75
140
118
Zerind
449
Sibiu
393
Timisoara
447
140
151
99
80
Arad
646
Oradea
526
Fagaras
417
Rimnicu Vilcea
413
146
97
80
Craiova
526
Pitesti
415
Sibiu
553
Pitesti 415
Fagaras 417
Timisoara 447
Zerind 449
Oradea 526
Craiova 526
Sibiu 553
Arad 646

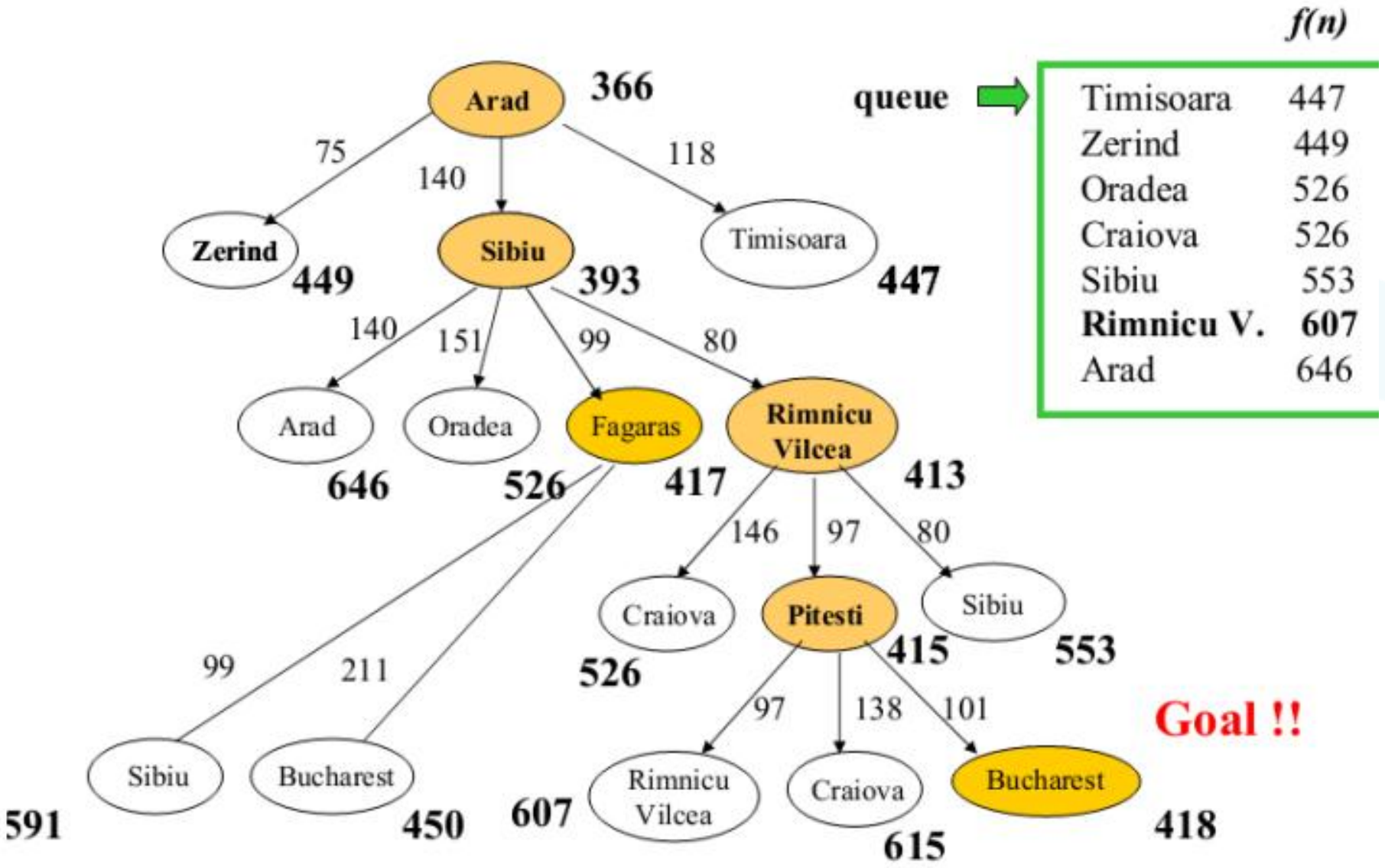
مثال (٤)


f(n)
Arad
366
queue
75
140
118
Zerind
449
Sibiu
393
Timisoara
447
140
151
99
80
Arad
646
Oradea
526
Fagaras
417
Rimnicu Vilcea
413
146
97
80
Craiova
526
Pitesti
415
Sibiu
553
97
138
101
Rimnicu Vilcea
607
Craiova
615
Bucharest
418
Fagaras 417
Bucharest 418
Timisoara 447
Zerind 449
Oradea 526
Craiova 526
Sibiu 553
Rimnicu V. 607
Arad 646

مثال (٤)


f(n)
queue
Bucharest 418
Timisoara 447
Zerind 449
Oradea 526
Craiova 526
Sibiu 553
Rimnicu V. 607
Arad 646
Arad
366
75
140
118
Zerind
449
Sibiu
393
Timisoara
447
140
151
99
80
Arad
646
Oradea
526
Fagaras
417
Rimnicu Vilcea
413
146
97
80
Craiova
526
Pitesti
415
Sibiu
553
99
211
97
138
101
Sibiu
591
Bucharest
450
607
Rimnicu Vilcea
Craiova
615
Bucharest
418

مثال (٤)


f(n)
queue
Timisoara 447
Zerind 449
Oradea 526
Craiova 526
Sibiu 553
Rimnicu V. 607
Arad 646
Arad
366
75
140
118
Zerind
449
Sibiu
393
Timisoara
447
140
151
99
80
Arad
646
Oradea
526
Fagaras
417
Rimnicu Vilcea
413
146
97
80
Craiova
526
Pitesti
415
Sibiu
553
99
211
97
138
101
Sibiu
591
Bucharest
450
Rimnicu Vilcea
607
Craiova
615
Bucharest
418
Goal !!

2 8 3 1 6 4 ■ 7 5	5	6
2 8 3 1 ■ 4 7 6 5	3	4
2 8 3 1 6 4 7 5 ■	5	6
	Tiles out of place	Sum of distances out of place

1 2 3
8 ■ 4
7 6 5
Goal

مثال لقطع الثمانية و توابع التقويم:

يأخذ تابع التقويم أحد الأشكال الثلاثة:

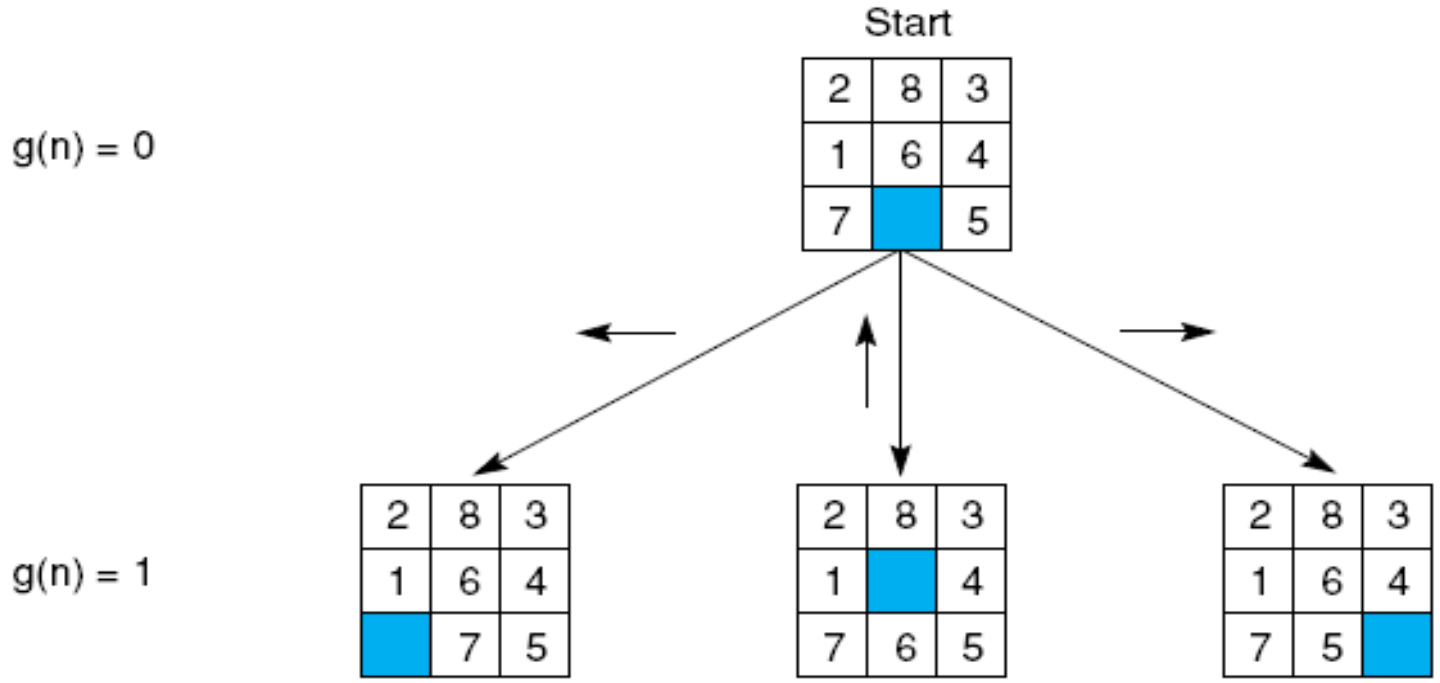
١- عدد القطع التي ليست في مكانها مقارنة مع الهدف (أصغر ما يمكن)

٢- عدد القطع التي في مكانها مقارنة مع الهدف (أكبر ما يمكن)

٣- مجموع بعد كل القطع عن مكانها في الحالة الهدف (مسافة مانهاتن) أي عدد الحركات الضرورية للانتقال من الحالة قيد الدراسة إلى حالة الهدف.

فيما يلي تابع التقويم الناتج عن عدد القطع التي ليست في مكانها:

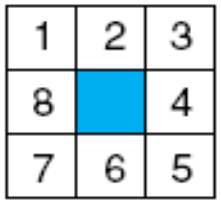
مثال (٦)



Values of f(n) for each state, **6** **4** **6**

where:

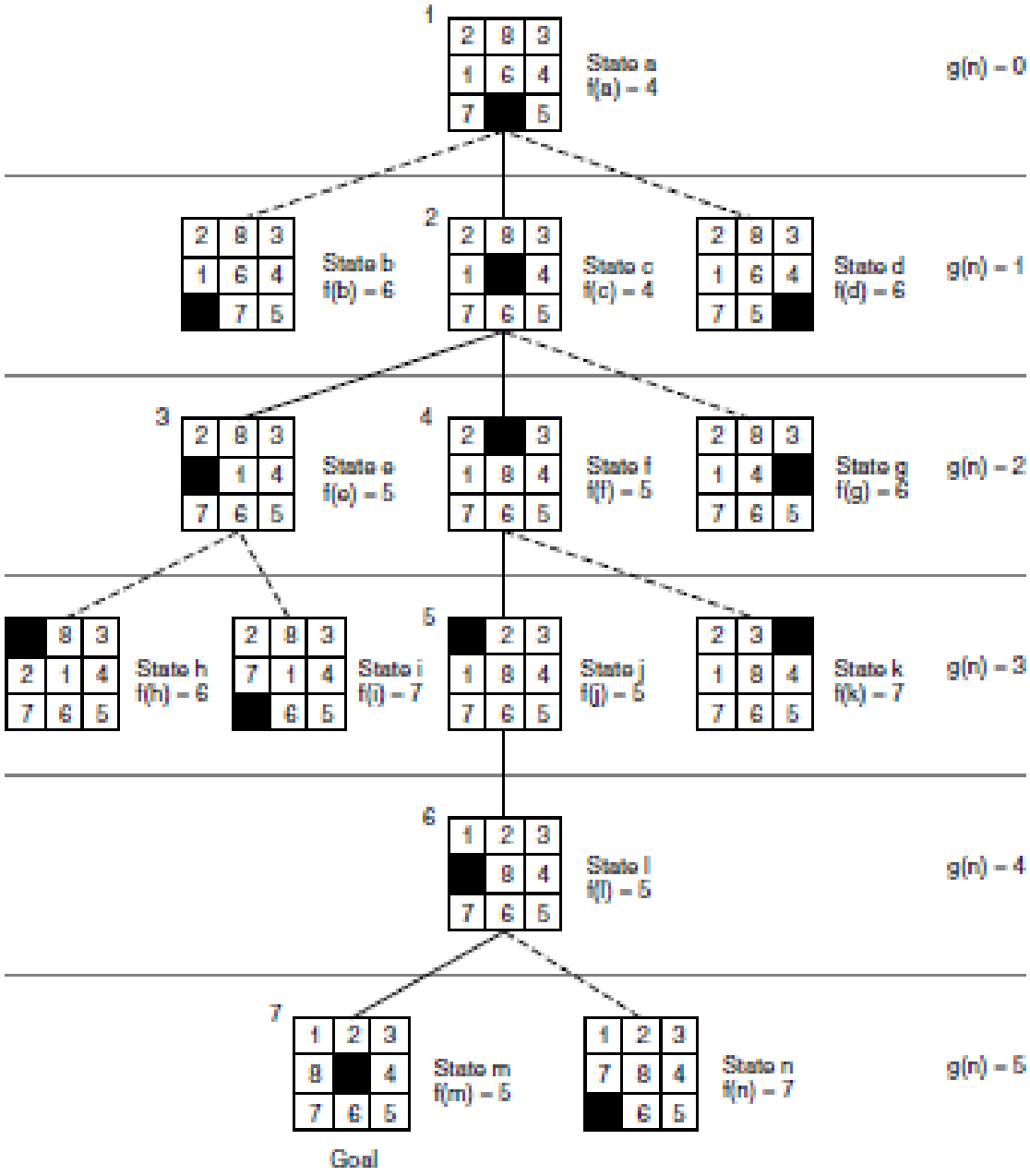
f(n) = g(n) + h(n),
g(n) = actual distance from n to the start state, and
h(n) = number of tiles out of place.



مثال (٦)

سنكتفي فيما يلي بتابع التقويم الناتج عن القطع التي ليست في مكانها، إلا أننا سنضيف إليه تقويما إضافيا يقدم لنا تمييزا إذا ما تساوى التقويم الناجم عن حساب عدد القطع التي ليست في مكانها ، و هو رقم يقيس عمق البحث (المحدد برقم المستوى الذي تقع فيه العقدة قيد الدراسة)مما يضمن لنا متابعة البحث انطلاقا من العقد الأكثر قربا من عقدة الهدف.

مثال (٦)


1
State a
f(a) = 4
g(n) = 0
2
State b
f(b) = 6
State c
f(c) = 4
State d
f(d) = 6
g(n) = 1
3
State e
f(e) = 5
4
State f
f(f) = 5
State g
f(g) = 5
g(n) = 2
State h
f(h) = 6
State i
f(i) = 7
5
State j
f(j) = 5
State k
f(k) = 7
g(n) = 3
6
State l
f(l) = 5
g(n) = 4
7
State m
f(m) = 5
State n
f(n) = 7
g(n) = 5
Goal

مثال (٦)

1. open = [a4];
   closed = [ ]
2. open = [c4, b6, d6];
   closed = [a4]
3. open = [e5, f5, b6, d6, g6];
   closed = [a4, c4]
4. open = [f5, h6, b6, d6, g6, l7];
   closed = [a4, c4, e5]
5. open = [j5, h6, b6, d6, g6, k7, l7];
   closed = [a4, c4, e5, f5]
6. open = [l5, h6, b6, d6, g6, k7, l7];
   closed = [a4, c4, e5, f5, j5]
7. open = [m5, h6, b6, d6, g6, n7, k7, l7];
   closed = [a4, c4, e5, f5, j5, l5]
8. success, m = goal!

خصائص البحث A^*
properties of A^* search

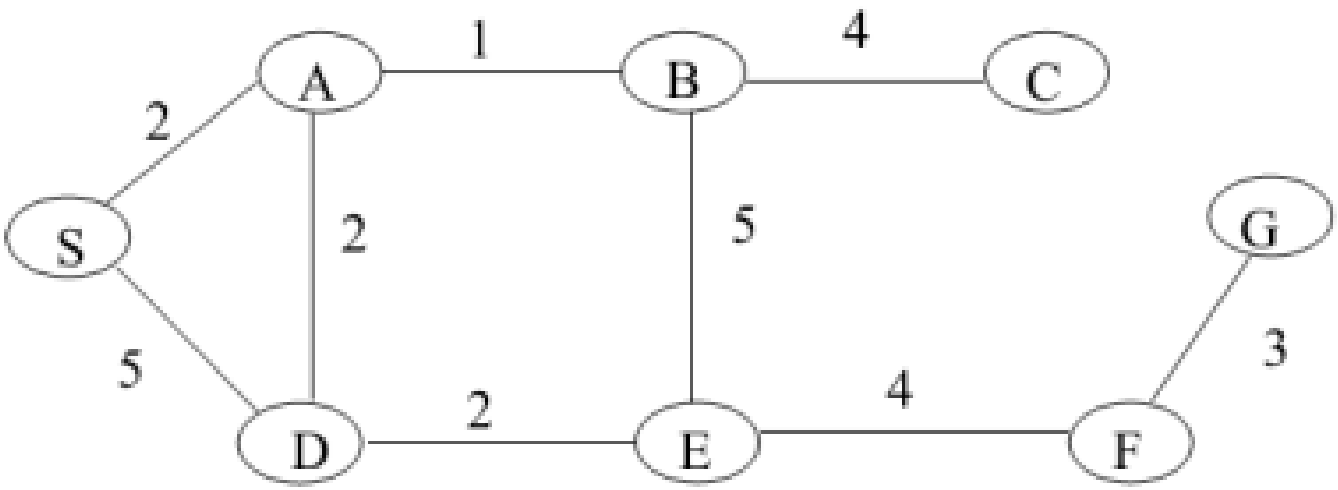
- Completeness: yes
- Optimality: yes (with admissible heuristic)
- Time complexity:

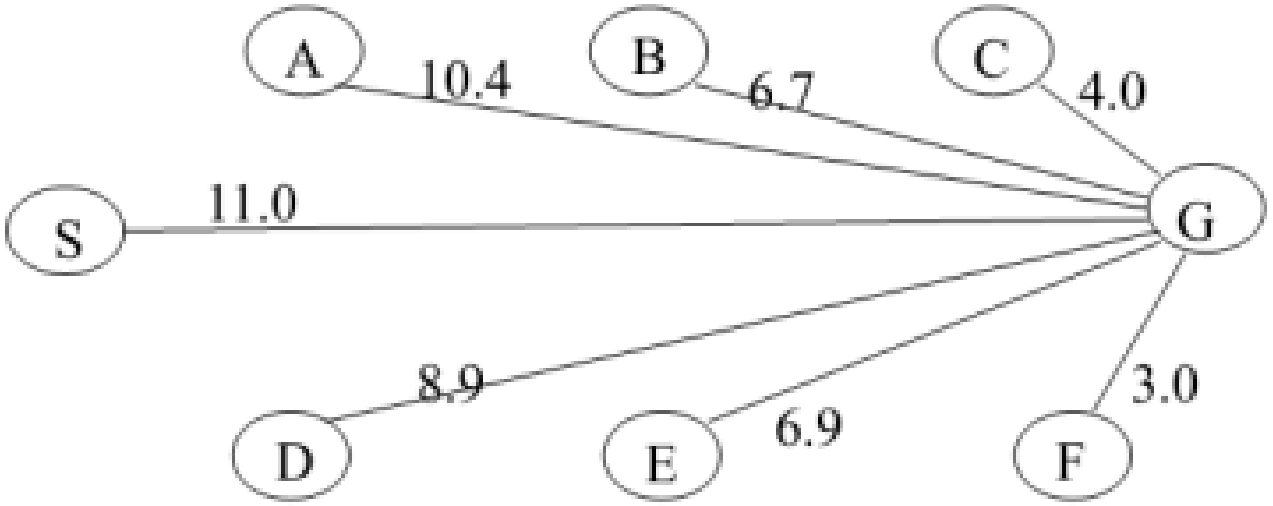
  - order roughly the number of nodes with ***f(n)*** smaller than the cost of optimal path g^*.

- Memory (space) complexity:

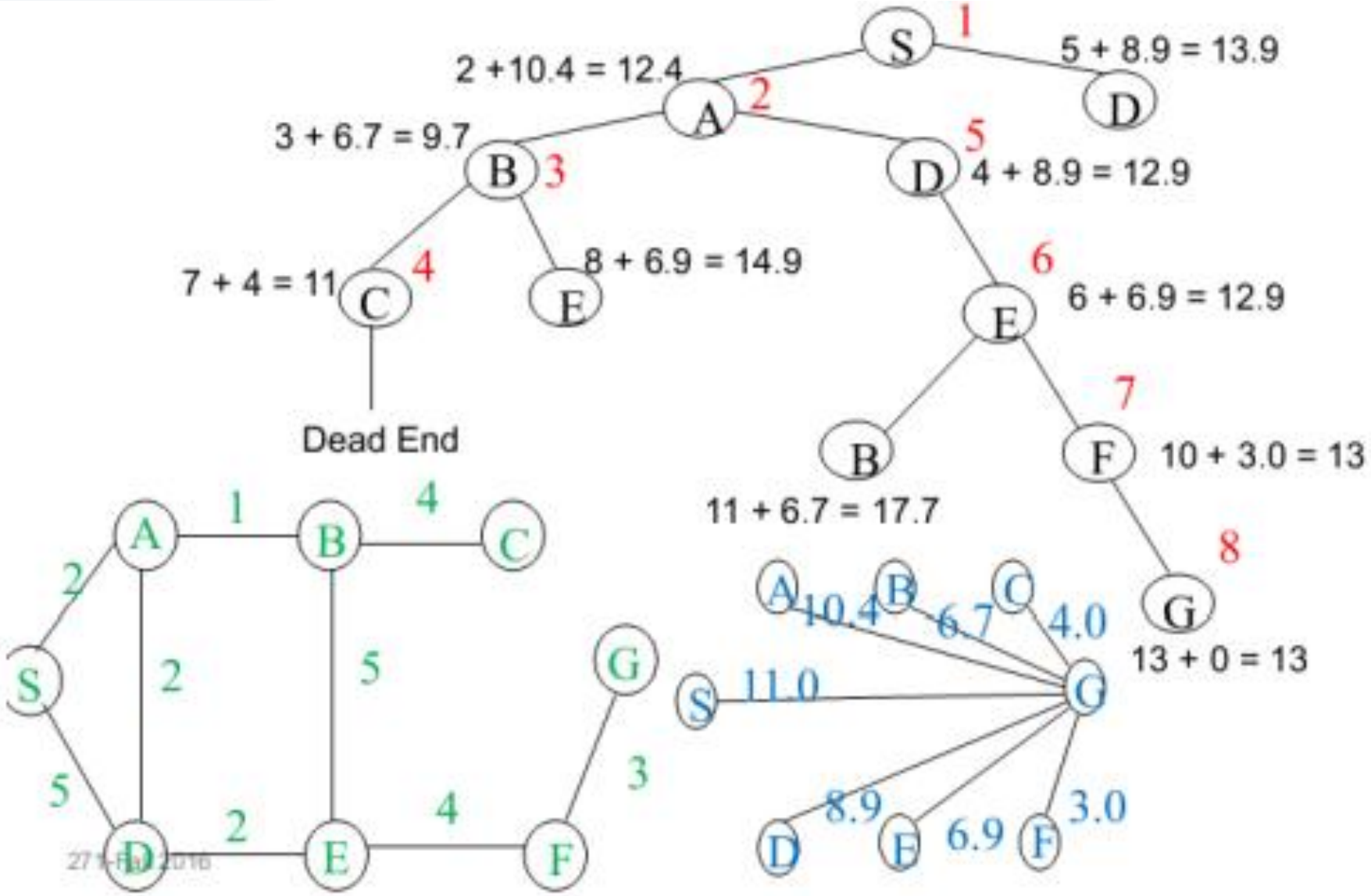
  - same as time complexity (all nodes in the memory)

مثال (۷)

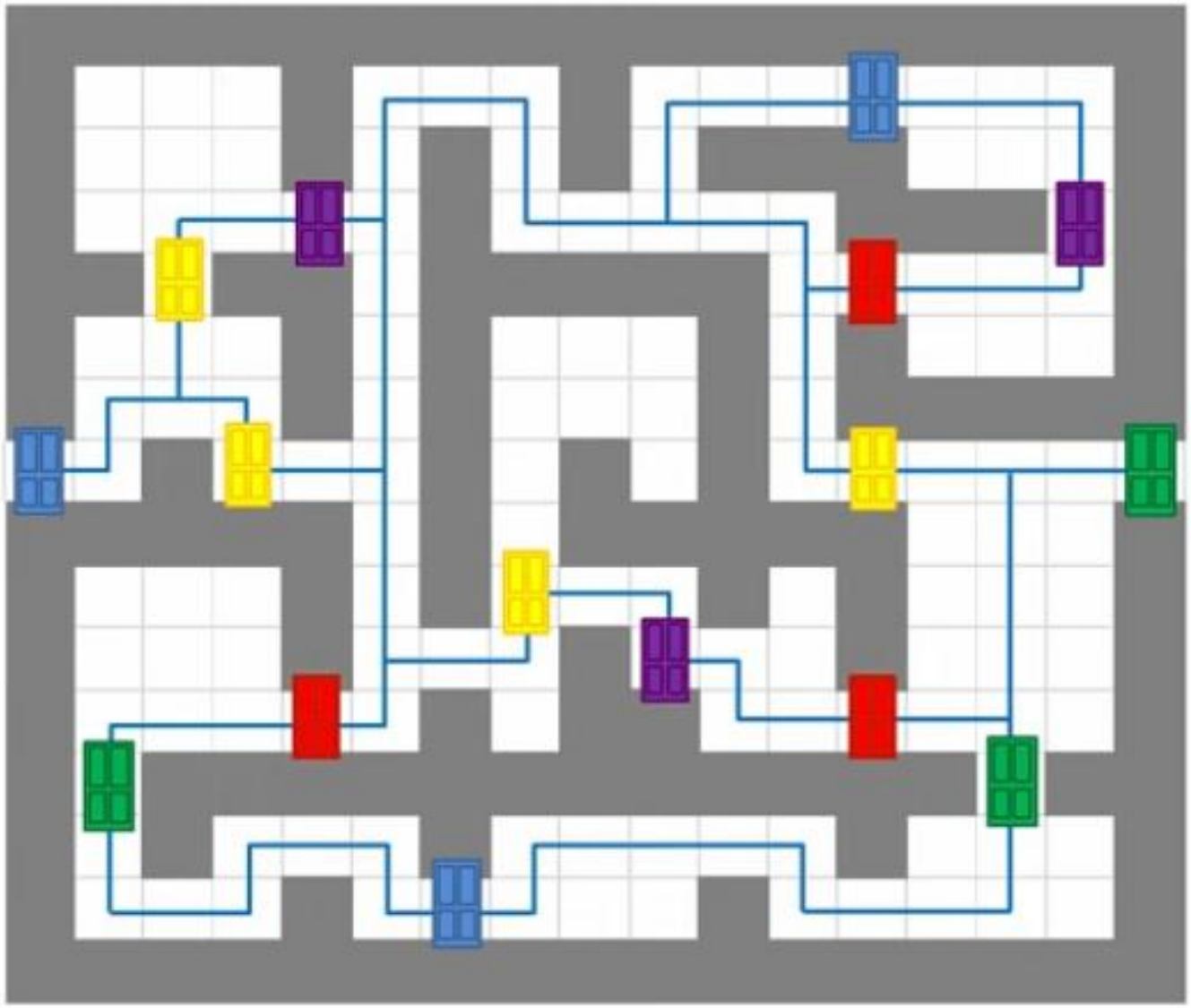




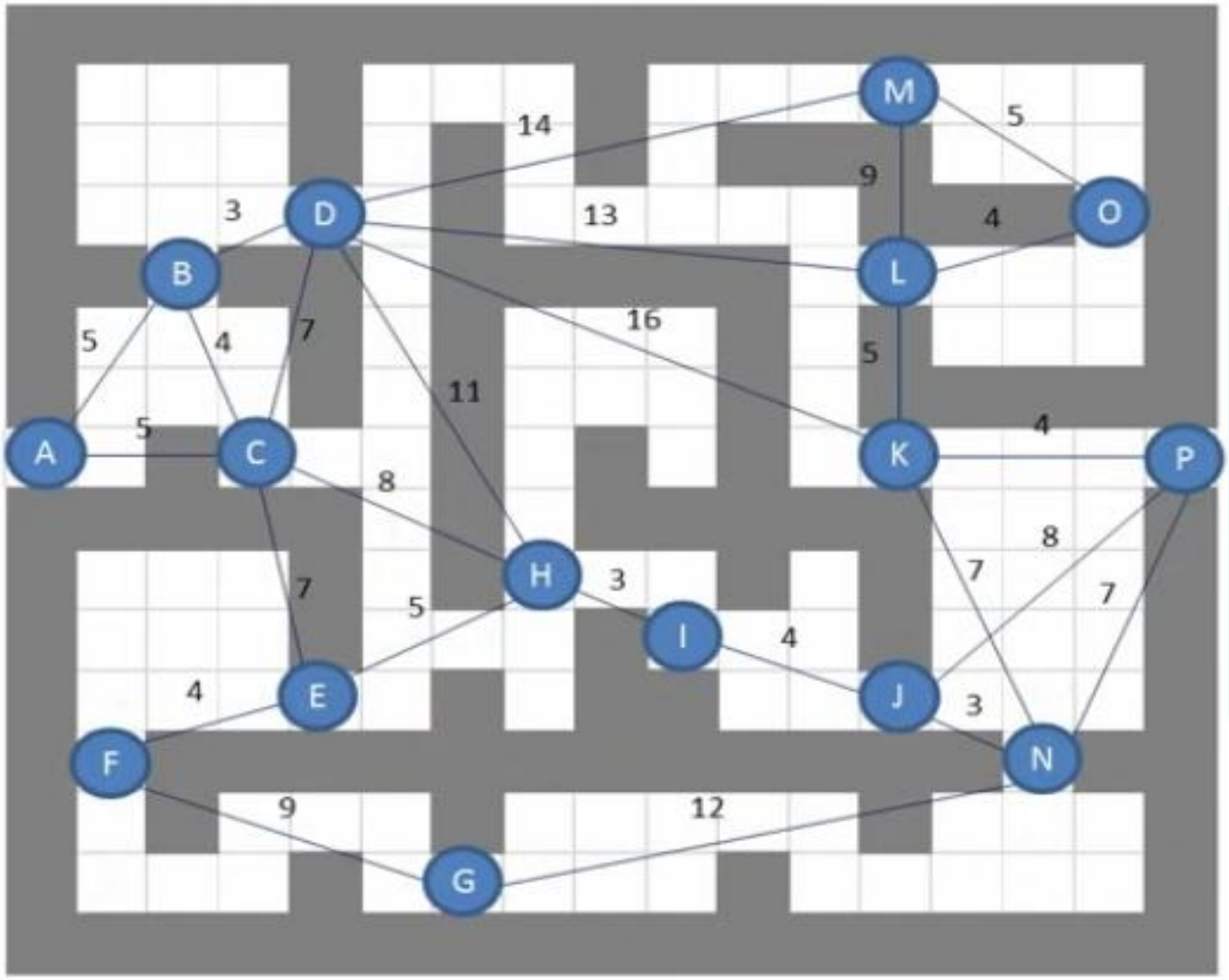
مثال (۷)


S
1
5 + 8.9 = 13.9
D
2 +10.4 = 12.4
A
2
3 + 6.7 = 9.7
B
3
D
5
4 + 8.9 = 12.9
7 + 4 = 11
C
4
8 + 6.9 = 14.9
E
6
6 + 6.9 = 12.9
E
Dead End
7
B
F
10 + 3.0 = 13
11 + 6.7 = 17.7
8
G
13 + 0 = 13
A
B
C
10.4
6.7
4.0
S
11.0
G
8.9
6.9
3.0
D
E
F
A
1
B
4
C
2
S
2
5
G
5
3
4
2
D
E
F

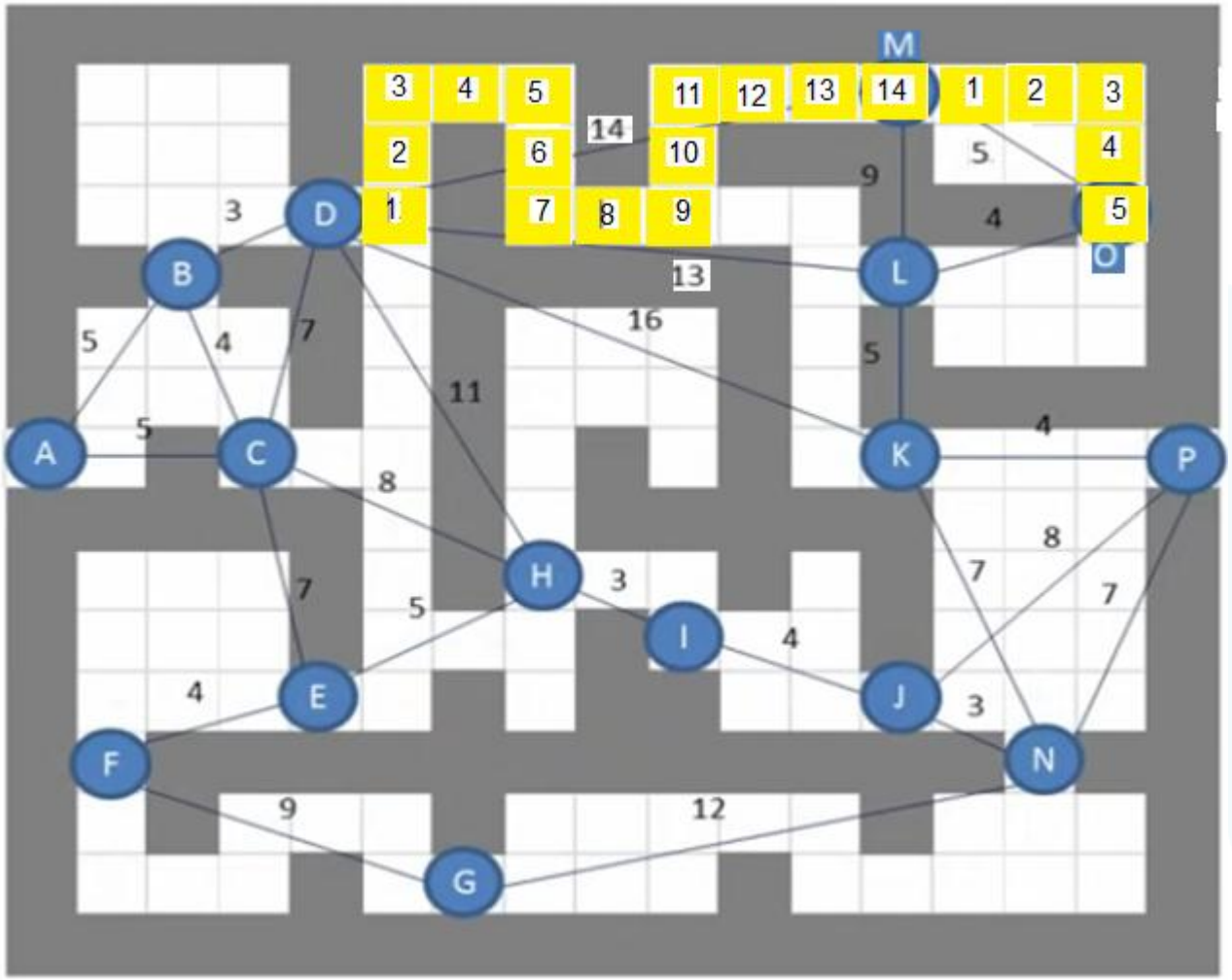
مثال (٨)



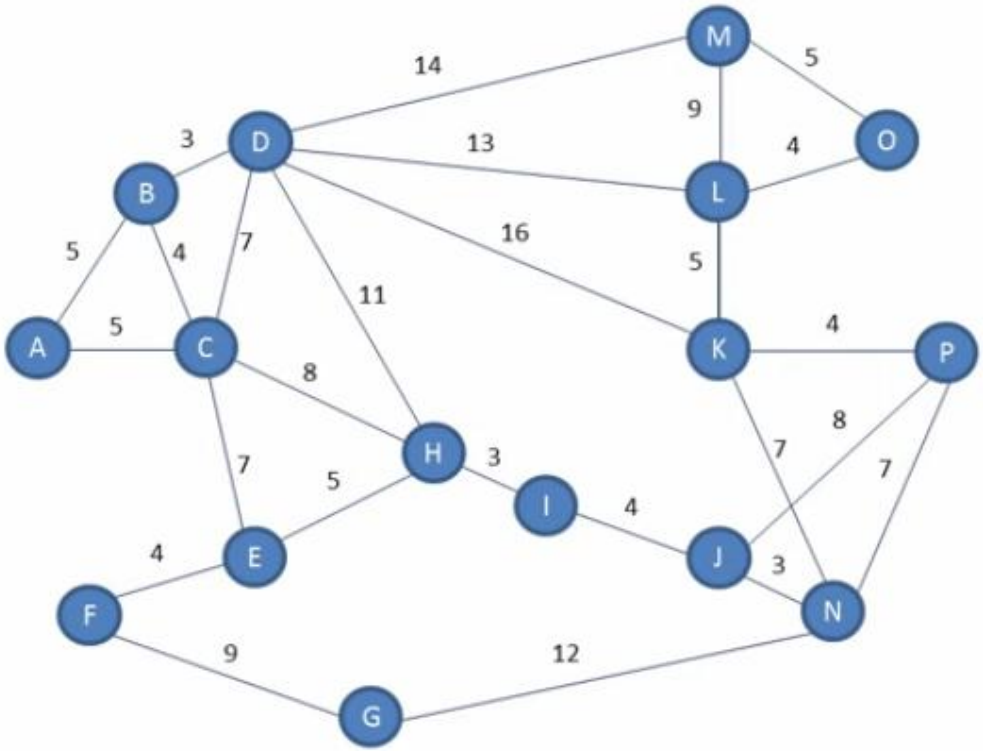
مثال (٨)



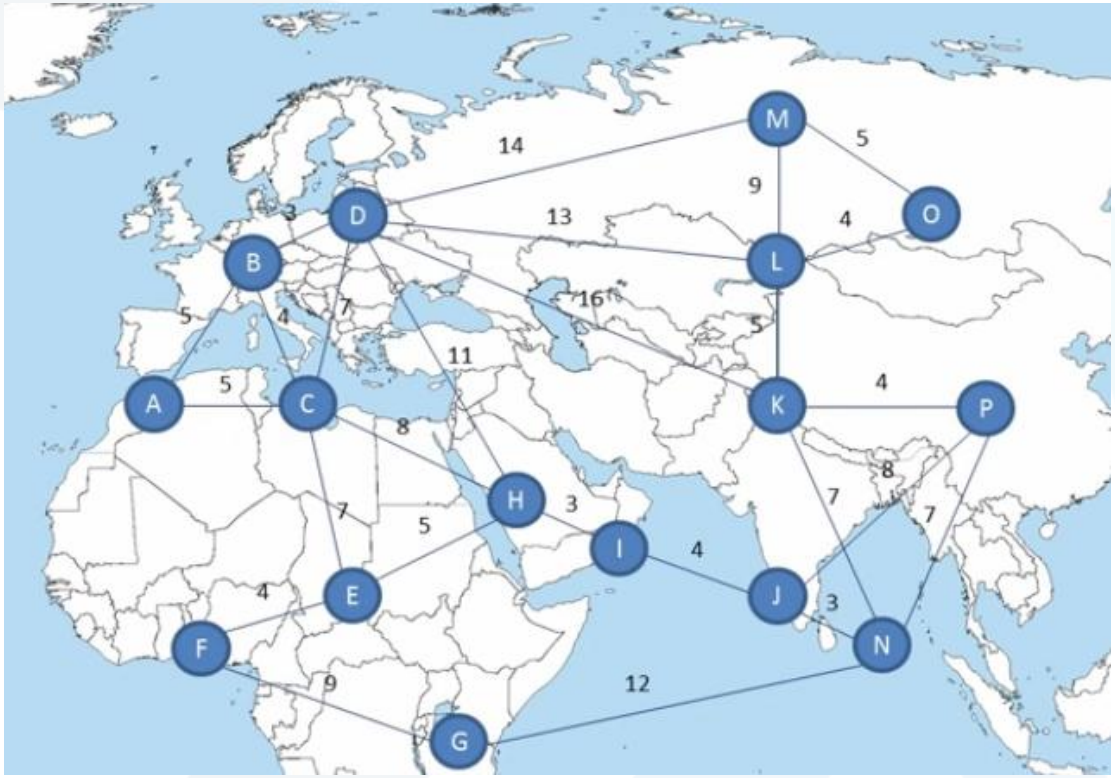
مثال (٨)


M
3 4 5 11 12 13 14 1 2 3
2 6 14 10 5 4
1 7 8 9 4 5
O
A
B
C
D
E
F
G
H
I
J
K
L
N
P
3
5
4
7
5
11
16
13
9
5
4
8
7
5
3
4
7
8
7
4
3
9
12

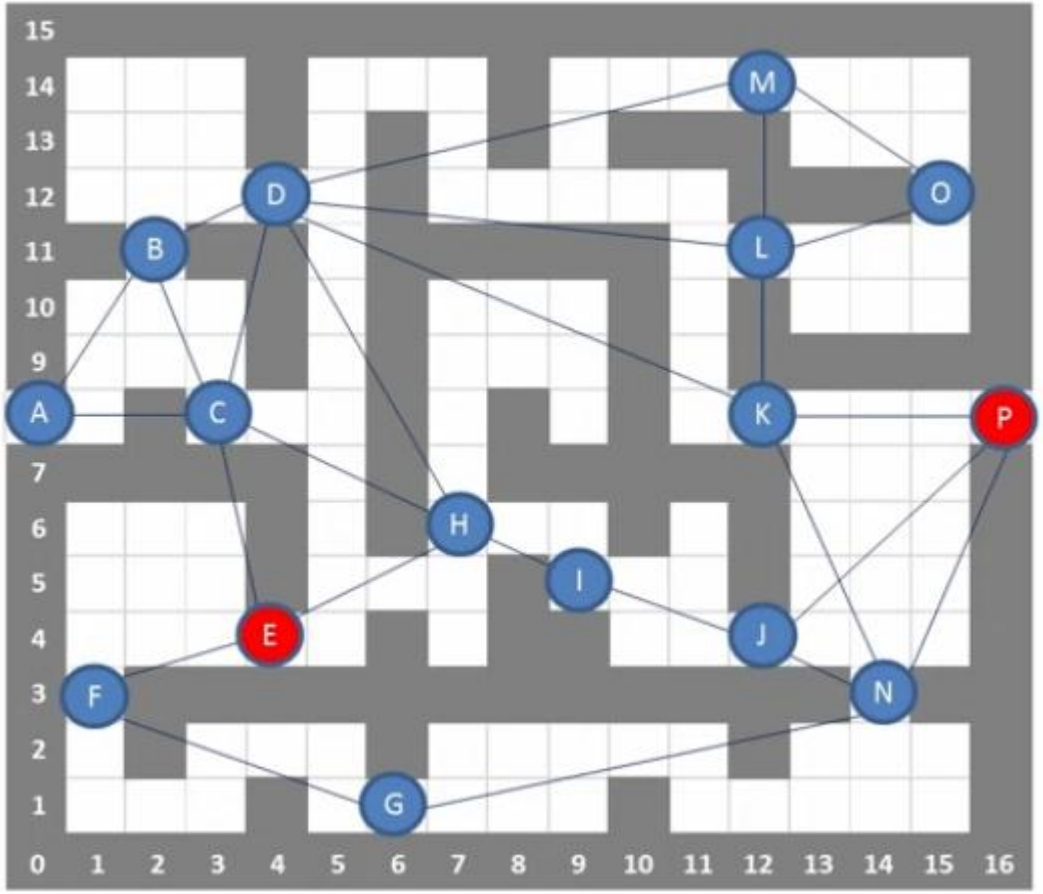
مثال (٨)



مثال (۸)

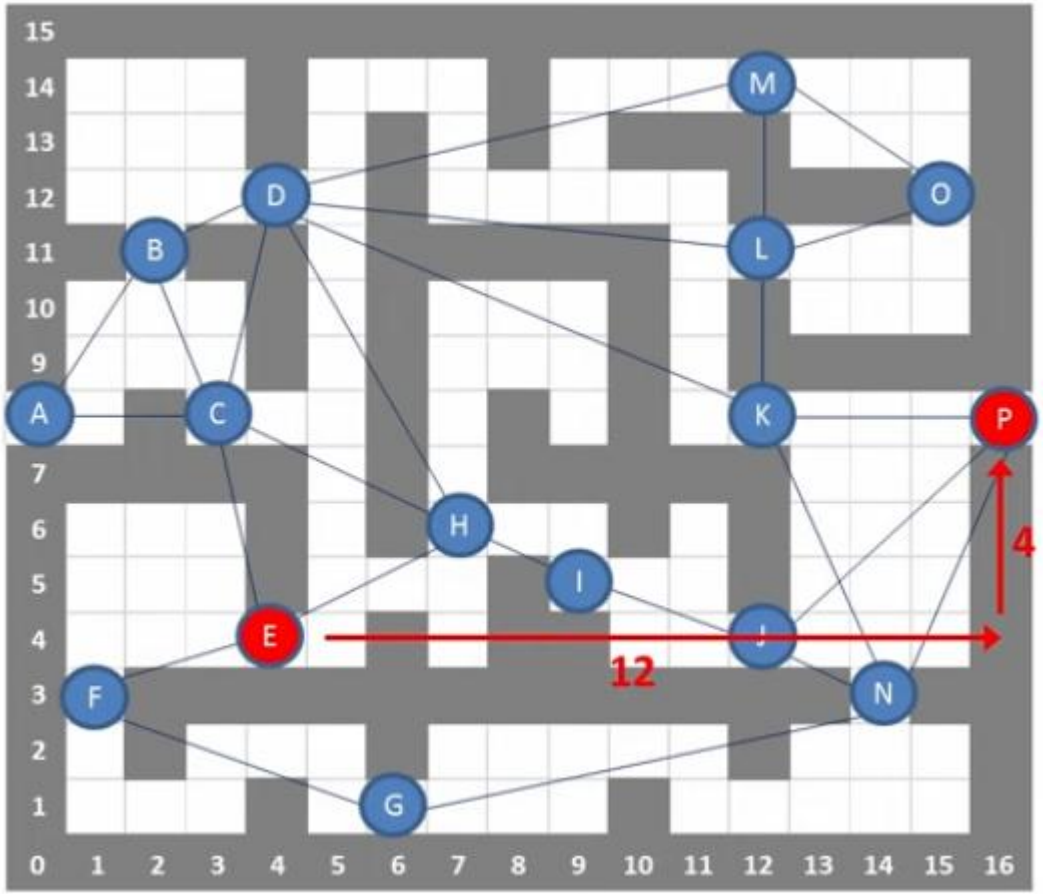


مثال (۸)

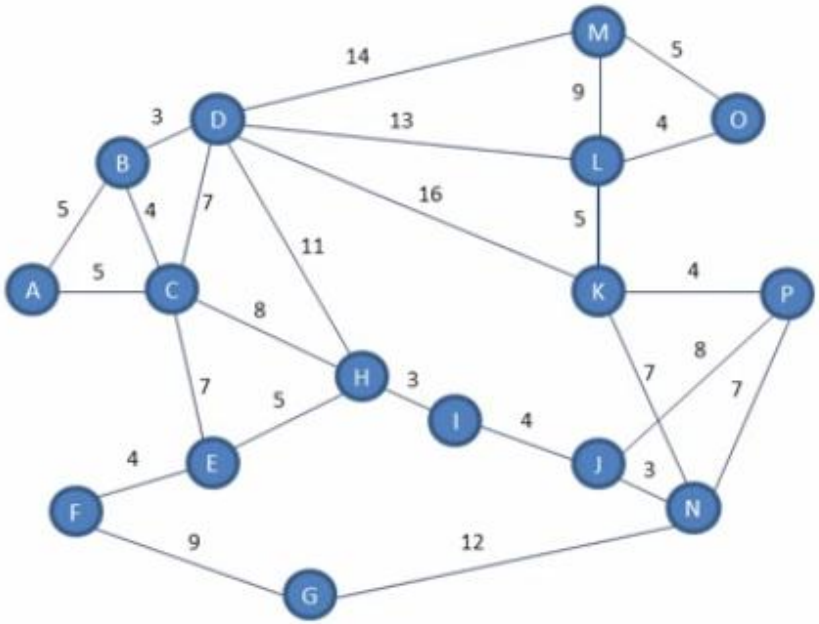

15
14
13
12
11
10
9
7
6
5
4
3
2
1
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
M
D
O
B
L
A
C
K
P
H
I
E
J
F
N
G

مثال (٨)

- الاجتهادية للعقدة E هي: 16=4+12 و هكذا.....



مثال (٨)

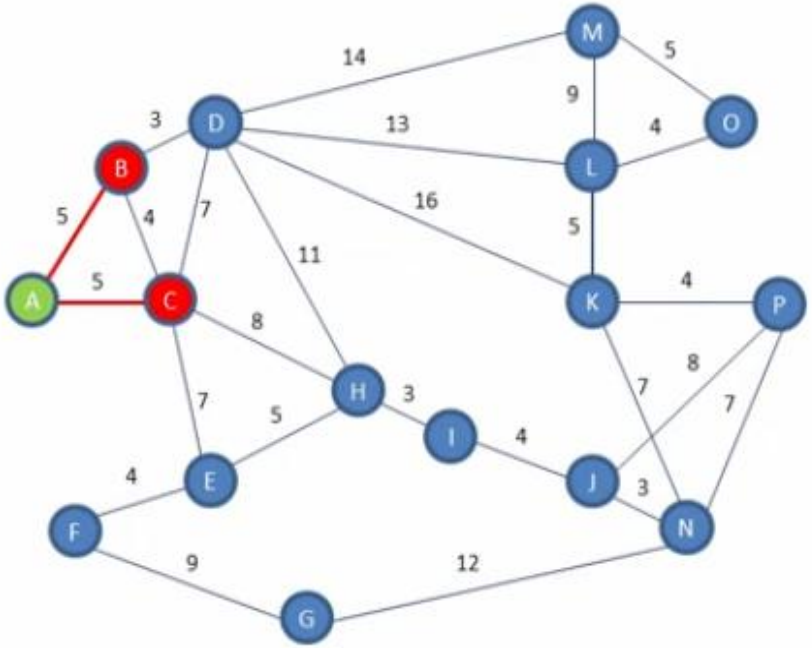


Vertex	Distance from A (g)	Heuristic distance (h)	f = g + h	Previous vertex
A		16		
B		17		
C		13		
D		16		
E		16		
F		20		
G		17		
H		11		
I		10		
J		8		
K		4		
L		7		
M		10		
N		7		
O		5		
P		0		

مثال (٨)

Open A B C

Closed

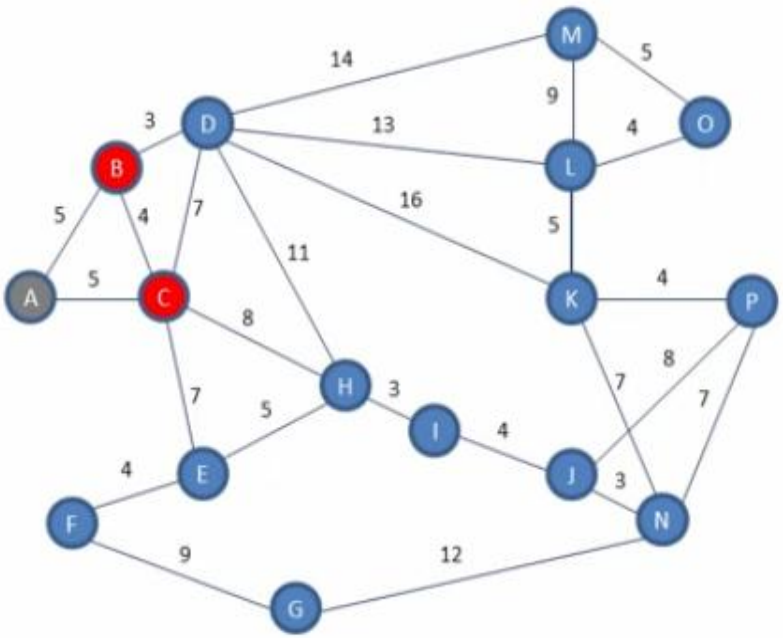


Vertex	Distance from A (g)	Heuristic distance (h)	f = g + h	Previous vertex
A	0	16	16	
B	5	17	22	A
C	5	13	18	A
D		16		
E		16		
F		20		
G		17		
H		11		
I		10		
J		8		
K		4		
L		7		
M		10		
N		7		
O		5		
P		0		

مثال (٨)

Open B C

Closed A

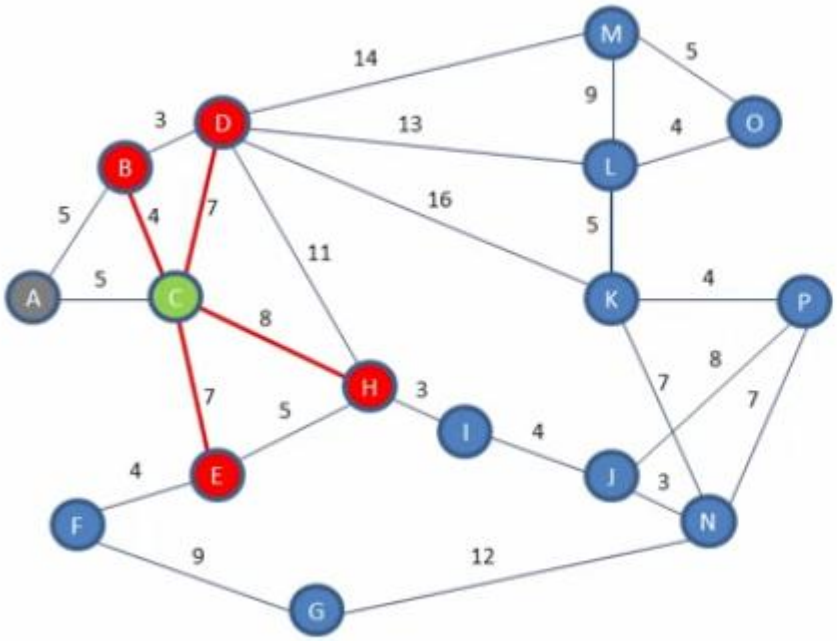


Vertex	Distance from A (g)	Heuristic distance (h)	f = g + h	Previous vertex
A	0	16	16	
B	5	17	22	A
C	5	13	18	A
D		16		
E		16		
F		20		
G		17		
H		11		
I		10		
J		8		
K		4		
L		7		
M		10		
N		7		
O		5		
P		0		

مثال (٨)

Open B C D H E

Closed A

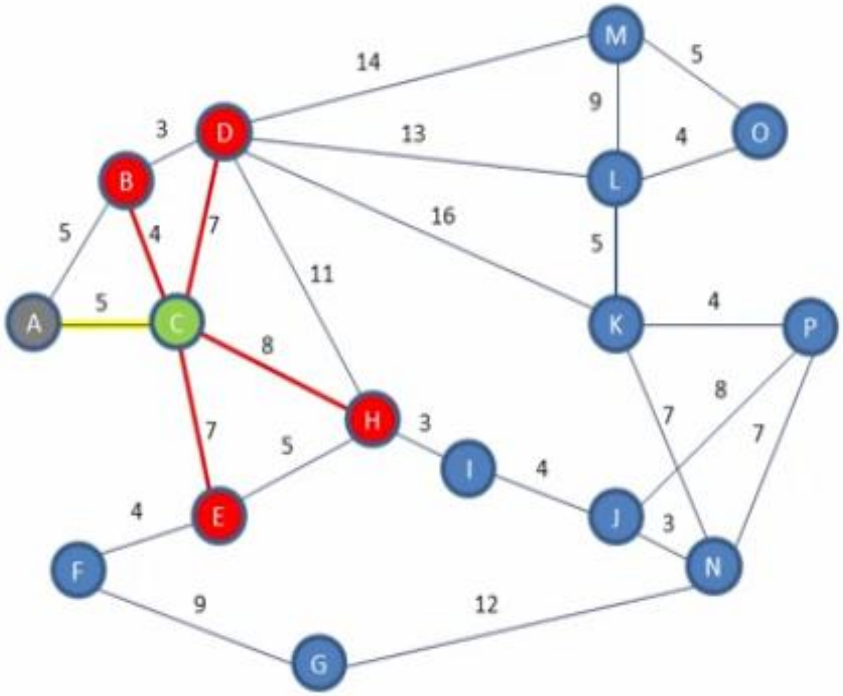


Vertex	Distance from A (g)	Heuristic distance (h)	f = g + h	Previous vertex
A	0	16	16	
B	5	17	22	A
C	5	13	18	A
D		16		
E		16		
F		20		
G		17		
H		11		
I		10		
J		8		
K		4		
L		7		
M		10		
N		7		
O		5		
P		0		

مثال (٨)

Open B C D H E

Closed A

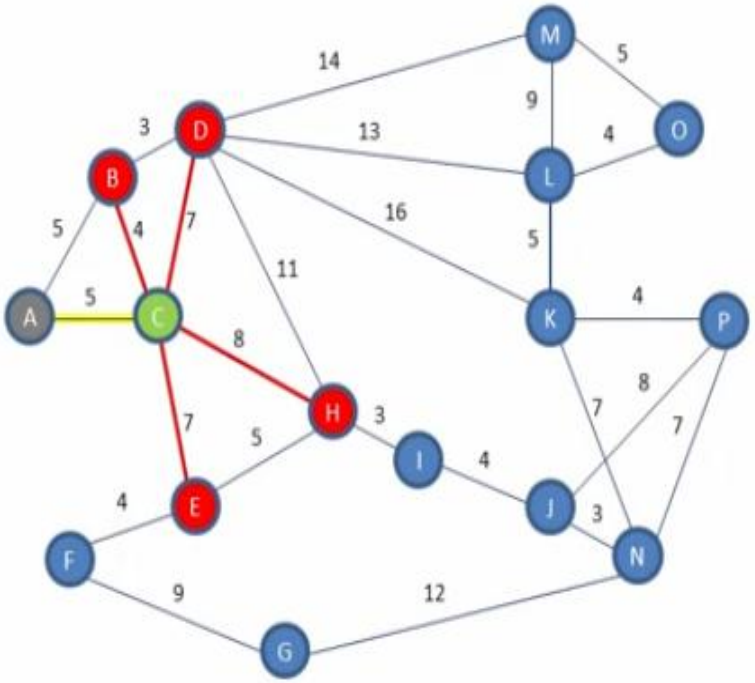


Vertex	Distance from A (g)	Heuristic distance (h)	f = g + h	Previous vertex
A	0	16	16	
B	5 **9**	17	22 **26**	A **C**
C	5	13	18	A
D	**12**	16	**28**	**C**
E	**12**	16	**28**	**C**
F		20		
G		17		
H	**13**	11	**24**	**C**
I		10		
J		8		
K		4		
L		7		
M		10		
N		7		
O		5		
P		0		

مثال (۸)

Open B C D H E

Closed A

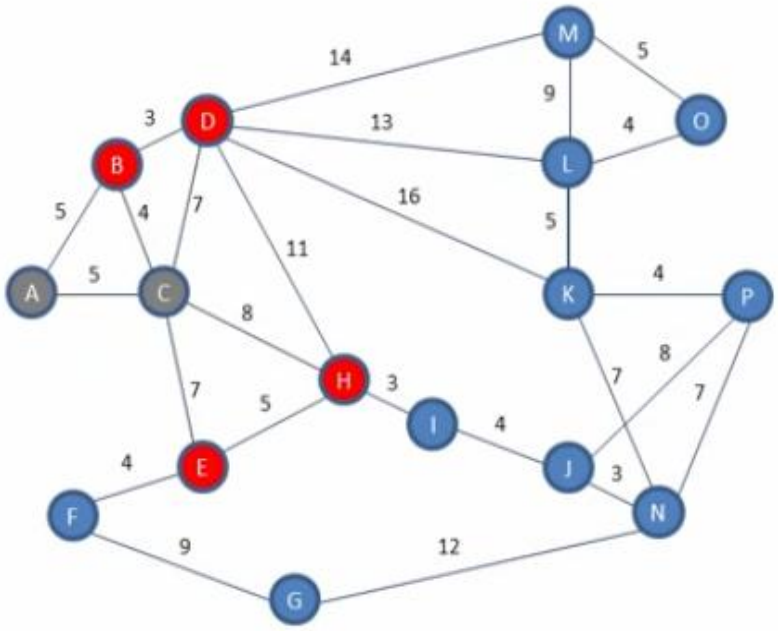


Vertex	Distance from A (g)	Heuristic distance (h)	f = g + h	Previous vertex
A	0	16	16	
B	5 9	17	22 26	A C
C	5	13	18	A
D	12	16	28	C
E	12	16	28	C
F		20		
G		17		
H	13	11	24	C
I		10		
J		8		
K		4		
L		7		
M		10		
N		7		
O		5		
P		0		

مثال (٨)

Open B D H E

Closed A C

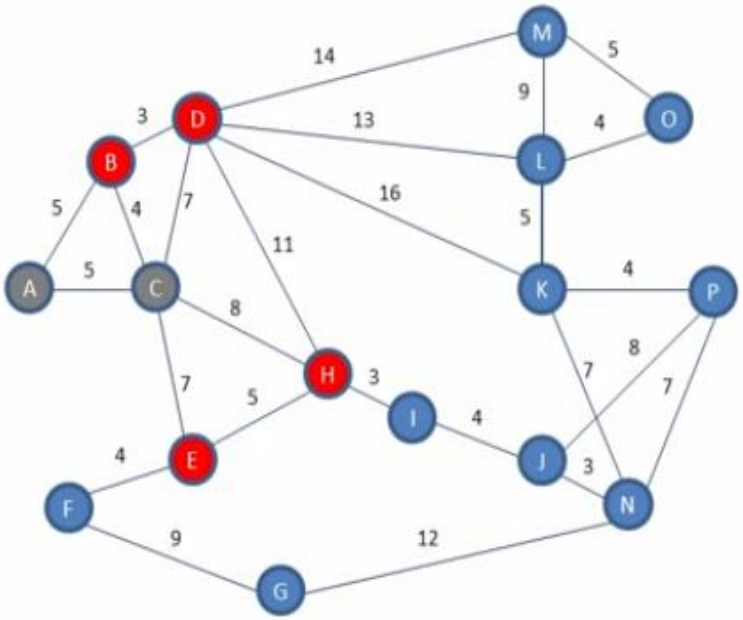


Vertex	Distance from A (g)	Heuristic distance (h)	f = g + h	Previous vertex
A	0	16	16	
B	5	17	22	A
C	5	13	18	A
D	12	16	28	C
E	12	16	28	C
F		20		
G		17		
H	13	11	24	C
I		10		
J		8		
K		4		
L		7		
M		10		
N		7		
O		5		
P		0		

مثال (۸)

Open B D H E

Closed A C

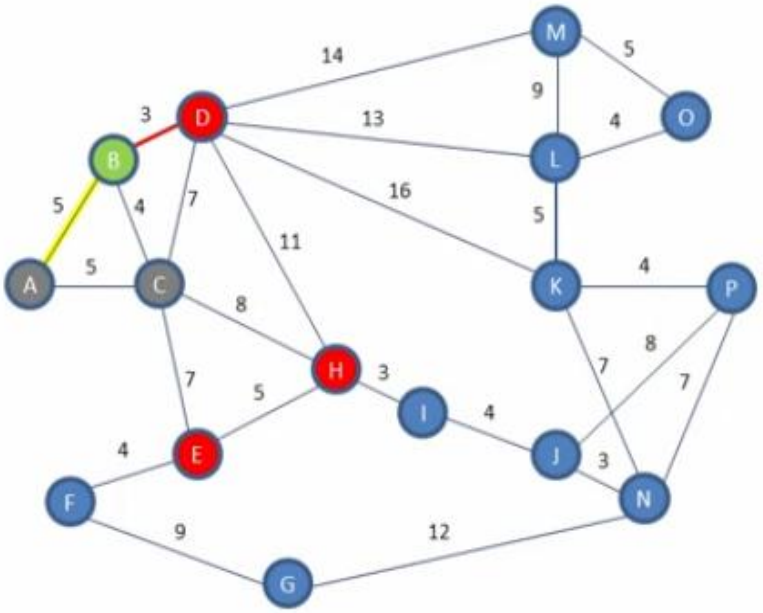


Vertex	Distance from A (g)	Heuristic distance (h)	f = g + h	Previous vertex
A	0	16	16	
B	5	17	22	A
C	5	13	18	A
D	12	16	28	C
E	12	16	28	C
F		20		
G		17		
H	13	11	24	C
I		10		
J		8		
K		4		
L		7		
M		10		
N		7		
O		5		
P		0		

مثال (۸)

Open B D H E

Closed A C

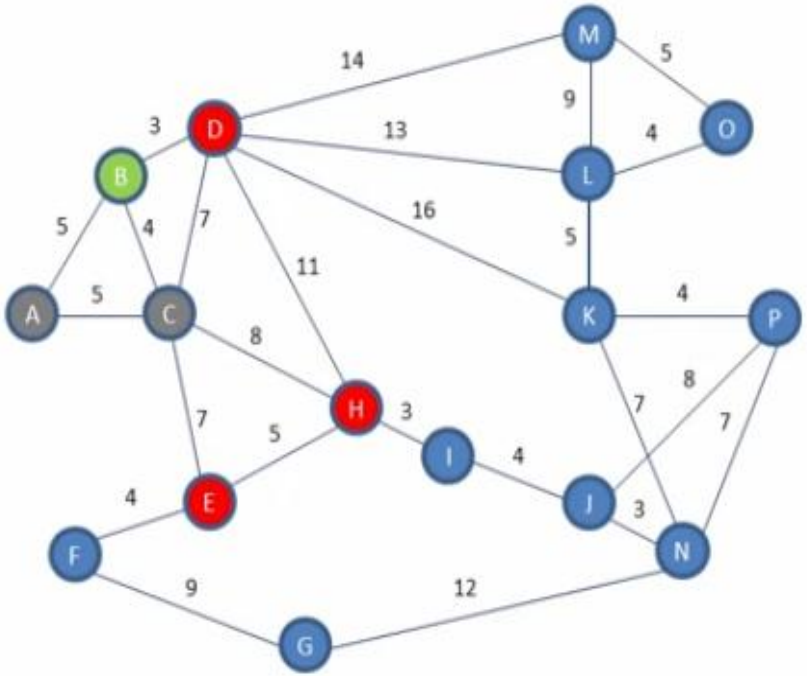


Vertex	Distance from A (g)	Heuristic distance (h)	f = g + h	Previous vertex
A	0	16	16	
B	5	17	22	A
C	5	13	18	A
D	12 8	16	28 24	C B
E	12	16	28	C
F		20		
G		17		
H	13	11	24	C
I		10		
J		8		
K		4		
L		7		
M		10		
N		7		
O		5		
P		0		

مثال (۸)

Open B D H E

Closed A C

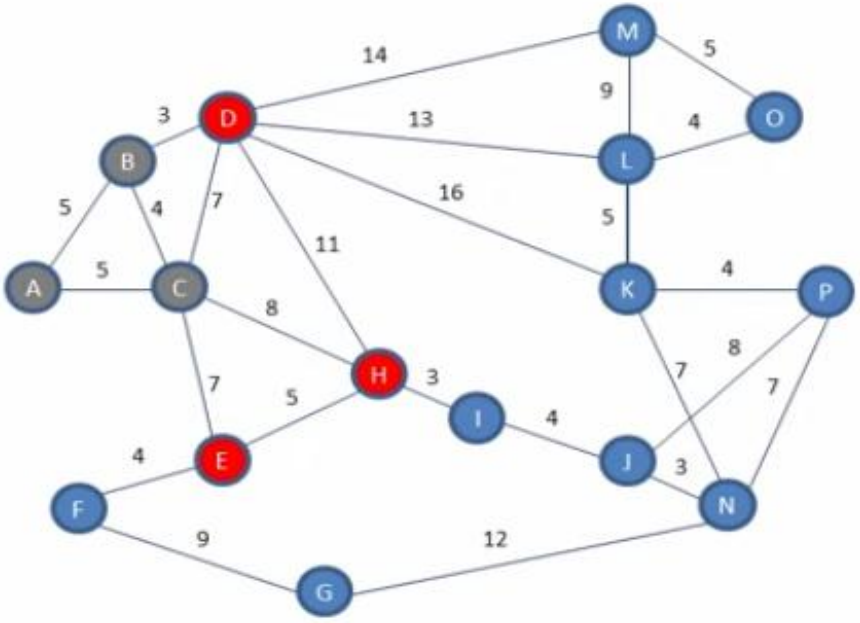


Vertex	Distance from A (g)	Heuristic distance (h)	f = g + h	Previous vertex
A	0	16	16	
B	5	17	22	A
C	5	13	18	A
D	8	16	24	B
E	12	16	28	C
F		20		
G		17		
H	13	11	24	C
I		10		
J		8		
K		4		
L		7		
M		10		
N		7		
O		5		
P		0		

مثال (٨)

Open D H E

Closed A C B

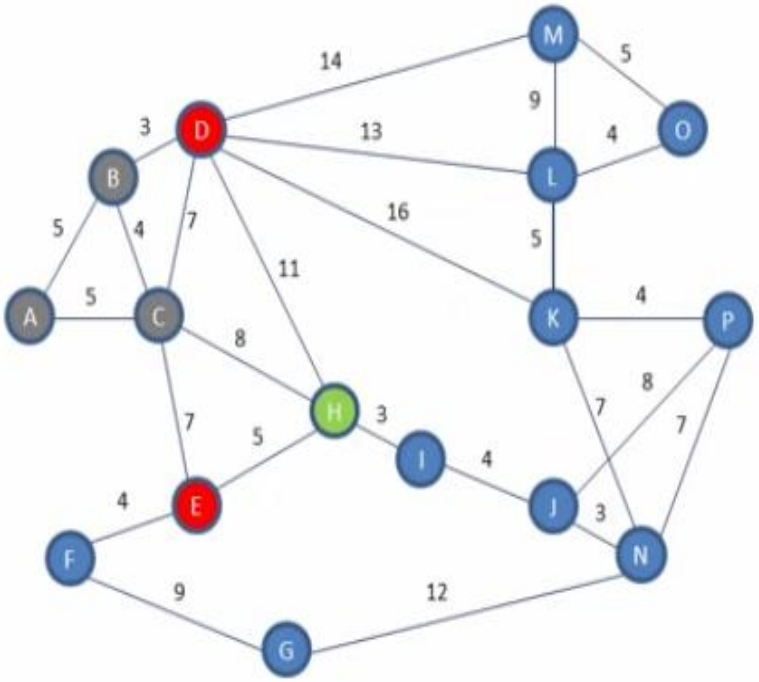


Vertex	Distance from A (g)	Heuristic distance (h)	f = g + h	Previous vertex
A	0	16	16	
B	5	17	22	A
C	5	13	18	A
D	8	16	24	B
E	12	16	28	C
F		20		
G		17		
H	13	11	24	C
I		10		
J		8		
K		4		
L		7		
M		10		
N		7		
O		5		
P		0		

مثال (۸)

Open D H E

Closed A C B

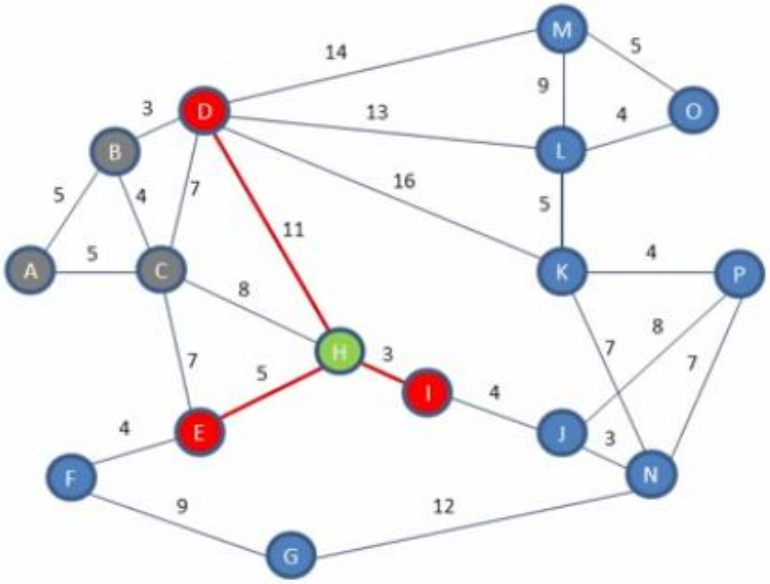


Vertex	Distance from A (g)	Heuristic distance (h)	f = g + h	Previous vertex
A	0	16	16	
B	5	17	22	A
C	5	13	18	A
D	8	16	24	B
E	12	16	28	C
F		20		
G		17		
H	13	11	24	C
I		10		
J		8		
K		4		
L		7		
M		10		
N		7		
O		5		
P		0		

مثال (٨)

Open D H E I

Closed A C B

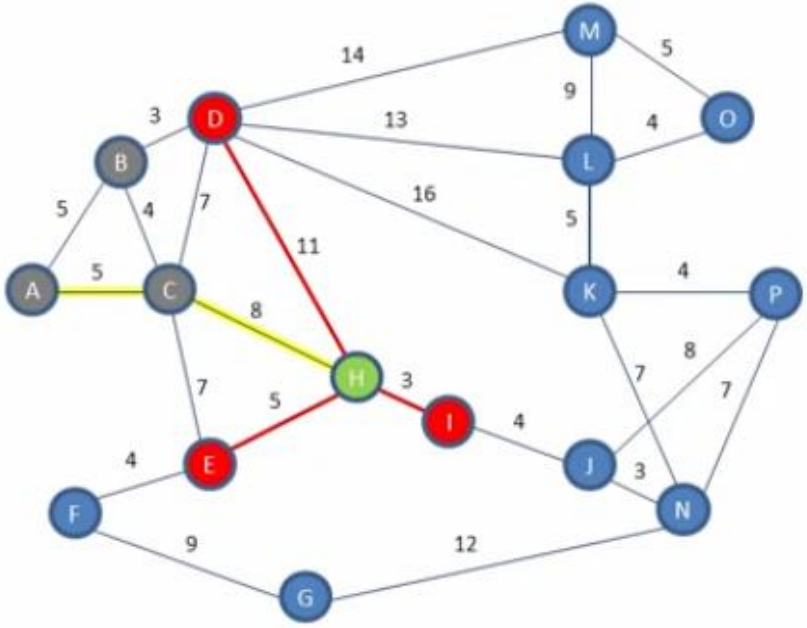


Vertex	Distance from A (g)	Heuristic distance (h)	f = g + h	Previous vertex
A	0	16	16	
B	5	17	22	A
C	5	13	18	A
D	8	16	24	B
E	12	16	28	C
F		20		
G		17		
H	13	11	24	C
I		10		
J		8		
K		4		
L		7		
M		10		
N		7		
O		5		
P		0		

مثال (٨)

Open D H E I

Closed A C B

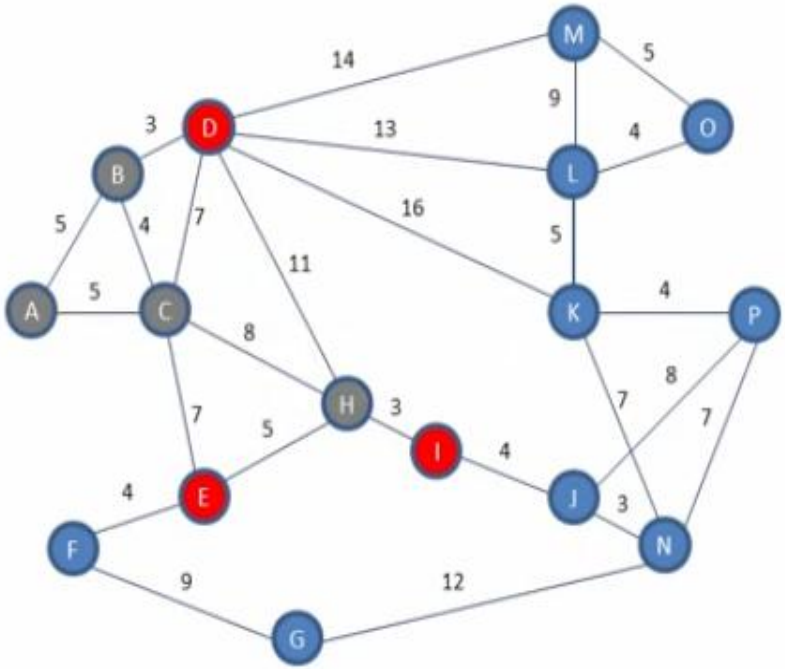


Vertex	Distance from A (g)	Heuristic distance (h)	f = g + h	Previous vertex
A	0	16	16	
B	5	17	22	A
C	5	13	18	A
D	8 **24**	16	24 **40**	B **H**
E	12 **18**	16	28 **34**	C **H**
F		20		
G		17		
H	13	11	24	C
I	**16**	10	**26**	**H**
J		8		
K		4		
L		7		
M		10		
N		7		
O		5		
P		0		

مثال (۸)

Open D E I

Closed A C B H

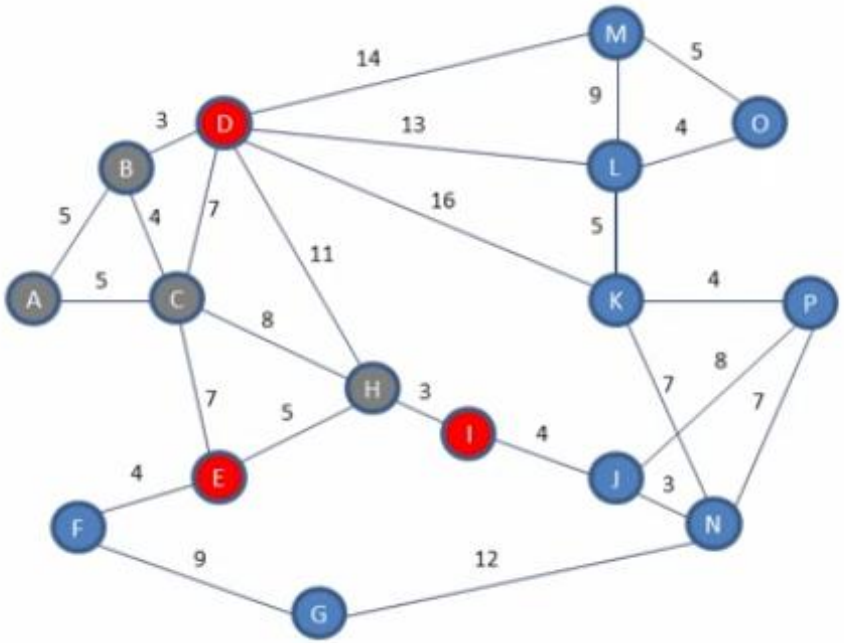


Vertex	Distance from A (g)	Heuristic distance (h)	f = g + h	Previous vertex
A	0	16	16	
B	5	17	22	A
C	5	13	18	A
D	8	16	24	B
E	12	16	28	C
F		20		
G		17		
H	13	11	24	C
I	16	10	26	H
J		8		
K		4		
L		7		
M		10		
N		7		
O		5		
P		0		

مثال (۸)

Open D E I

Closed A C B H

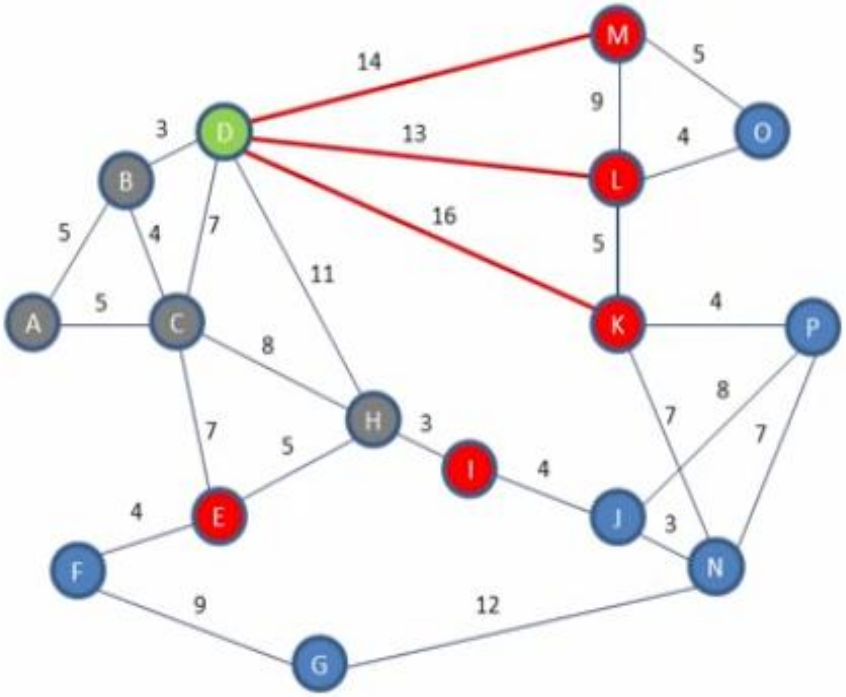


Vertex	Distance from A (g)	Heuristic distance (h)	f = g + h	Previous vertex
A	0	16	16	
B	5	17	22	A
C	5	13	18	A
D	8	16	24	B
E	12	16	28	C
F		20		
G		17		
H	13	11	24	C
I	16	10	26	H
J		8		
K		4		
L		7		
M		10		
N		7		
O		5		
P		0		

مثال (٨)

Open D E I M L K

Closed A C B H

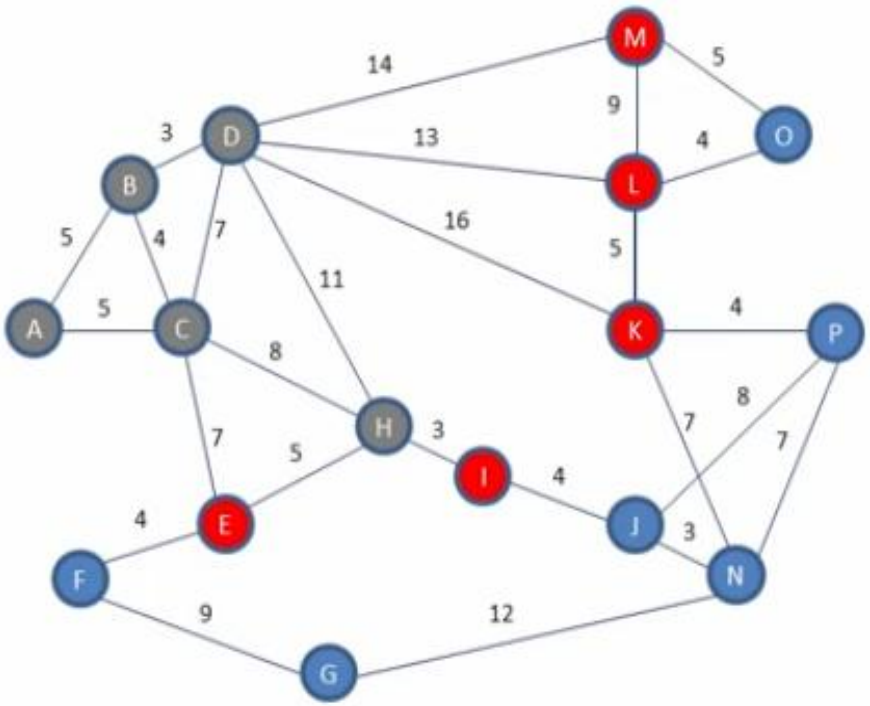


Vertex	Distance from A (g)	Heuristic distance (h)	f = g + h	Previous vertex
A	0	16	16	
B	5	17	22	A
C	5	13	18	A
D	8	16	24	B
E	12	16	28	C
F		20		
G		17		
H	13	11	24	C
I	16	10	26	H
J		8		
K		4		
L		7		
M		10		
N		7		
O		5		
P		0		

مثال (٨)

Open E I M L K

Closed A C B H D

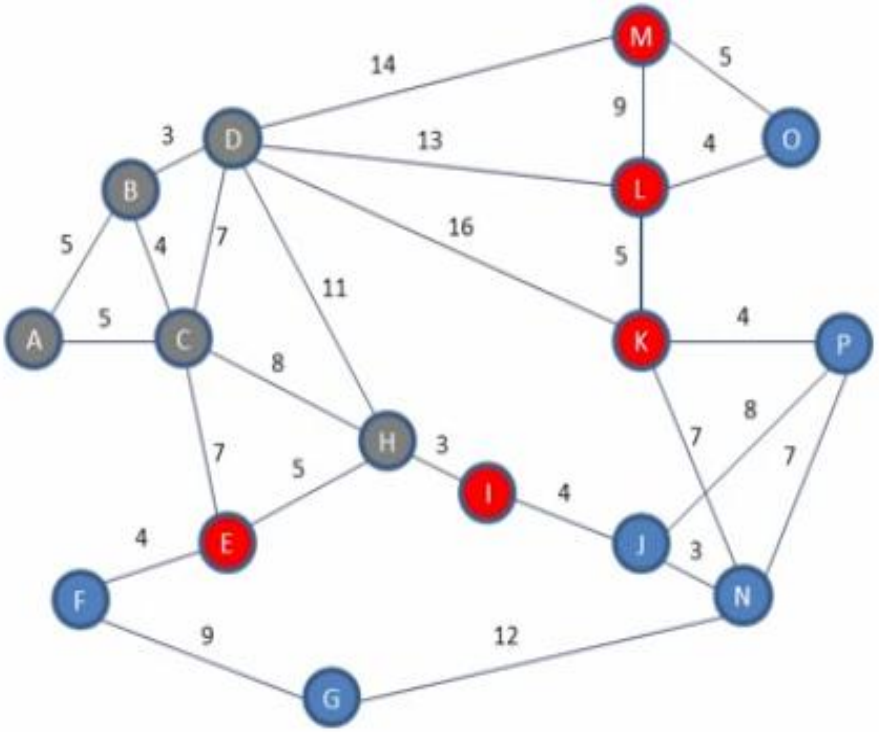


Vertex	Distance from A (g)	Heuristic distance (h)	f = g + h	Previous vertex
A	0	16	16	
B	5	17	22	A
C	5	13	18	A
D	8	16	24	B
E	12	16	28	C
F		20		
G		17		
H	13	11	24	C
I	16	10	26	H
J		8		
K	24	4	28	D
L	21	7	28	D
M	22	10	32	D
N		7		
O		5		
P		0		

مثال (۸)

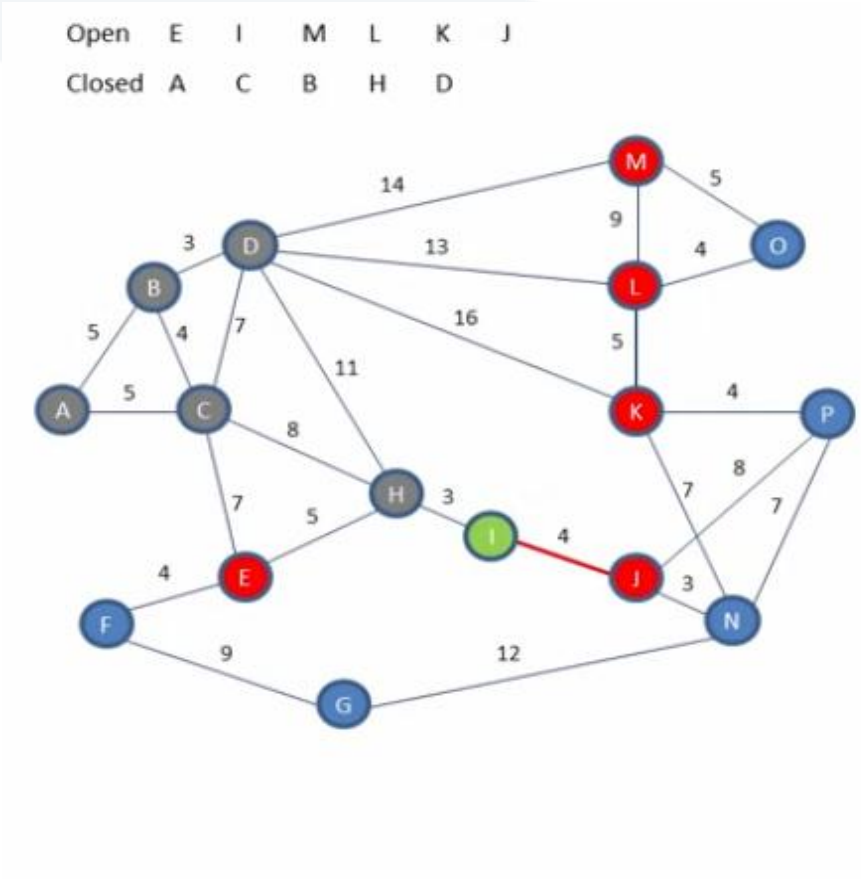
Open E I M L K

Closed A C B H D



Vertex	Distance from A (g)	Heuristic distance (h)	f = g + h	Previous vertex
A	0	16	16	
B	5	17	22	A
C	5	13	18	A
D	8	16	24	B
E	12	16	28	C
F		20		
G		17		
H	13	11	24	C
I	16	10	26	H
J		8		
K	24	4	28	D
L	21	7	28	D
M	22	10	32	D
N		7		
O		5		
P		0		

مثال (۸)

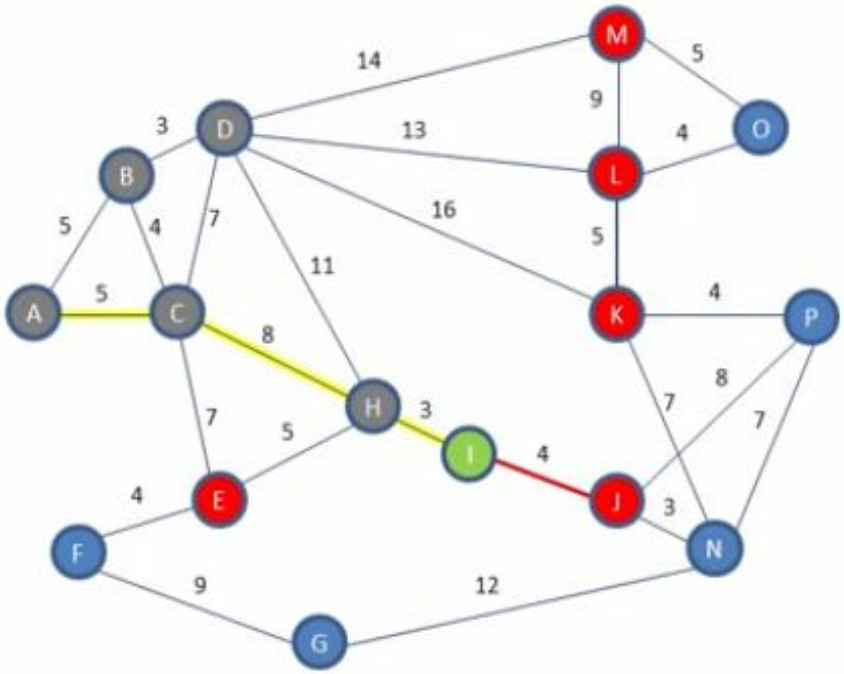


Vertex	Distance from A (g)	Heuristic distance (h)	f = g + h	Previous vertex
A	0	16	16	
B	5	17	22	A
C	5	13	18	A
D	8	16	24	B
E	12	16	28	C
F		20		
G		17		
H	13	11	24	C
I	16	10	26	H
J		8		
K	24	4	28	D
L	21	7	28	D
M	22	10	32	D
N		7		
O		5		
P		0		

مثال (٨)

Open E I M L K J

Closed A C B H D

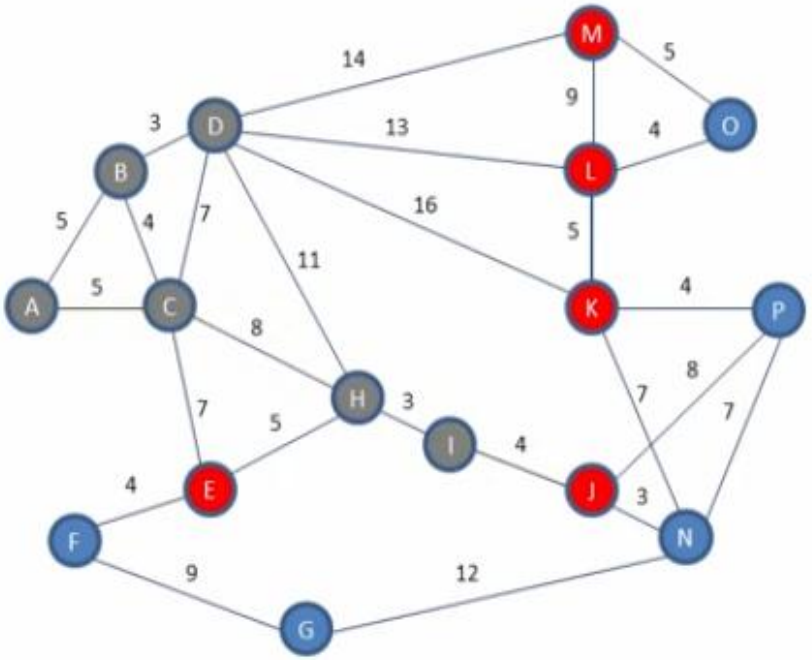


Vertex	Distance from A (g)	Heuristic distance (h)	f = g + h	Previous vertex
A	0	16	16	
B	5	17	22	A
C	5	13	18	A
D	8	16	24	B
E	12	16	28	C
F		20		
G		17		
H	13	11	24	C
I	16	10	26	H
J	20	8	28	I
K	24	4	28	D
L	21	7	28	D
M	22	10	32	D
N		7		
O		5		
P		0		

مثال (۸)

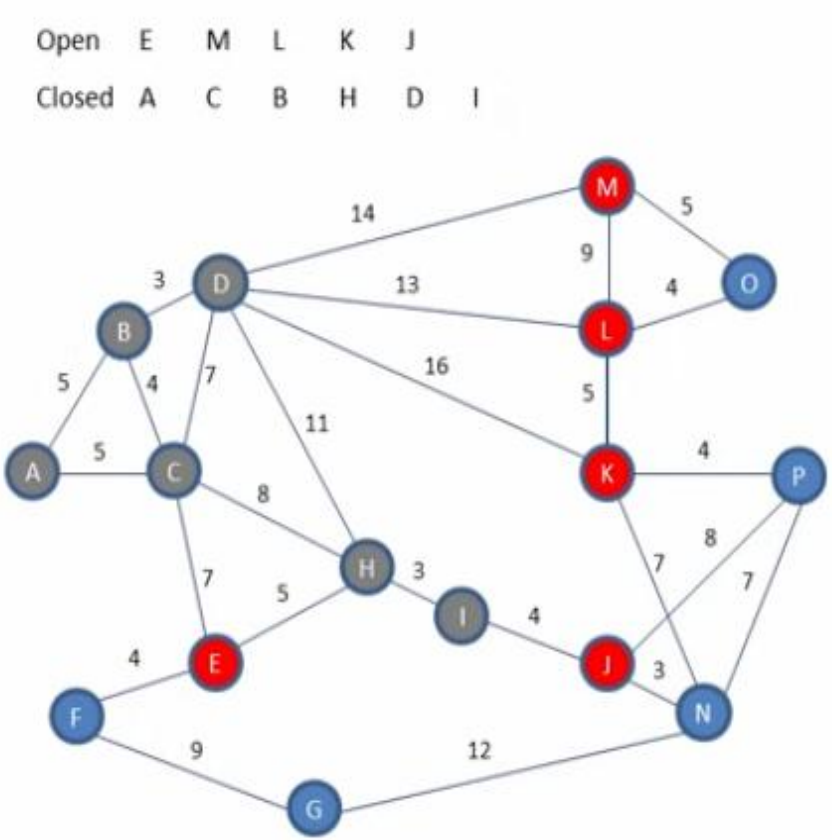
Open E M L K J

Closed A C B H D I



Vertex	Distance from A (g)	Heuristic distance (h)	f = g + h	Previous vertex
A	0	16	16	
B	5	17	22	A
C	5	13	18	A
D	8	16	24	B
E	12	16	28	C
F		20		
G		17		
H	13	11	24	C
I	16	10	26	H
J	20	8	28	I
K	24	4	28	D
L	21	7	28	D
M	22	10	32	D
N		7		
O		5		
P		0		

مثال (۸)

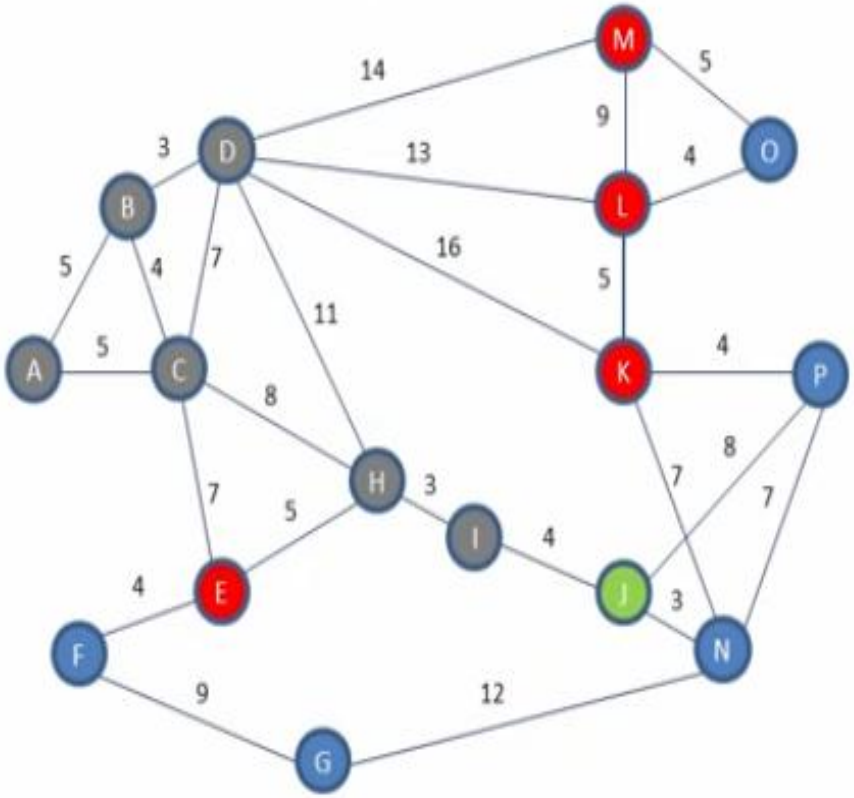


Vertex	Distance from A (g)	Heuristic distance (h)	f = g + h	Previous vertex
A	0	16	16	
B	5	17	22	A
C	5	13	18	A
D	8	16	24	B
E	12	16	28	C
F		20		
G		17		
H	13	11	24	C
I	16	10	26	H
J	20	8	28	I
K	24	4	28	D
L	21	7	28	D
M	22	10	32	D
N		7		
O		5		
P		0		

مثال (۸)

Open E M L K J

Closed A C B H D I

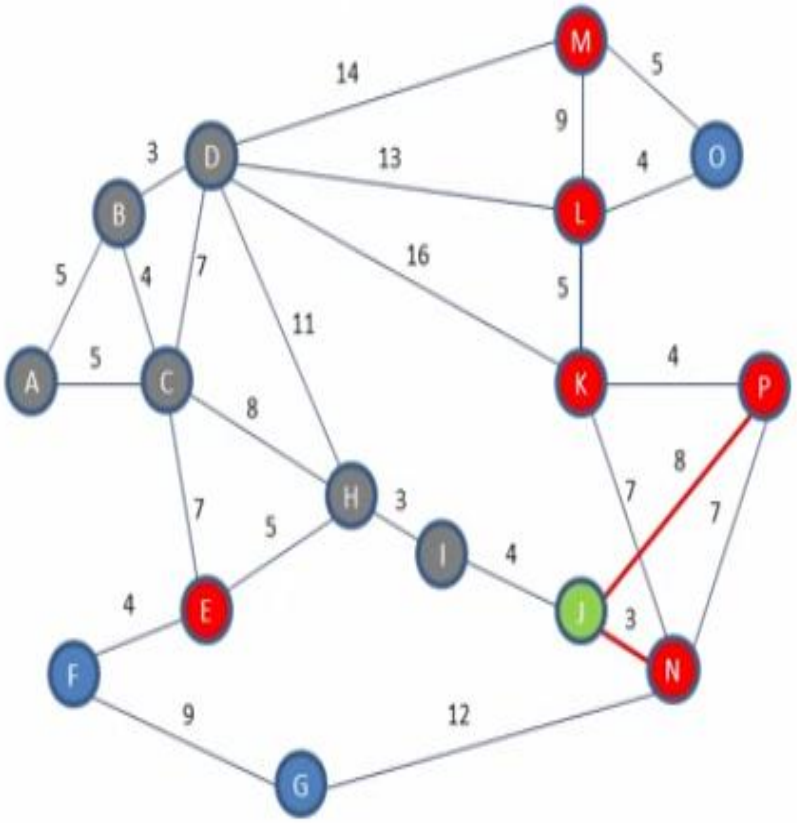


Vertex	Distance from A (g)	Heuristic distance (h)	f = g + h	Previous vertex
A	0	16	16	
B	5	17	22	A
C	5	13	18	A
D	8	16	24	B
E	12	16	28	C
F		20		
G		17		
H	13	11	24	C
I	16	10	26	H
J	20	8	28	I
K	24	4	28	D
L	21	7	28	D
M	22	10	32	D
N		7		
O		5		
P		0		

مثال (٨)

Open E M L K J P N

Closed A C B H D I

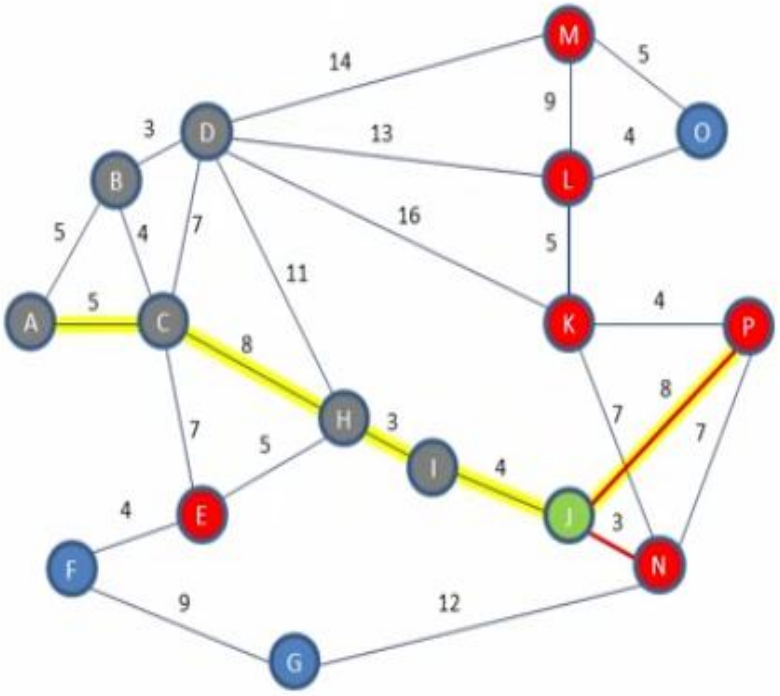


Vertex	Distance from A (g)	Heuristic distance (h)	f = g + h	Previous vertex
A	0	16	16	
B	5	17	22	A
C	5	13	18	A
D	8	16	24	B
E	12	16	28	C
F		20		
G		17		
H	13	11	24	C
I	16	10	26	H
J	20	8	28	I
K	24	4	28	D
L	21	7	28	D
M	22	10	32	D
N	23	7	30	J
O		5		
P	28	0	28	J

مثال (۸)

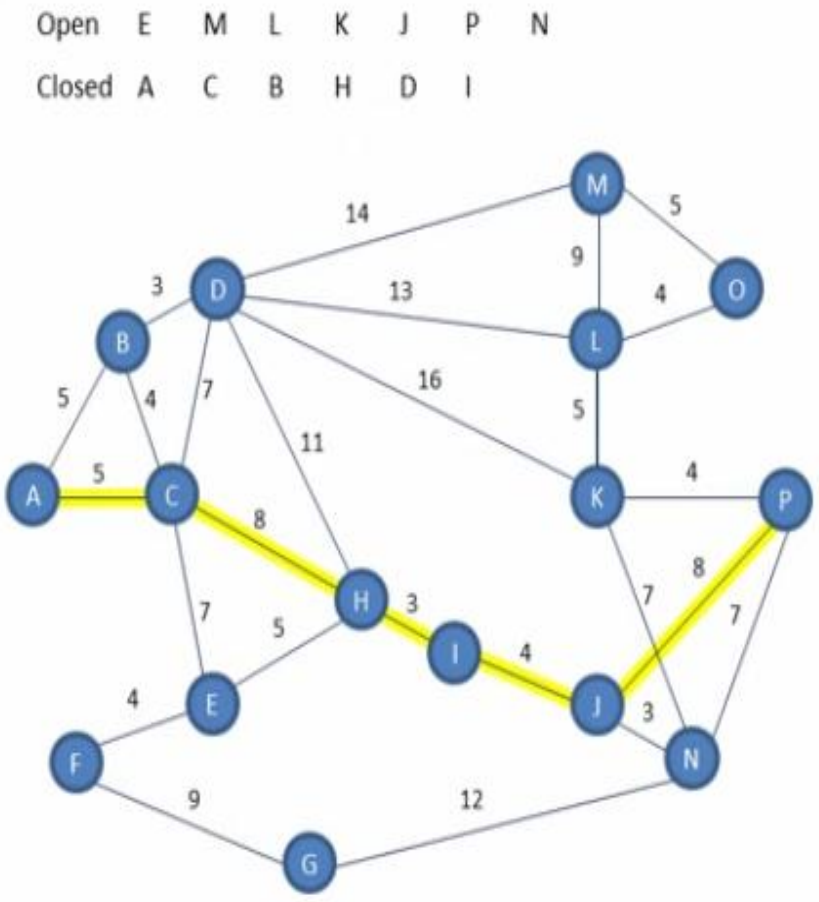
Open E M L K J P N

Closed A C B H D I



Vertex	Distance from A (g)	Heuristic distance (h)	f = g + h	Previous vertex
A	0	16	16	
B	5	17	22	A
C	5	13	18	A
D	8	16	24	B
E	12	16	28	C
F		20		
G		17		
H	13	11	24	C
I	16	10	26	H
J	20	8	28	I
K	24	4	28	D
L	21	7	28	D
M	22	10	32	D
N	23	7	30	J
O		5		
P	28	0	28	J

مثال (٨)

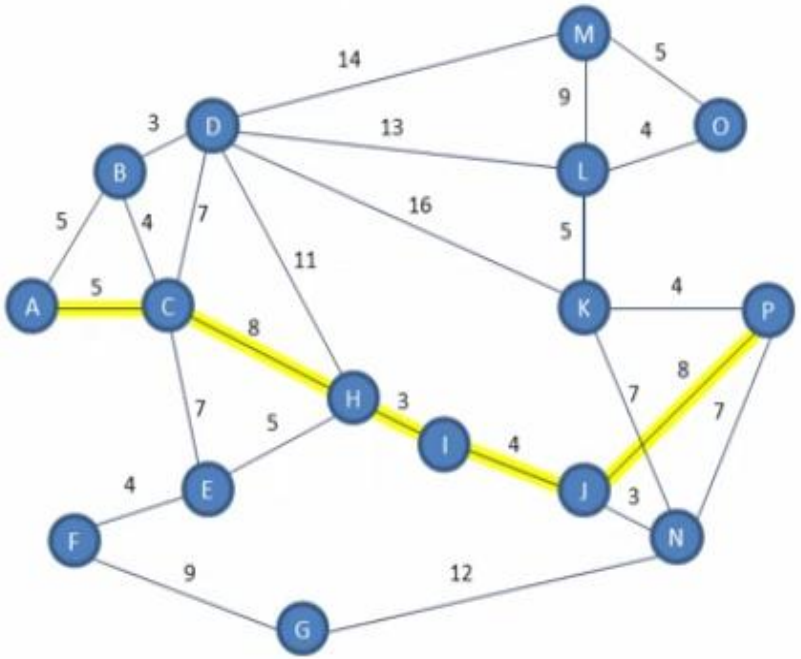


Vertex	Distance from A (g)	Heuristic distance (h)	f = g + h	Previous vertex
A	0	16	16	
B	5	17	22	A
C	5	13	18	A
D	8	16	24	B
E	12	16	28	C
F		20		
G		17		
H	13	11	24	C
I	16	10	26	H
J	20	8	28	I
K	24	4	28	D
L	21	7	28	D
M	22	10	32	D
N	23	7	30	J
O		5		
P	28	0	28	J

مثال (٨)

Open E M L K J P N

Closed A C B H D I

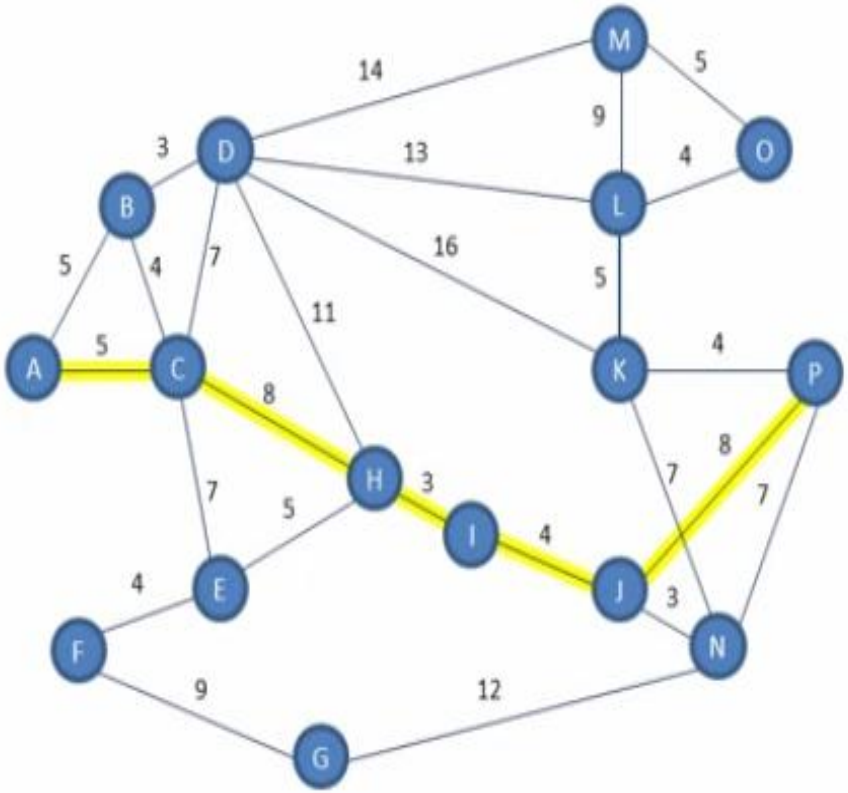


Vertex	Distance from A (g)	Heuristic distance (h)	f = g + h	Previous vertex
A	0	16	16	
B	5	17	22	A
C	5	13	18	A
D	8	16	24	B
E	12	16	28	C
F		20		
G		17		
H	13	11	24	C
I	16	10	26	H
J	20	8	28	I
K	24	4	28	D
L	21	7	28	D
M	22	10	32	D
N	23	7	30	J
O		5		
P	28	0	28	J

مثال (٨)

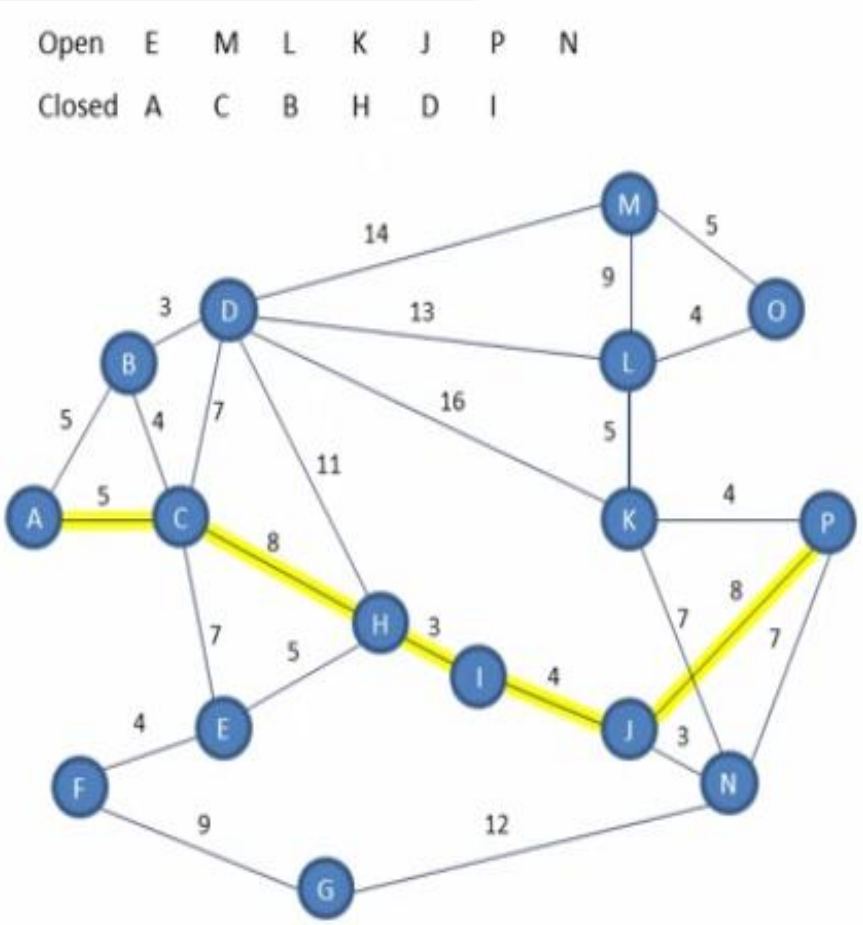
Open E M L K J P N

Closed A C B H D I



Vertex	Distance from A (g)	Heuristic distance (h)	f = g + h	Previous vertex
A	0	16	16	
B	5	17	22	A
C	5	13	18	A
D	8	16	24	B
E	12	16	28	C
F		20		
G		17		
H	13	11	24	C
I	16	10	26	H
J	20	8	28	I
K	24	4	28	D
L	21	7	28	D
M	22	10	32	D
N	23	7	30	J
O		5		
P	28	0	28	J

مثال (۸)

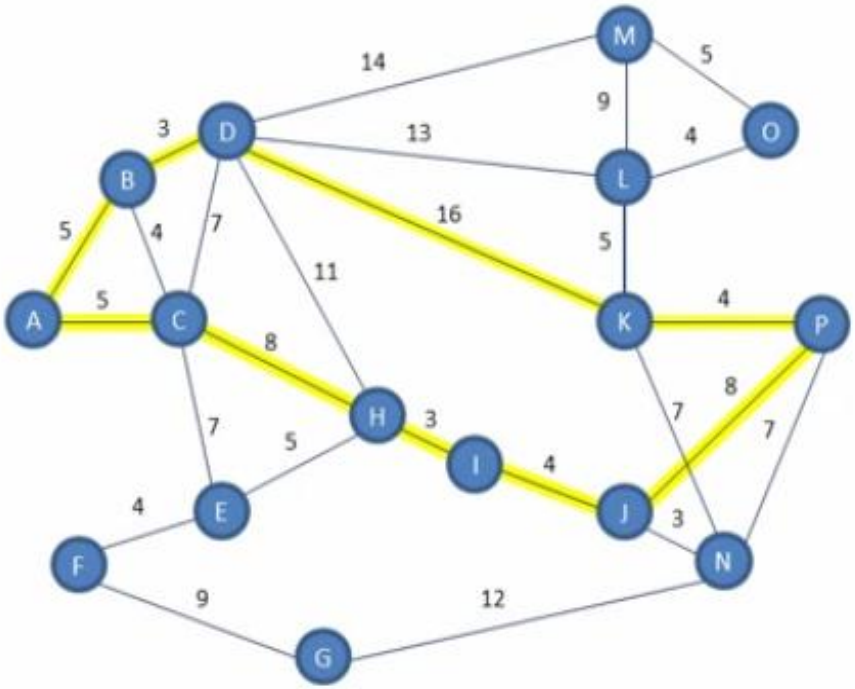


Vertex	Distance from A (g)	Heuristic distance (h)	f = g + h	Previous vertex
A	0	16	16	
B	5	17	22	A
C	5	13	18	A
D	8	16	24	B
E	12	16	28	C
F		20		
G		17		
H	13	11	24	C
I	16	10	26	H
J	20	8	28	I
K	24	4	28	D
L	21	7	28	D
M	22	10	32	D
N	23	7	30	J
O		5		
P	28	0	28	J

مثال (٨)

Open E M L K J P N

Closed A C B H D I



Vertex	Distance from A (g)	Heuristic distance (h)	f = g + h	Previous vertex
A	0	16	16	
B	5	17	22	A
C	5	13	18	A
D	8	16	24	B
E	12	16	28	C
F		20		
G		17		
H	13	11	24	C
I	16	10	26	H
J	20	8	28	I
K	24	4	28	D
L	21	7	28	D
M	22	10	32	D
N	23	7	30	J
O		5		
P	28	0	28	J

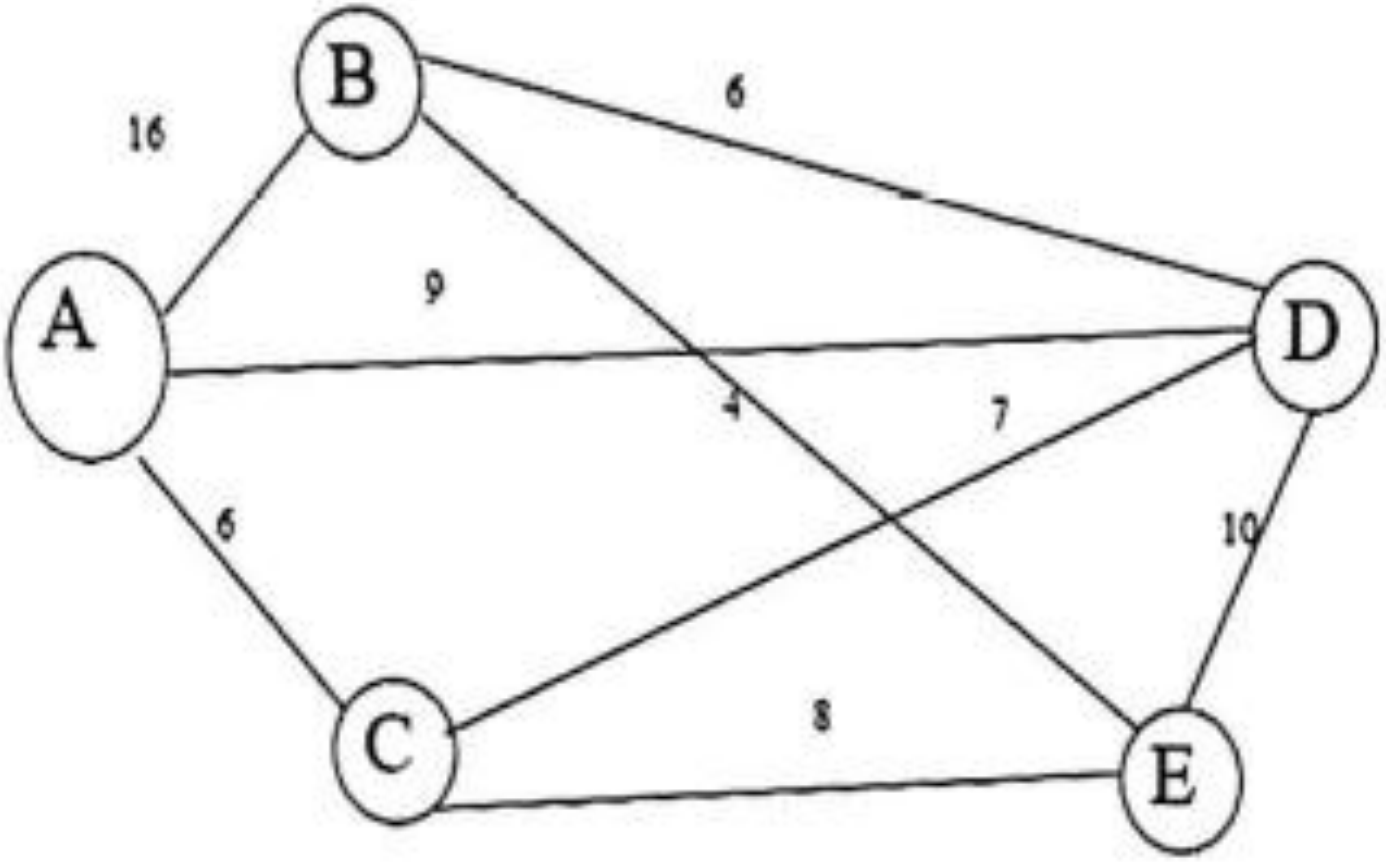
مثال (٨)

```
Initialise open and closed lists
Make the start vertex current
Calculate heuristic distance of start vertex to destination (h)
Calculate f value for start vertex (f = g + h, where g = 0)
WHILE current vertex is not the destination
    FOR each vertex adjacent to current
        IF vertex not in closed list and not in open list THEN
            Add vertex to open list
            Calculate distance from start (g)
            Calculate heuristic distance to destination (h)
            Calculate f value (f = g + h)
            IF new f value < existing f value or there is no existing f value THEN
                Update f value
                Set parent to be the current vertex
            END IF
        END IF
    NEXT adjacent vertex
    Add current vertex to closed list
    Remove vertex with lowest f value from open list and make it current
END WHILE
```

مثال (٩)

- ليكن لدينا البيان التالي و المطلوب تحديد الممر عند الانتقال من A إلى E عند اعتماد خوارزميات البحث التالية:
- العمق أولا
- العرض أولا
- الكلفة المنظمة (الموحدة)

مثال (٩)



مثال (٩)

الحل:

العمق أولاً	A, B, E
العرض أولاً	A, B, D, C, E
الكلفة المنتظمة	A, C, E

2. ماذا سيكون الناتج باستخدام خوارزمية تسلق التلة وخوارزمية A^* علماً بأن لدينا تقدير للمسافة بين كل عقدة والعقدة E كما يلي:

A	B	C	D	E
10	2	8	5	0

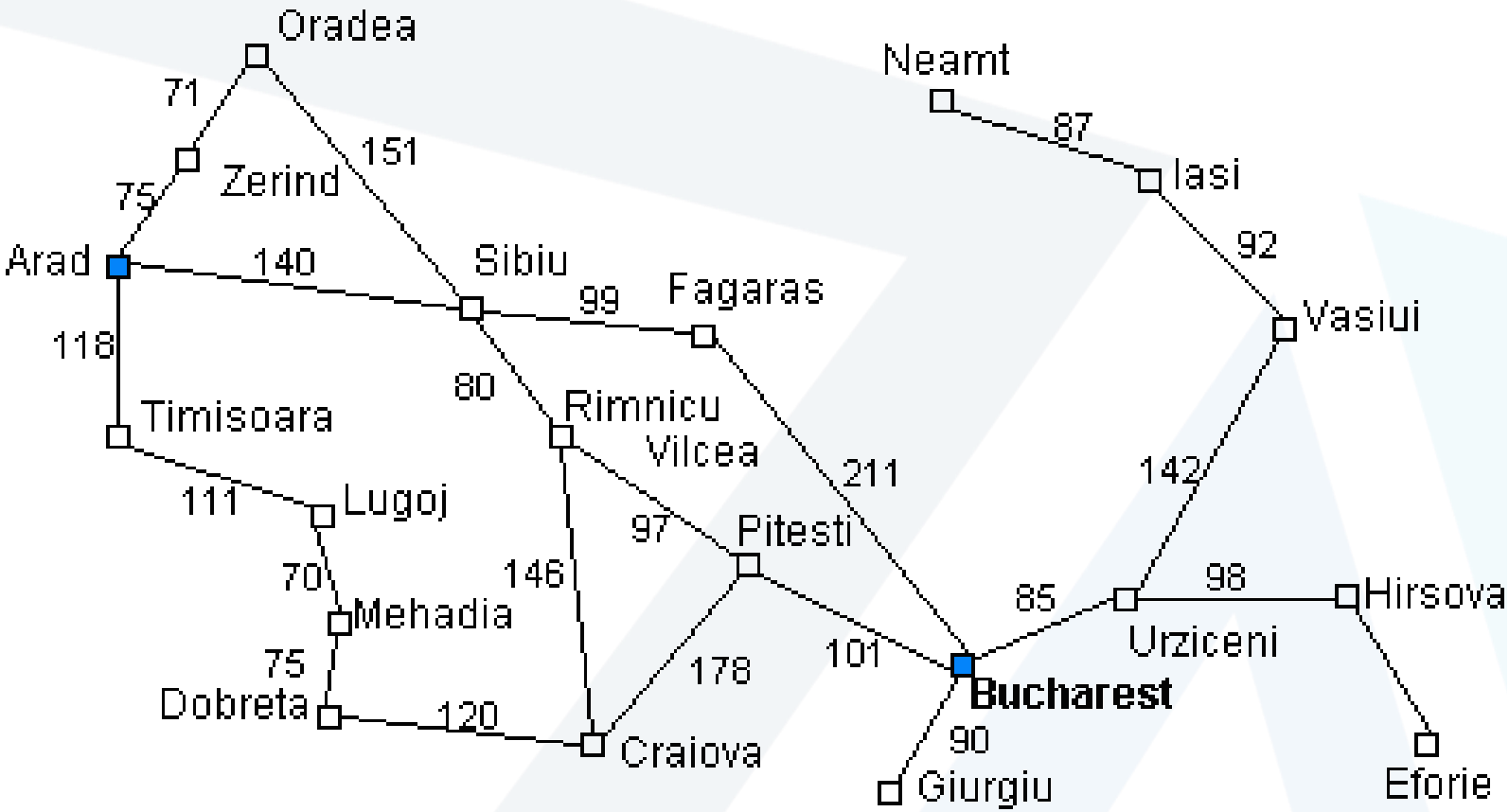
مثال (٩)

الحل:

تسلق التلة	
A^*	A, C, E

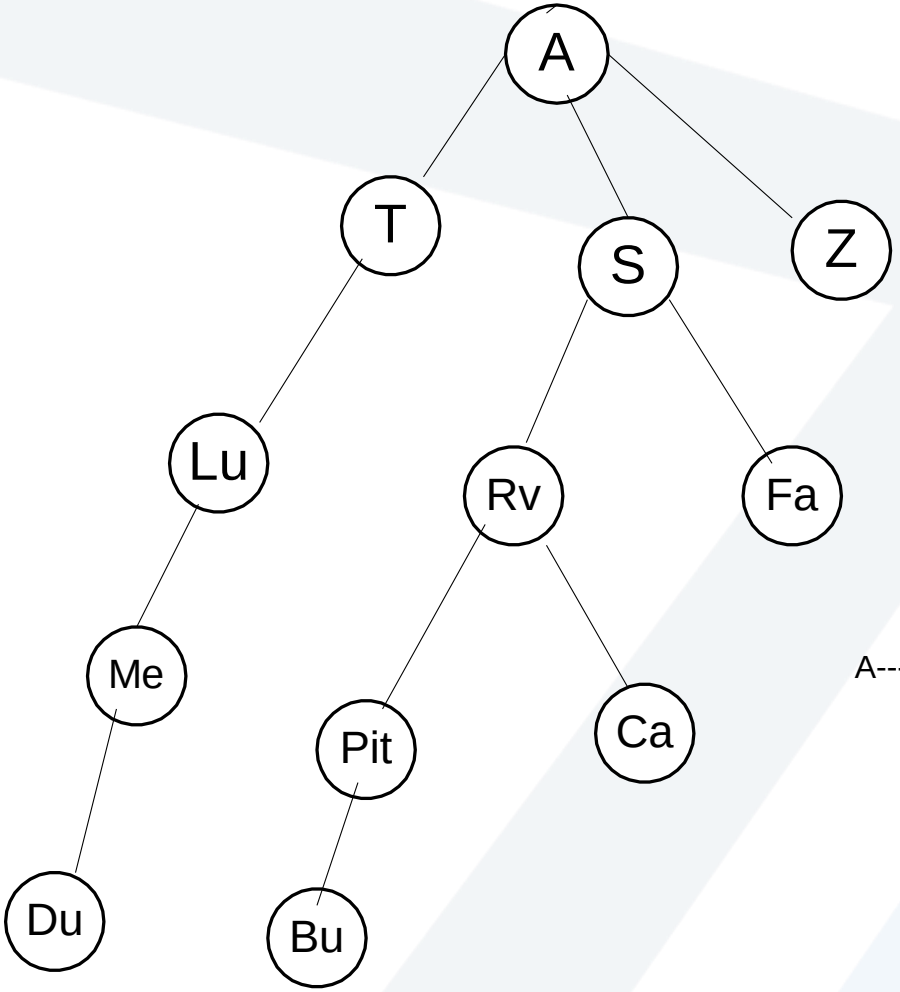
24

Example



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

Example

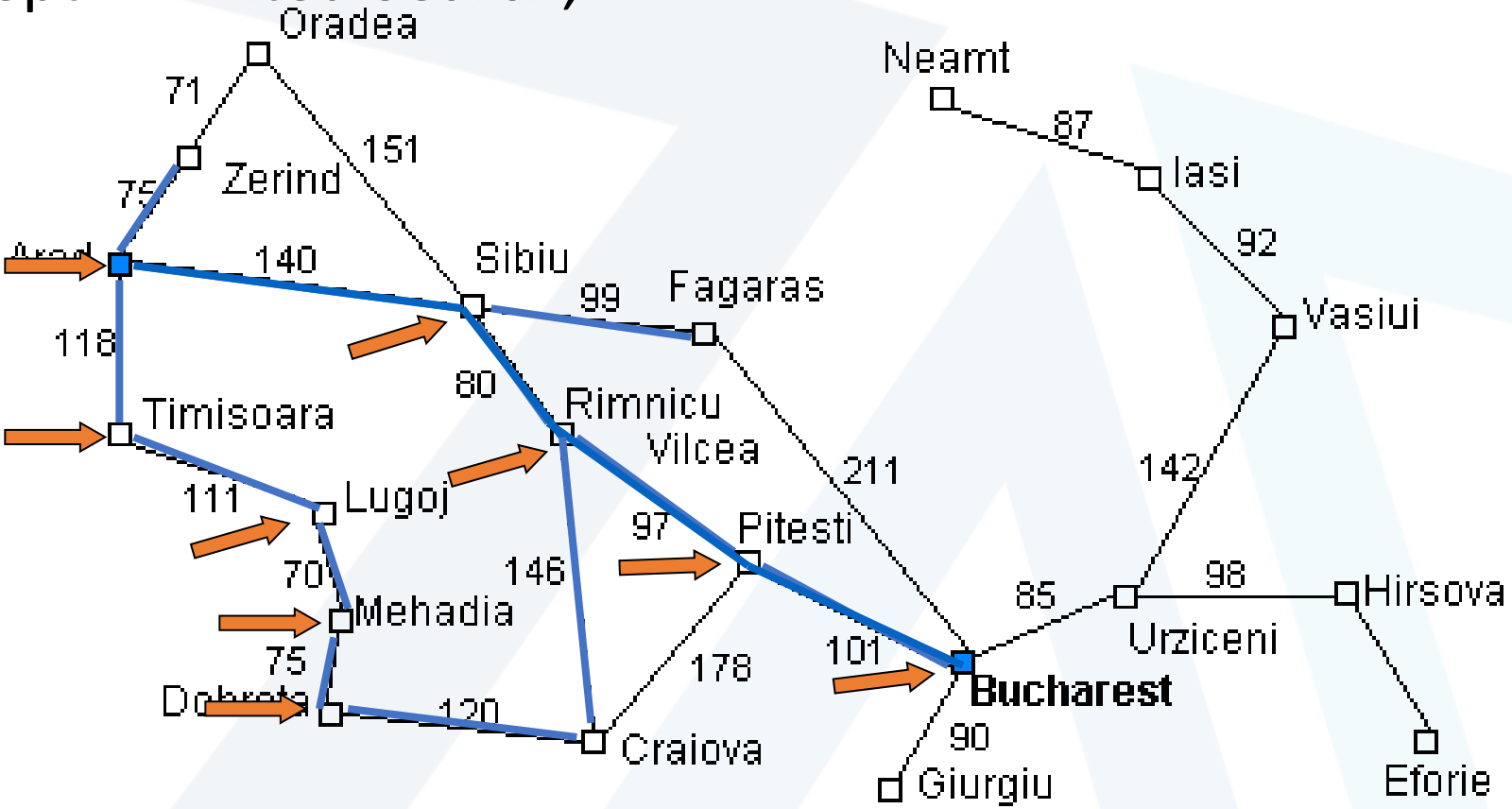


Depth-limited search 4

A--->T--->LU--->ME--->DU--->S--->RV--->PIT--->BU

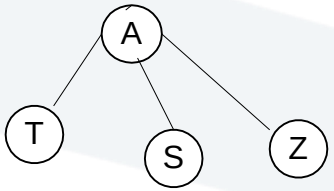
Summary / Example

- Depth-Limited Search, l = 4

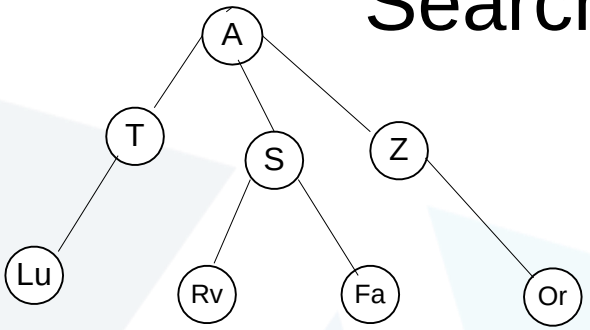


Example

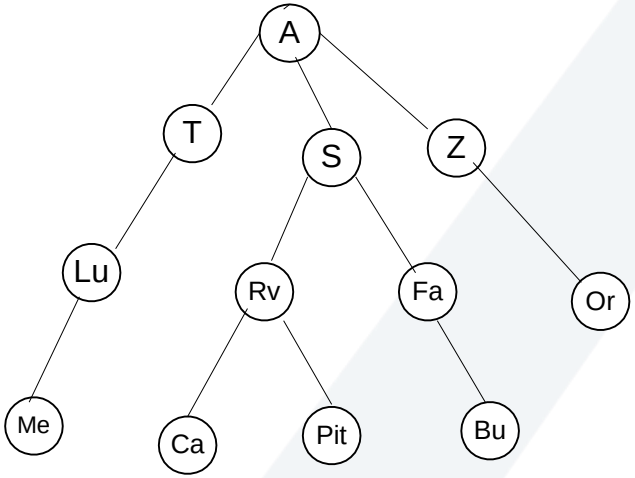
Iterative Deeping Search



A--->T--->S--->Z



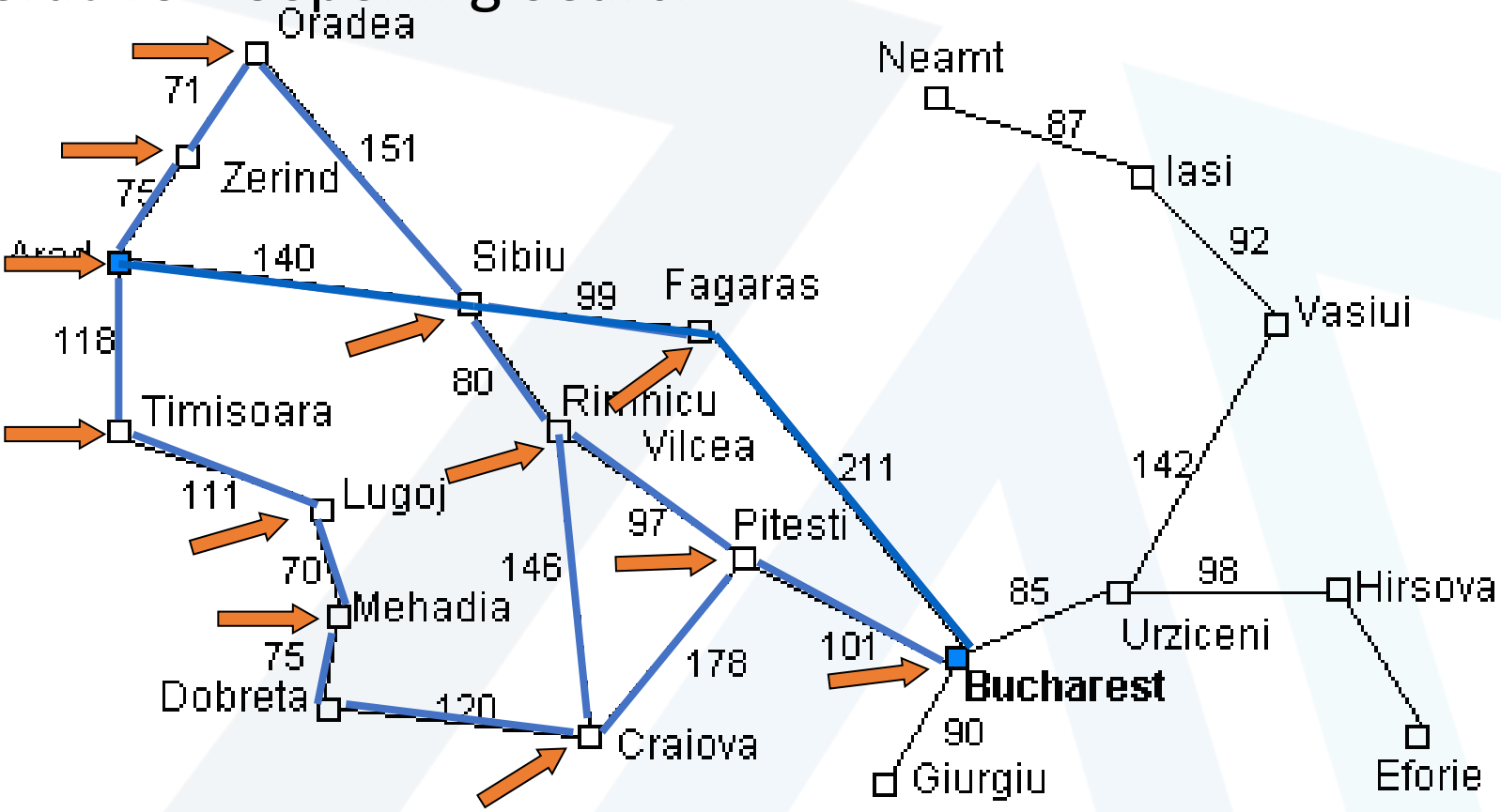
A--->T--->LU--->S--->Rv--->Fa--->Z--->Or



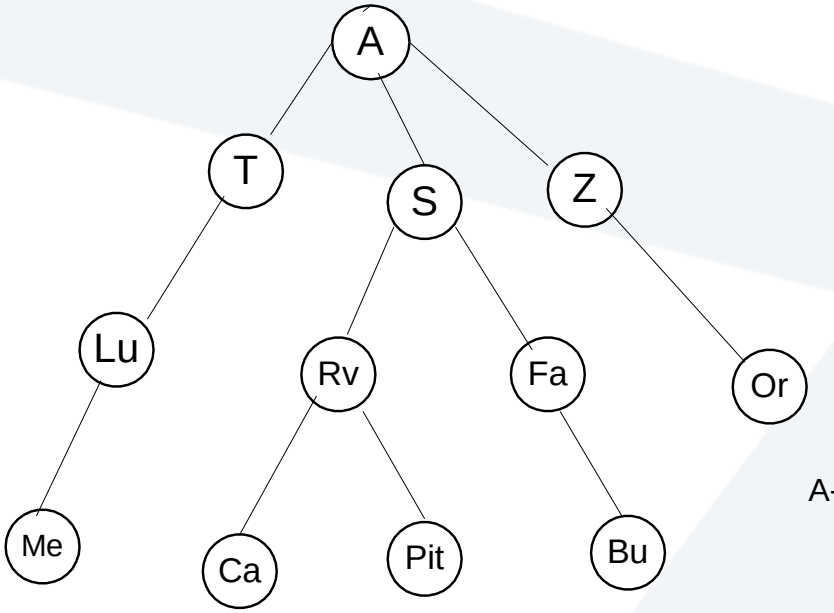
A--->T--->LU--->ME--->S--->RV--->CA--->PIT--->Fa--->
Bu

Summary / Example

- Iterative Deepening Search



Example

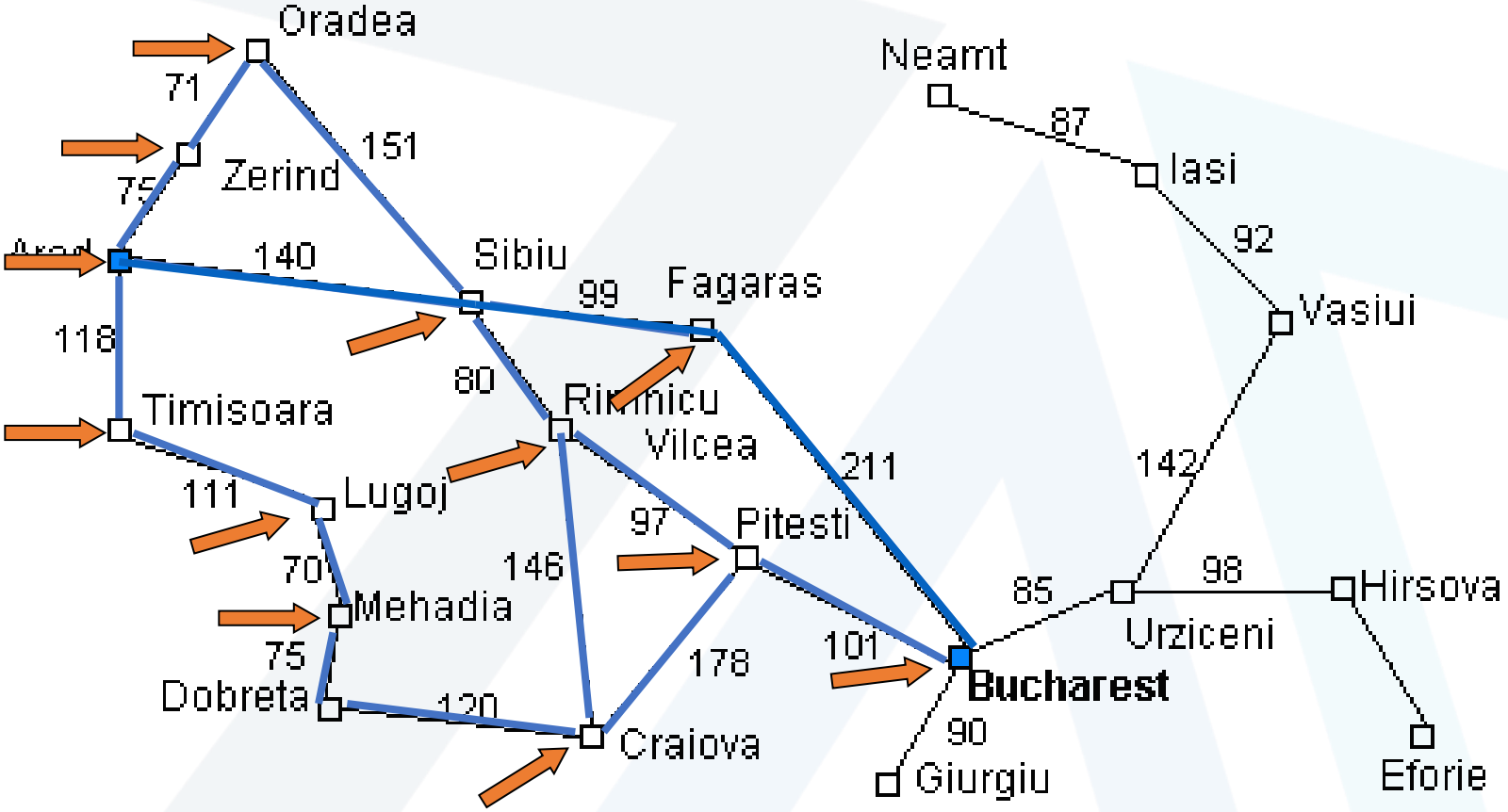


B F S

A--->T--->S--->Z--->Lu--->RV--->Fa--->Or--->Me--->Ca
--->Pit--->Bu

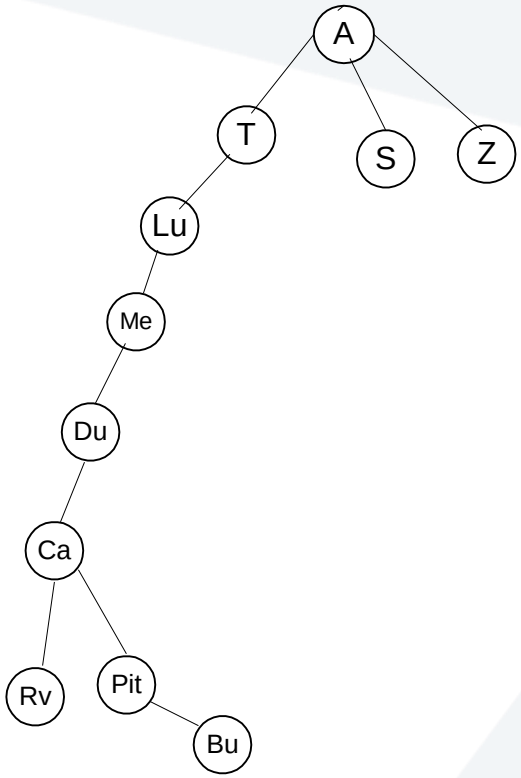
Summary / Example

- Breadth-First Search



Example

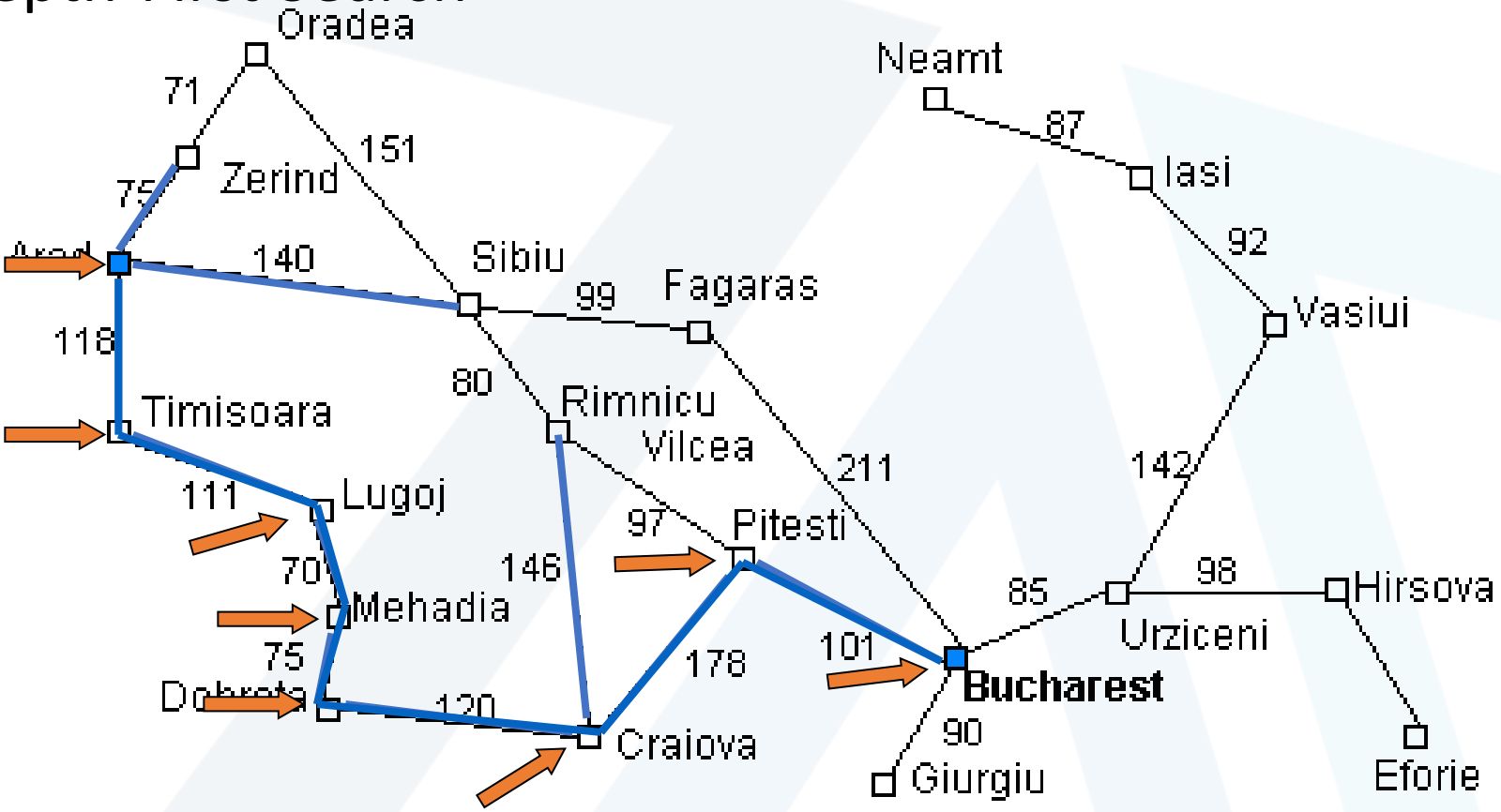
D F S



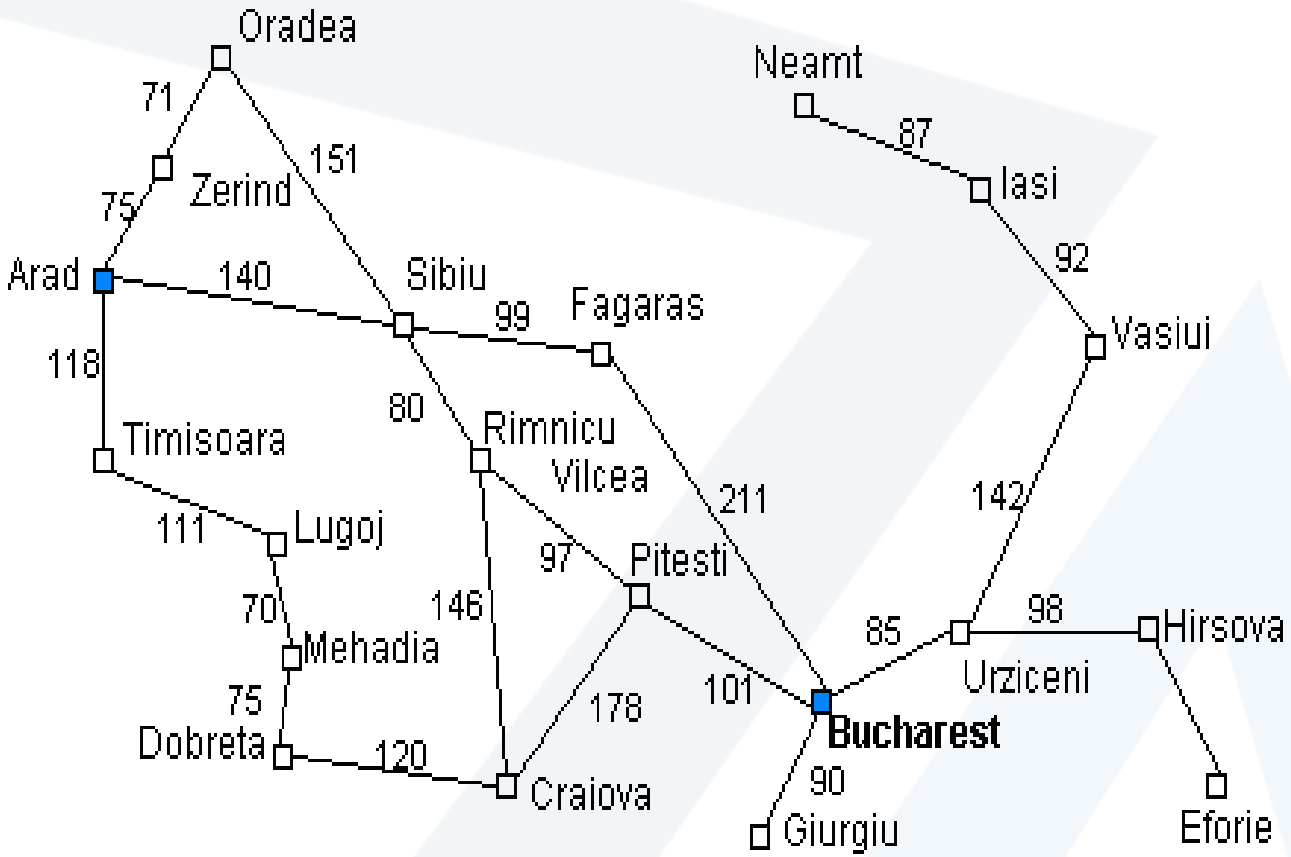
A--->T--->Lu--->Me--->Du--->Ca--->Rv--->Pit--->Bu

Summary / Example

- Depth-First Search



Example



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

UCS

OPEN	CLOSED
$[A_0]$	[ ]
$[Z_{79}, T_{118}, S_{140}]$	$[A_0]$
$[T_{118}, S_{140}, Or_{146}]$	$[A_0, Z_{79}]$
$[S_{140}, Or_{146}, Lu_{229}]$	$[A_0, Z_{79}, T_{118}]$
$[Or_{146}, Rv_{220}, Lu_{229}, Fa_{239}]$	$[A_0, Z_{79}, T_{118}, S_{140}]$
$[Rv_{220}, Lu_{229}, Fa_{239}]$	$[A_0, Z_{79}, T_{118}, S_{140}, Or_{146}]$
$[Fa_{239}, Lu_{229}, Pit_{317}, Ca_{366}]$	$[A_0, Z_{79}, T_{118}, S_{140}, Or_{146}, Rv_{220}]$
$[Lu_{229}, Pit_{317}, Ca_{366}, Bu_{418}]$	$[A_0, Z_{79}, T_{118}, S_{140}, Or_{146}, Rv_{220}, Fa_{239}]$
$[Me_{299}, Pit_{317}, Ca_{366}, Bu_{418}]$	$[A_0, Z_{79}, T_{118}, S_{140}, Or_{146}, Rv_{220}, Fa_{239}, Lu_{229}]$

UCS

OPEN	CLOSED
$[Pit_{317}, Ca_{366}, Do_{373}, Bu_{418}]$	$[A_0, Z_{79}, T_{118}, S_{140}, Or_{146}, Rv_{220}, Fa_{239}, Lu_{229}, Me_{299}]$
$[Ca_{366}, Do_{373}, Bu_{418}]$	$[A_0, Z_{79}, T_{118}, S_{140}, Or_{146}, Rv_{220}, Fa_{239}, Lu_{229}, Me_{299}, Pit_{317}]$
$[Do_{373}, Bu_{418}]$	$[A_0, Z_{79}, T_{118}, S_{140}, Or_{146}, Rv_{220}, Fa_{239}, Lu_{229}, Me_{299}, Pit_{317}, Ca_{366}]$
$[Bu_{418}]$	$[A_0, Z_{79}, T_{118}, S_{140}, Or_{146}, Rv_{220}, Fa_{239}, Lu_{229}, Me_{299}, Pit_{317}, Ca_{366}, Do_{373}]$

Summary / Example

- Uniform-Cost Search