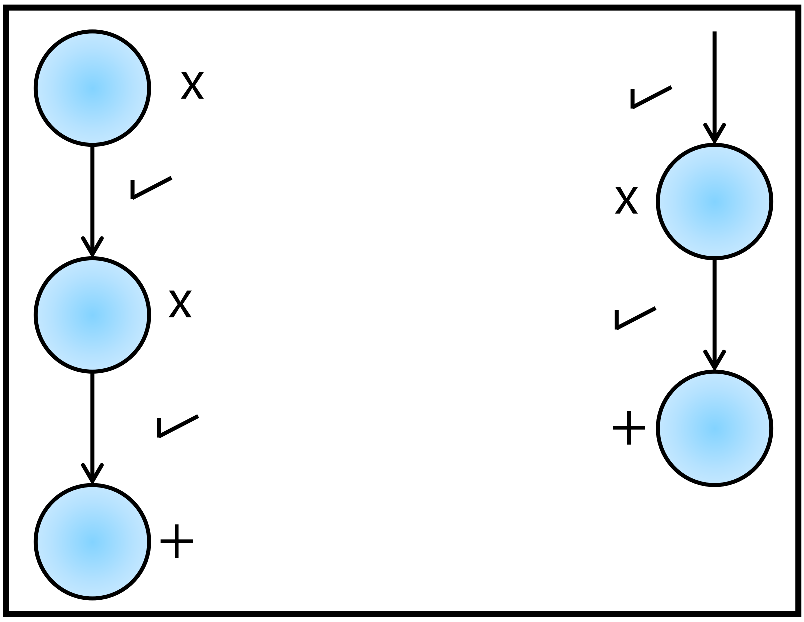
بِسۡمِ ٱللَّهِ ٱلرَّحۡمَٰنِ ٱلرَّحِيمِ

خوارزمية A^*

A^* *Algorithm*

عقدة التكلفة + عقدة المسار (أي يتم حساب كُلفة المسار وكُلفة العقدة وتجمع تكلفة كُلفة المسار وتهمل العقدة السابقة. المسارات مع العقد التي نبحث فيها فقط أي لا تحتسب العقد القديمة فقط المسار. لدينا في هذه الطريقة كُلفة المسار (الطريق).



ملاحظة: عند حساب هذه العقدة فإن المسار السابق والسابق مع العقدة بدون العقدة السابقة.

$$A^* \textit{Algorithm} = \textit{Best First Search} + \textit{Cost Function}$$

A^* خوارزمية البحث هي واحدة من أفضل التقنيات الواسعة الانتشار المستخدمة في إيجاد المسار والانتقالات (العبور أو المرور) عبر البيان *Graph Traversals*.

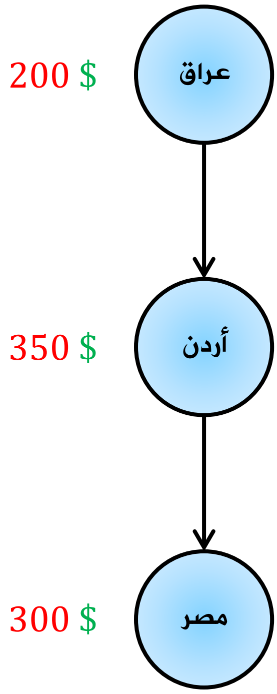
A^* هي *(Admissibility)* أي تجد أقصر طريق للحل

الدالة التي تجد أقصر طريق للهدف تعتبر *(Admissibility)*

الدالة التخمينية أو الحدسية التي تجد أقصر طريق للهدف بأول زيارة للعقدة *(Node)* أي تصل على أقصر طريق (مسار) بزيارة واحدة للعقدة أي لا يحدث حذف داخل العقد *(Open Node)*

ملاحظة: A^* لا يوجد لها تطبيقات

مثلاً: إذا أردنا الذهاب الى مصر نذهب الى الأردن ثم الى مصر سوف نجمع كُلفة الطريق الى الأردن + كُلفة الطريق الى مصر كما موضح بالمخطط أدناه:



تعليمات الخوارزمية (مهمة جداً)

Pseudocode Of A^* Algorithm

Input:

Queue: *Path only containing root.*

Algorithm: *Wile (Queue not empty **&** First Path not reach goal) Do*

- *Remove First Path from Queue.*
- *Create Paths to All Children.*
- *Reject Paths with Loops.*
- *Add Paths and Sort Queue by* $(\boldsymbol{f} = \boldsymbol{Cost} + \boldsymbol{heuristic})$.

If Queue contains Paths: $(\boldsymbol{P}, \boldsymbol{Q})$. *And* $\boldsymbol{P}$ *ends in node* $\boldsymbol{N_i}$ & $\boldsymbol{Q}$ *contains node* $\boldsymbol{N_i}$ *and* $\boldsymbol{Cost\_P} \geq \boldsymbol{Cost\_Q}$

Then remove ***P_IF*** *goal reached* ***THEN*** *success* ***ELSE*** *failure.*

$Queue$: المسار فقط يحتوي على جذر.

1- إزالة المسار الأول من $Queue$.

2- إنشاء مسارات لكل من الأبناء.

3- رفض المسارات مع الحلقات.

4- إضافة مسارات وفرزها بواسطة:

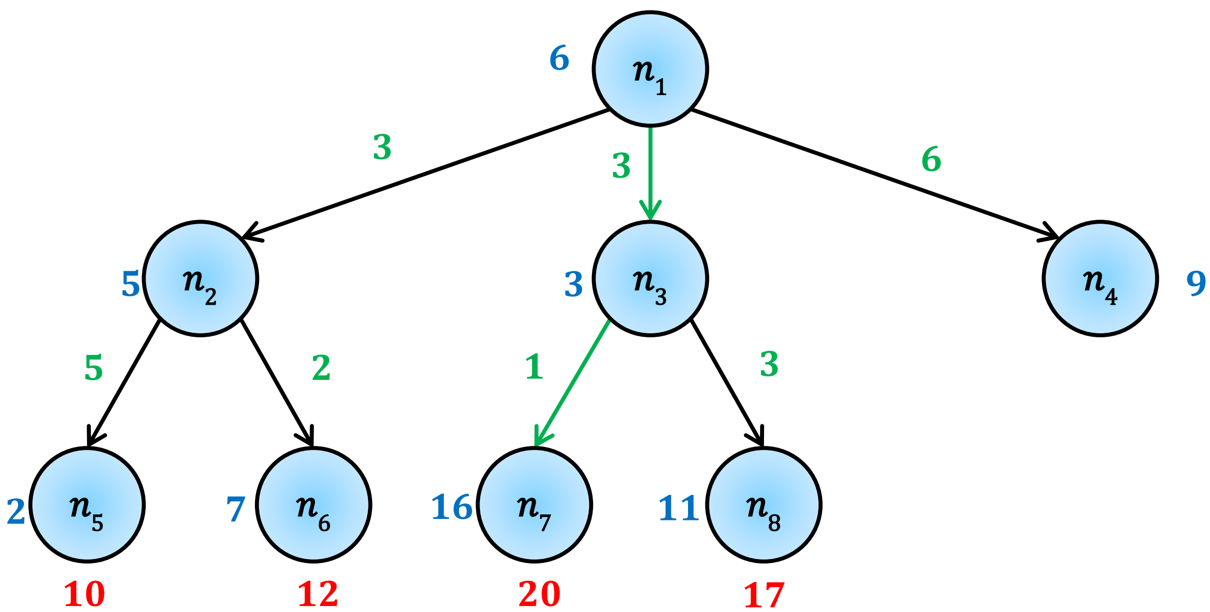
$$(f = Cost + heuristic)$$

5- إذا كانت $Queue$ تحتوي على مسارات (P, Q) و P نهاية في العقدة Q & N_i تحتوي على عقدة N_i وتكلفة $Cost\_P \geq Cost\_Q$.

نقوم بإزالة P

إذا وصلنا للهدف نجاح وإلا فشلنا.

مثال:



الحل: في الصفحة القادمة.

Iteration	***Open***	***Path***
1	$[n_1(6)]$ or $n_1(6)$	[ ]
2	$n_3(6), n_2(8), n_4(15)$	$n_1(6)$
3	$n_2(8), n_4(15),$ $n_8(17), n_7(20)$	$n_1(6), n_3(6)$
4	$n_5(10), n_6(12), n_4(15),$ $n_8(17), n_7(20)$	$n_1(6), n_3(6),$ $n_2(8)$
5	$n_6(12), n_4(15),$ $n_8(17), n_7(20)$	$n_1(6), n_3(6),$ $n_2(8), n_5(10)$
6	$n_4(15), n_8(17), n_7(20)$	$n_1(6), n_3(6), n_2(8),$ $n_5(10), n_6(12)$
7	$n_8(17), n_7(20)$	$n_1(6), n_3(6), n_2(8),$ $n_5(10), n_6(12), n_4(15)$
8	$n_7(20)$	$n_1(6), n_3(6), n_2(8),$ $n_5(10), n_6(12), n_4(15), n_8(17)$
9	$\boldsymbol{n_7}$ ***is the Goal***	$n_1(6), n_3(6), n_2(8), n_5(10),$ $n_6(12), n_4(15), n_8(17), n_7(20)$

$State\ Space: All\ Node$

$Search\ Space: \{n_1, n_3, n_2, n_5, n_6, n_9, n_4, n_8, n_7\}$

$Solution\ Path: \boldsymbol{n_1 \rightarrow n_3 \rightarrow n_7}$

بِسۡمِ ٱللَّهِ ٱلرَّحۡمَٰنِ ٱلرَّحِيمِ

خوارزمية البحث الأفضل اولاً (الأسبقية للأفضل)

$Best - First\ Search\ Algorithm$

تعتبر طريقة $(Best - First\ Search\ )$ تصحيح (تصحيحية) لطريقة $(Hill\ Climbing)$ حيث عندما تفشل (حدوث فشل) في ايجاد الهدف (لم نجد الهدف) في طريق $(Hill\ Climbing)$ نستخدم طريقة $(B.F.S.)$ وذلك لأنه في طريقة الـ $(Hill\ Climbing)$ تأخذ أقل قيمة وتبحث فيها من دون الرجوع الى بقية العقد. أما في طريقة $(B.F.S.)$ حيث نعود (نرجع) للعقد البقية (السابقة) ونجد مسار او مسلك آخر.

- الأبناء يتسابقون مع الآباء.

تبحث في الكُلفة الأقل ولا تحذف (تهمل) الكُلف الكبيرة لبقية العقد وإنما تبحث فيها الى أن نصل الى الهدف.

نأخذ الكلفتين المتشابهة ونوحد اما أقل كُلفة (Min) او اعلى كُلفة (Max) سوياً دون ان نهمل اي عقدة منها.

- يتم ترتيب العقد من أقل كُلفة الى أعلى كُلفة.

تعتمد طريقة الـ $(B.F.S.)$ على $(backtracking)$ الرجوع الخلفي حيث أن:

$$Best\ First\ Search = Hill\ Climbing\ Search + Backtracking$$

- يمكن الرجوع لبقية الاحتمالات:

في حالة عدم الوصول الى الهدف تفشل طريقة $(Hill\ )$ وذلك لأنها تأخذ الطريق أو التفرع وتبحث به فقط.

أما في طريقة $(B.F.S.)$ سوف نأخذ تفرع واذا أخطأ ترجع وتبحث في تفرع أخر.

- يتم الوصول الى الهدف بأقل كُلفة أي الخطوات حسب الكلفة.

الفرق بين خوارزمية التسلق الشاهق *Hill Climbing* و خوارزمية البحث الأفضل أولاً *Best – First Search Algorithm* (مهمة جداً)

التسلق الشاهق *Hill Climbing*	خوارزمية البحث الأفضل *Best – First Search Algorithm*

1

تبحث في الكلفة الأقل وتهمل الكلف الكبيرة وليس من الضروري أن تصل الى الهدف	تبحث في الكلفة الأقل (العقدة الأقل كُلفة) ثم تنتقل الى الأخرى دون أن تهمل الكلف الكبيرة (لا تحذف كُلفة العقد أو العقدة الأكبر) تبقيها الى أن تصل الى الهدف

2

غير مضمونة الوصول الى الحل (فشل الحل أو فشل الوصول الى الهدف)	مضمونة الوصول الى الحل

3

أقل كفاءة من طريقة الـ *Best – First Search*	أفضل من طريقة الـ *Hill Climbing*

4

طريقة البحث فيها تكون بدلالة يعني لا تبحث بجميع العقد. فقط الأقل كُلفة ولا تحتوي على ***backtracking***	طريقة البحث فيها تعتمد على ***backtracking*** (الرجوع والبحث في بقية العقد لإيجاد الهدف) يمكن الرجوع.

5

تختصر الوقت والخزن والجهد (لأنها تبحث في كل أو جميع العقد ولكن تبحث بالعقد الأقل كُلفة	تحتاج الى وقت وجهد وخزن أكبر من *Hill Climbing*

6

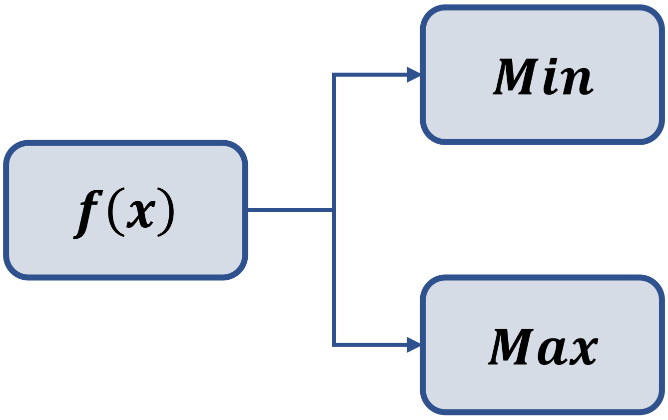
فشل الحل في حالة عدم الوصول الى الهدف	هي تصحيح لطريقة الـ ***Hill Climbing Search*** + ***Backtracking*** تصل الى الهدف لأنها تبحث في جميع العقد

7

لا يمكن الرجوع لإيجاد الهدف (لأنه سوف نختار التكلفة الأقل)	يمكن الرجوع لإيجاد الهدف

تنفيذ خوارزمية التسلق الشاهق $Hill\ Climbing$ على مسألة $8 - Puzzle$ كــــدالة كُلفة $(Cost\ Function)$.

هذه الطريقة تعتمد على دالة الكُلفة (أي مقدار الكُلفة والتكلفة الفعلية) أي يجب أن نأخذ دالة تحسب التكلفة (الكُلفة) وهي على نوعيتين:



إما تأخذ أقل قيمة

أو تأخذ أكبر قيمة

حيث أن:

$f(x) = L + G$ **قيمة تكلفة العقدة**

(L) **: رقم المستوى** $(Level)$.

(G) **: عدد المربعات (التي يتم مقارنتها مع الهدف** $(Goal)$).

$Min\ f(x) = (L)$ **رقم المستوى + عدد المربعات المختلفة مع الهدف**

- **عند الحل بالـ** (Min) **لا يتم حساب المربع الفارغ مع الاختلاف نختار أقل كُلفة موجودة ونفرعها (يتم التفريع).**

$Max\ f(x) = (L)$ **رقم المستوى + عدد المربعات المتشابهة مع الهدف**

- **عند الحل بالـ** (Max) **لا يتم حساب المربع الفارغ مع الهدف. عدم حساب المربع الفارغ اي ان الفراغ في عقدة التفرع يترك في حسابه عند المقارنة مه الهدف**
- **لا يمكن الرجوع لبقية الاحتمالات أي يتم التوقف في** $Hill\ Climbing$ **ولا يمكن الرجوع.**

ملاحظات مهمة

1-

- اذا بدأنا بالـ (Min) فـنستمر بحسابها (لا نتوقف) لـجميع المستويات ونأخذ أقل كُلفة وتهمل باقي الكُلف.
- اذا بدأنا بالـ (Max) فـنستمر بحسابها (لا نتوقف) لـجميع المستويات ونأخذ أعلى كُلفة وتهمل باقي الكُلف.

2- لا تحتاج الى ترتيب العقد لأنها تأخذ تكلفة ولحساب الـ $(Cost)$ نحسب عدد المربعات في التفرع مع الهدف.

3- إذا تكررت أكثر من كُلفة (أي اذا كانت الكلفتان متساويتان فيتم اختيار الكلفة الموجودة الى اقصى اليسار.

تعليمات الخوارزمية (مهمة)

Pseudocode of Best – First Search Algorithm (B.F.S.)

Step 1: *form a one–element list consisting of the root*

Step 2: *Remove the first element from the list. Expand the first element. If one of the descendants of first element is a Goal node, then stop; otherwise, add the descendants into the list.*

Step 3: *Sort the entire list by the values of some estimation function.*

Step 4: *if the list is empty, then failure, otherwise, go to step* **2**.

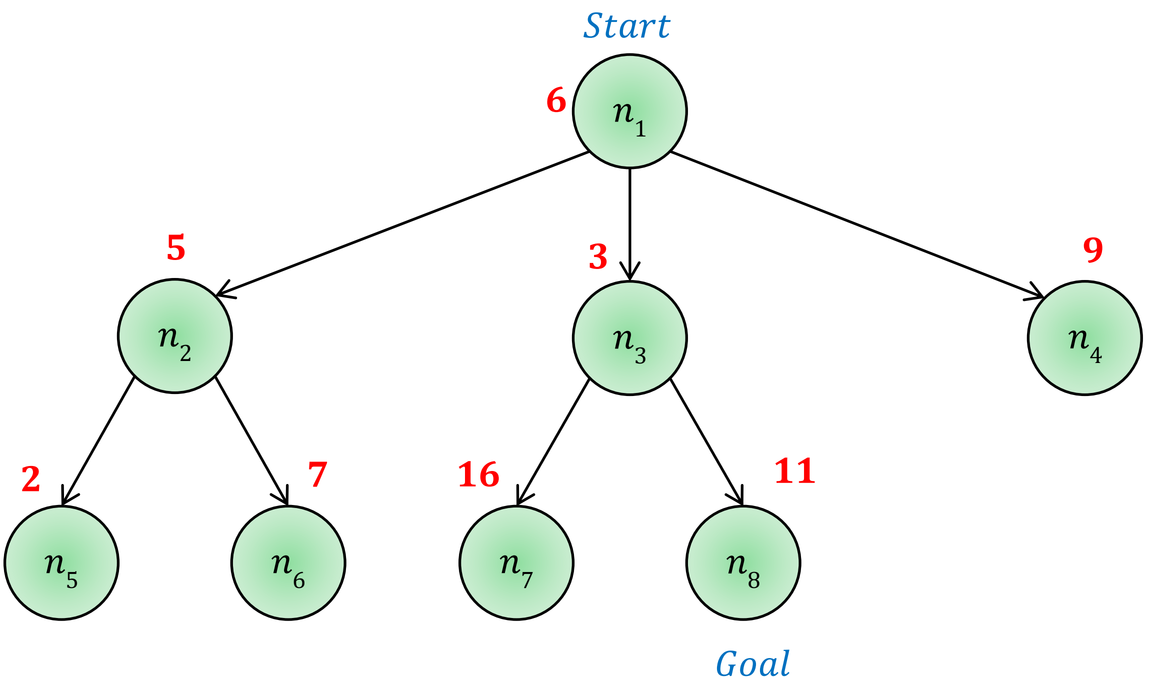
1- تشكيل من قائمة عنصر واحد تتكون من عقدة الجذر.

2- إزالة العنصر الأول من القائمة ونوسع العنصر الأول اذا كان أحد احفاد العنصر الأول هو عقدة الهدف ثم نتوقف ماعدا ذلك نضيف (يتم إضافة) الاحفاد الى القائمة.

3- نرتب المدخلات في القائمة (فرز القائمة) بواسطة قيم بعض دالة التقدير.

4- إذا كانت القائمة فارغة (خالية) إذاً فشل. ماعدا ذلك و إلا اذهب الى الخطوة رقم 2.

مثال:



Iteration	C_S	***Open***	***Path***	***Optimal***
1	$n_1(6)$	$n_1(6)$	[ ]	[ ]
2	$n_3(9)$	$n_3(9), n_2(11), n_4(15)$	$n_1(6), n_3(9)$	$n_2 = 11, n_3 = 9, \boldsymbol{x = n_3 = 9}$
3	$n_8(20)$	$n_8(20), n_7(25)$	$n_8(20), n_7(25)$	$n_7 = 25, n_8 = 20, \boldsymbol{x = n_8 = 20}$
4	$n_8(20)$	***Goal***	$n_8(20), n_7(25)$	$\boldsymbol{x = n_8 = 20}$