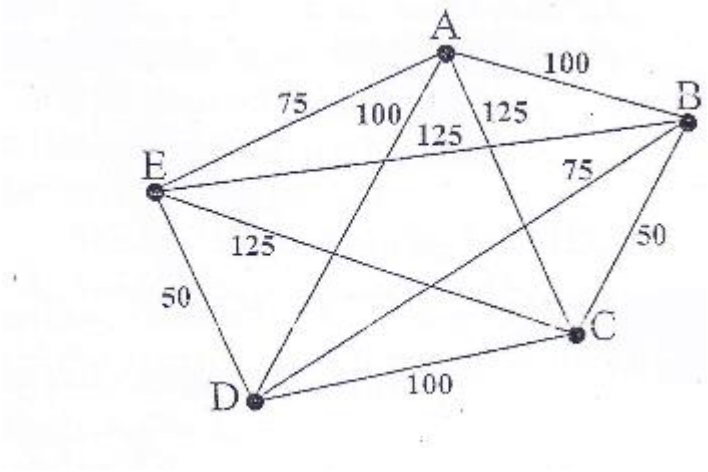
Example : Traveling Salesperson Problem(مسألة البائع المتجول) : suppose a salesperson has five cities to visit and then must return home. The goal of the problem is to find the shortest path for the salesperson to travel, visiting each city, and then returning to the starting city.

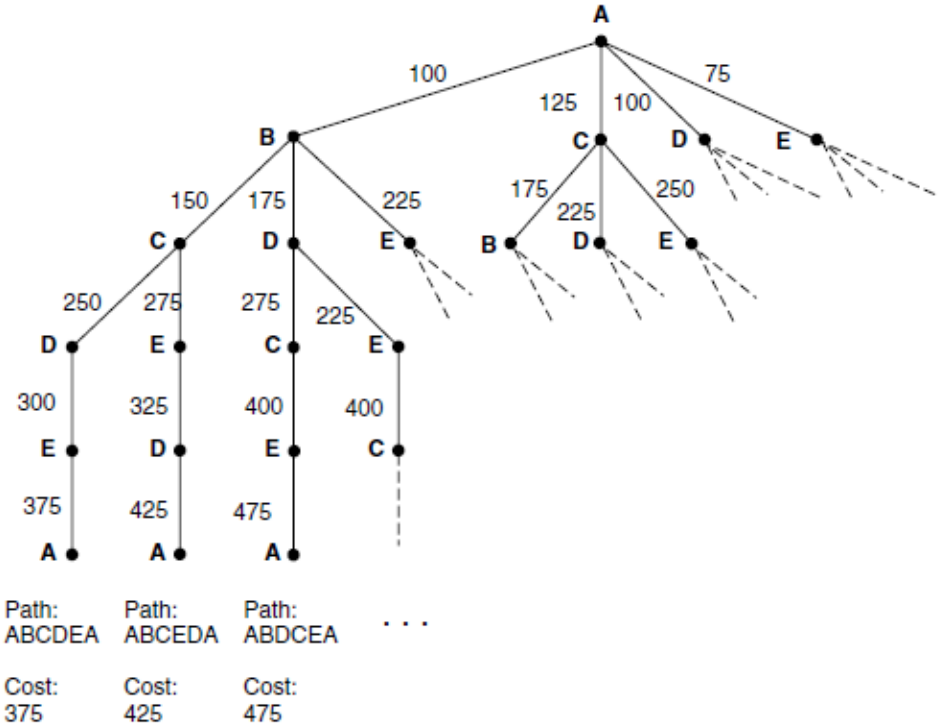
The nodes of the graph below represent cities, and each are is labeled with a weight indicating the cost of traveling that arc.

البائع المتجول يبدأ بالمدينة (A) ويتجول في جميع المدن ثم يعود الى المدينة (A) من دون تكرار للمدن في زيارته. الهدف من هذه المسألة هو ايجاد اقصر مسار للبائع المتجول خلال زيارته للمدن والرجوع الى نقطة البداية (الانطلاق).



وعدد الاحتمالات هي : !(N-1) أي (24) احتمال

N number of cities



Search of the traveling salesperson problem. Each arc is marked with the total weight of all paths from the start node (A) to its endpoint.

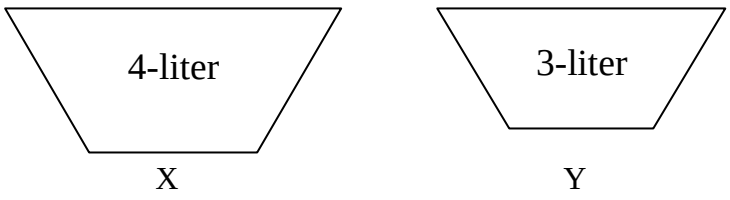
The path [A,D,C,B,E,A], with associated cost of 450 miles, is an example of a possible circuit. The goal description requires a complete circuit with minimum cost.

The Path is : A—B — C — D — E — A

Where the minimum cost(goal) = 375

Example: Water Jug Problem: Suppose you have two jugs(3 liters) and (4 liters), the goal is to fill one of them in (2 liters) without using any extra equipment. Translate the solution as tree.

افرض أن لدينا اناءين سعة احدهما (3 لتر) والآخر (4 لتر)، المطلوب استخدم الاناءين لغرض الحصول على سعة (2 لتر) وبدون استخدام ادوات اضافية، حول او ترجم الحل على شكل شجرة.



1- نملأ الاناء سعة (3 لتر) بشكل كامل.

2- نقوم بنقل محتويات الاناء (3 لتر) الى الاناء (4 لتر) فنحصل على (3 لتر) بداخل الاناء (4 لتر).

3- نقوم مرة ثانية بملئ الاناء (3 لتر).

4- نملأ الاناء (4 لتر) بالكامل.

5- النتيجة الحصول على (2 لتر) في الاناء (3 لتر).

State space :

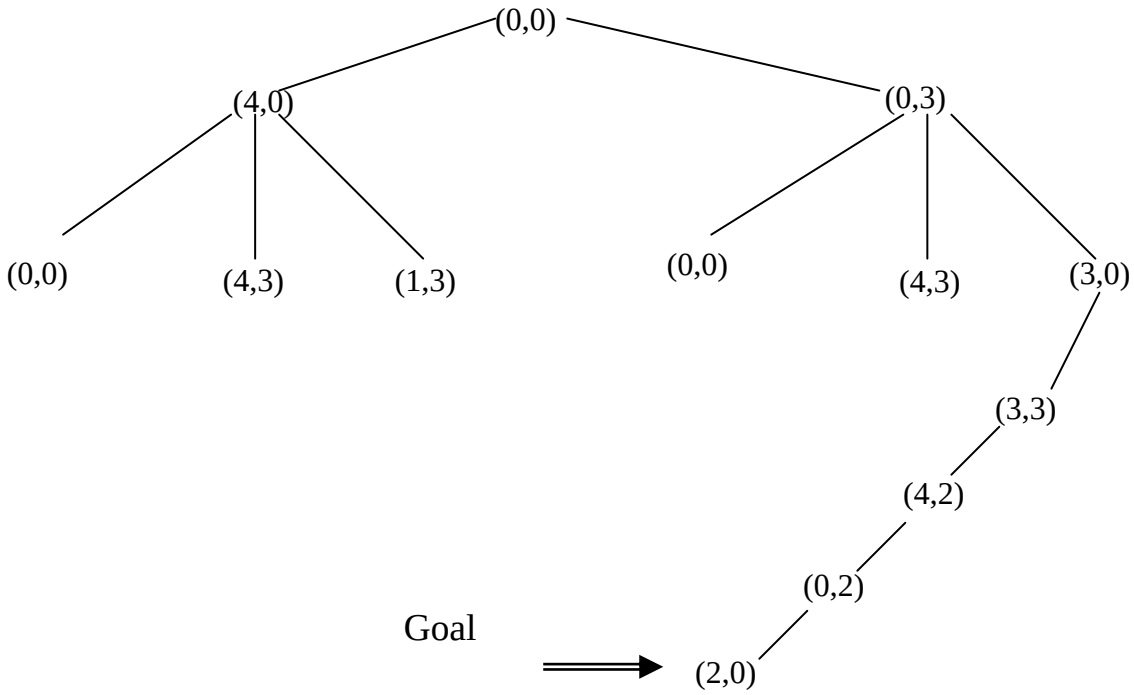
1- start state → (X,Y) → (0,0)

2- X= 0 Y= 3 → (0,3)

3- X= 3 Y= 0 → (3,0)

4- X= 3 Y= 3 → (3,3)

5- X= 4 Y= 2 → (4,2)



The search algorithm is admissible (مقبول) if the guaranteed to find minimal path goal.

خوارزمية البحث تعتبر مقبولة اذا تضمنت بشكل اكيد ايجاد اقصر طريق (مسار) للحل.

Example : Coins Problem (العملة النقدية) : head(h) and tail (t)

Initial state : hhh لو كان لدينا ثلاث عملات نقدية بالوجه الاول

Goal state : ttt والمطلوب قلب العملات النقدية الثلاثة الى الوجه الثاني

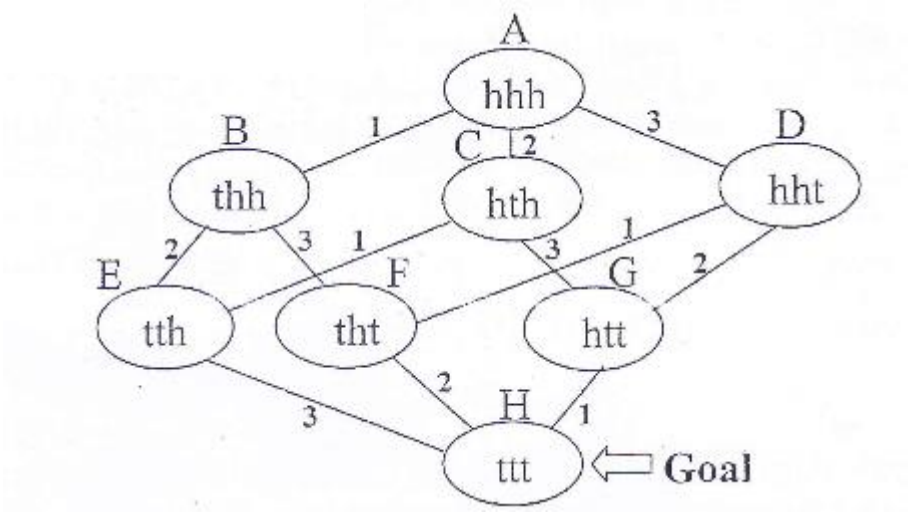
- لايجوز قلب اكثر من عملة نقدية في نفس الوقت
- لايجوز تكرار ال (Nodes)

Rules :

1- hAB ⟶ tAB قلب العملة الاولى فقط

2- AhB ⟶ AtB قلب العملة الثانية فقط

3- ABh ⟶ ABt قلب العملة الثالثة فقط



The figure above is the state space for Coins Problem, where the initial state(hhh) and the goal state (ttt).

EX: Give the state space to sliding-tile puzzle consists of two black tile, two white tile and an empty space in the configuration shown below:

B	B	∇	W	W

Initial state

W	W	∇	B	B

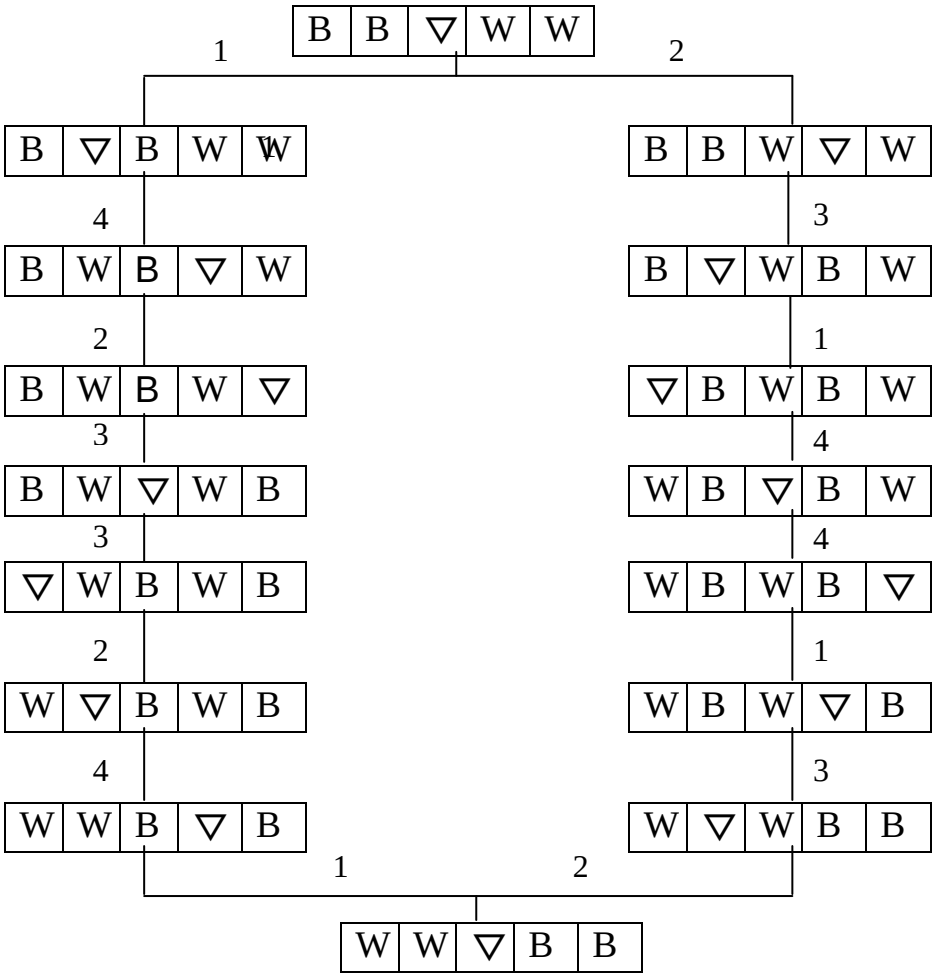
Goal state

Rules : 1- B, ∇ ⟶ ∇, B

2- ∇, W ⟶ W, ∇

3- B,W, ∇ ⟶ ∇, W, B

4- ∇, B, W ⟶ W, B, ∇



Search Algorithms

- Formal search properties

- ***Blind search:***

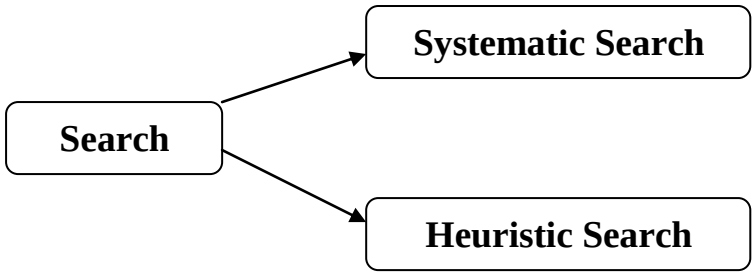
1. take solution some or all
2. stop with comparison
3. get solution from all available

- ***Heuristic search:***

1. take only best solution
2. stop only when get solution near optimal
3. good solution

- ***Optimization search:***

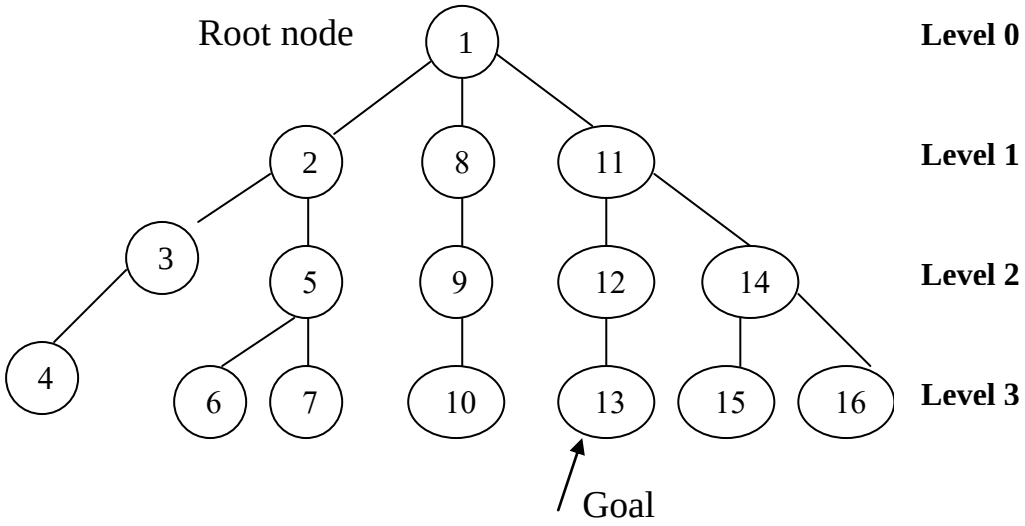
1. generate best solution
2. stop when get solution
3. optimal solution



1- Depth-First Search

Starting at the root, the algorithm proceeds to the depth of the tree first passing through all the levels and reaching the last level as defined in the problem. The nodes to the left of the tree are visited first, and when there are no more nodes to the left or the defined depth has been

reached, the search starts to visit the nodes on the right and so on until the goal is found or the tree is finished.



Advantages:
1- it is guaranteed to find a solution if one exists
Disadvantages:

1- it waists considerable time in exploring long paths that don't lead to a solution.
2- It waists a lot of time in backtracking

- ***Depth first search (DFS)method***

Step 1: from a list consisting of the root node.
Step2: until the path is empty or the goal is found, determine if the first element in the list is a goal.

a. If it is a goal, then exit with solution.
b. Otherwise, remove it from the list and add its children to front of the list.

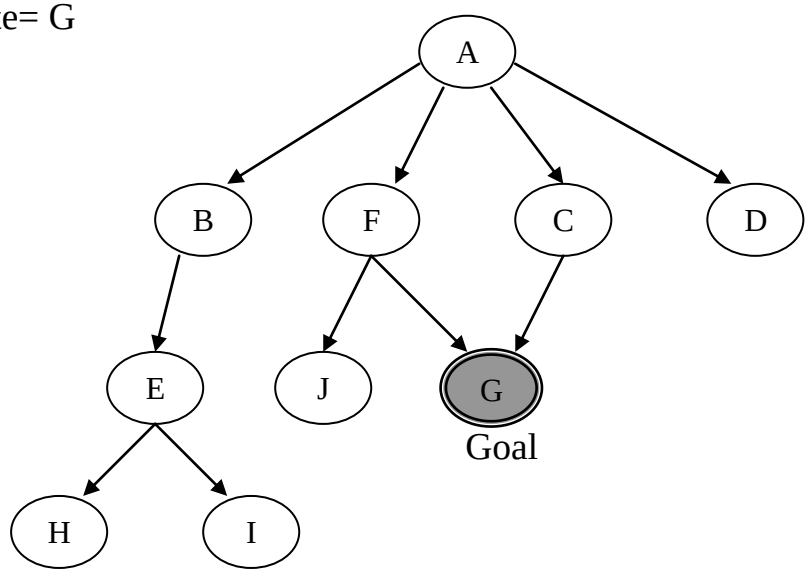
Step3: if a goal is found then declare success, otherwise declare failure.

- ***Depth first search(DFS) algorithm***

Initialize :open =[start state], close =[];
While open<>[] do
Remove the first state from left of open, call it x.
If x is the goal then return (path)
Generate all children of x
Put x in close
Remove from open any children of x already in open.

Discard any child of x already in close
Add the remaining children of x to the left of open
Return(fail) % no states left
End

Ex: find the goal (G) using depth first search algorithm to the following tree
Start state=A
Goal state= G

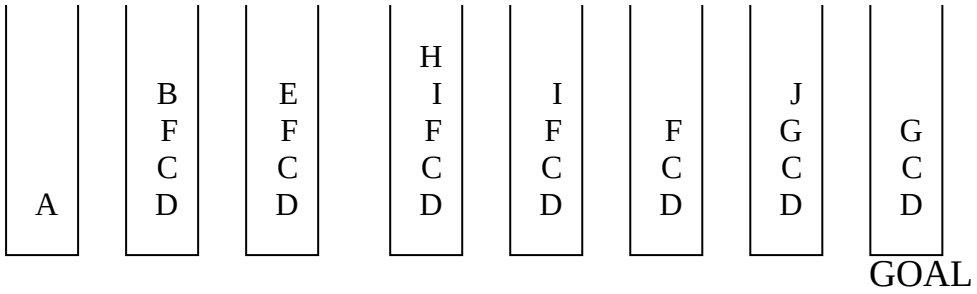


Iteration	open	Close
0	[(A,0)]	[]
1	[(B,A),(F,A),(C,A),(D,A)]	[(A,0)]
2	[(E,B),(F,A),(C,A),(D,A)]	[(B,A),(A,0)]
3	[(H,E),(I,E), (F,A),(C,A),(D,A)]	[(E,B), (B,A),(A,0)]
4	[(I,E),(F,A),(C,A),(D,A)]	[(H,E), (E,B), (B,A),(A,0)]
5	[(F,A),(C,A),(D,A)]	[(I,E), (H,E), (E,B), (B,A),(A,0)]
6	[(J,F),(G,F),(C,A),(D,A)]	[(F,A), (I,E), (H,E), (E,B), (B,A),(A,0)]
7	[(G,F), (C,A),(D,A)]	[(J,F), (F,A),(I,E), (H,E), (E,B),B,A),(A,0)]
8	G is Goal	[(G,F), (J,F), (F,A),(I,E), (H,E), (E,B), (B,A),(A,0)]

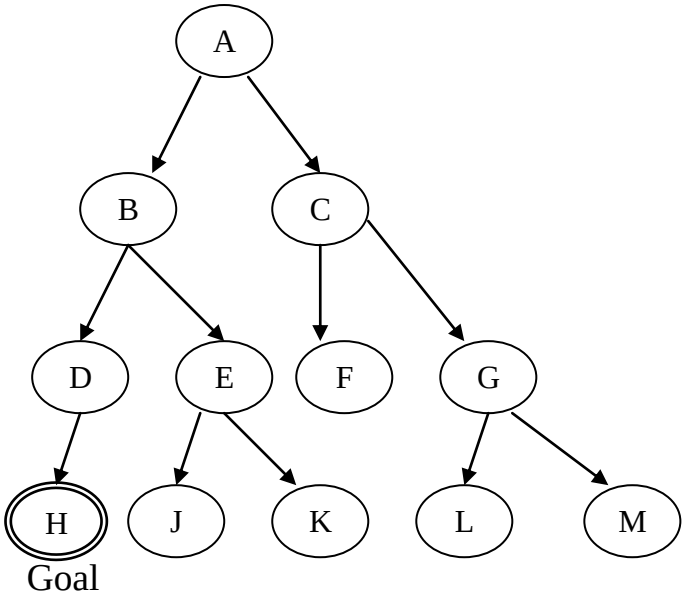
State space: all the nodes of Tree
Search space : A, B, E, H, I, F, J, G
Solution path: A ⟶ F ⟶ G

One of the most important blind search methods is the depth search method, and in this algorithm two groups are used, namely the (open, close) and in both, elements are added and removed from the left side.(Last In First Out) , the element finally entering comes out first, and

the idea of the algorithm is directed towards one path until it reaches deeper and starts from the left and then the right



Ex: find the goal (H) using depth first algorithm to the following tree:

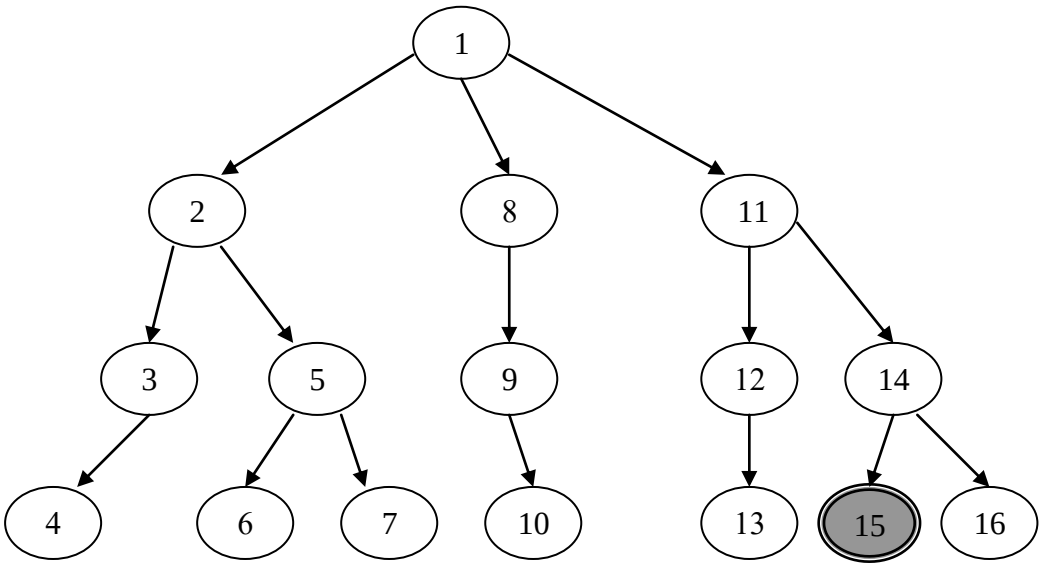


اولا: طريقة الـ (open, close)

iteration	open	close
0	[A]	[]
1	[B,C]	[A]
2	[D,E,C]	[B,A]
3	[H,E,C]	[D,B,A]
4	[E,C]	[H,D,B,A]

State space: all the nodes of tree
Search space : A, B, D, H
Solution path: A ⟶ B ⟶ D ⟶ H

EX: find the goal (15) using depth first search algorithm to the following tree



1- الحل بطريق الـ open, close

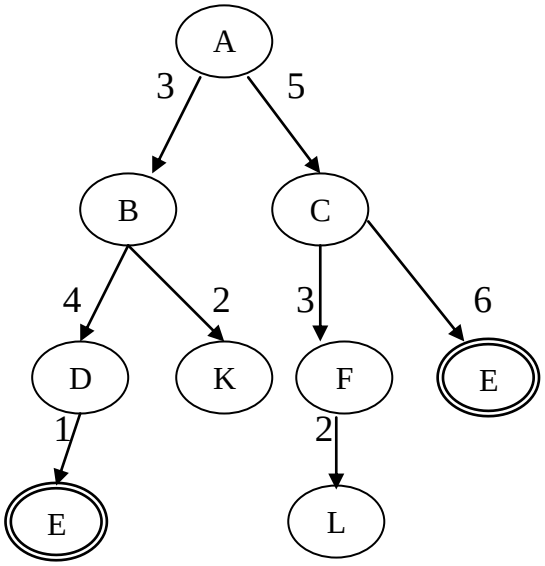
iteration	open	Close
0	[1]	[]
1	[2,8,11]	[1]
2	[3,5,8,11]	[1,2]
3	[4,5,8,11]	[1,2,3]
4	[5,8,11]	[1,2,3,4]
5	[6,7,8,11]	[1,2,3,4,5]
6	[7,8,11]	[1,2,3,4,5,6]
7	[8,11]	[1,2,3,4,5,6,7]
8	[9,11]	[1,2,3,4,5,6,7,8]
9	[10,11]	[1,2,3,4,5,6,7,8,9]
10	[11]	[1,2,3,4,5,6,7,8,9,10]
11	[12,14]	[1,2,3,4,5,6,7,8,9,10,11]
12	[13,14]	[1,2,3,4,5,6,7,8,9,10,11,12]
13	[14]	[1,2,3,4,5,6,7,8,9,10,11,12,13]
14	[15] is a goal	[1,2,3,4,5,6,7,8,9,10,11,12,13,14]

2- الحل بطريقة المصفوفة matrix

[1]
[2,8,11]
[3,5,8,11]
[4,5,8,11]
[5,8,11]
[6,7,8,11]

[7,8,11]
[8,11]
[9,11]
[10,11]
[11]
[12,14]
[13,14]
[14]
[15] is a goal

Ex: use depth first search algorithm to finding the minimum cost for reaching the goal (E) to the following tree:



iteration	Open	Close
0	[A]	[]
1	[B,C]	[A]
2	[D,K,C]	[B,A]
3	[E,K,C]	[D,B,A]
4	[K,C]	[E,D,B,A]
5	[C]	[K,E,D,B,A]
6	[F,E]	[C,K,E,D,B,A]
7	[L,E]	[F, C,K,E,D,B,A]
8	[E]	[L, F, C,K,E,D,B,A]
9	[]	[E, L, F, C,K,E,D,B,A]

State space : all nodes of the tree
Search space: A,B,D,E,K,C,F,L,E
Solution path: A ⟶ B ⟶ D ⟶ E & A ⟶ C ⟶ E

Cost of solution paths : path one (A – B – D – E) = 3+4+1=8
Path two : (A – C – E) = 5+6= 11

اذا المسار الاول هو الاقل كلفة

H.W