جامـــــعـة المـــســتـقـبـل
AL MUSTAQBAL UNIVERSITY

كلية العلـوم

قســـم علوم الذكاء الاصطناعي

المحاضرة الرابعة

المادة: Heuristic Search

المرحلة: الثانية

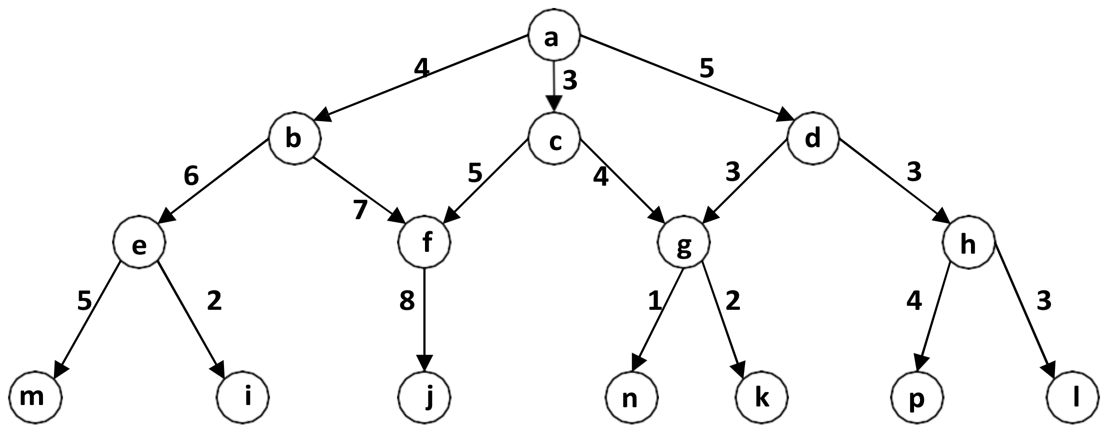
اسم الاستاذ: م.م هادي صلاح

A-search algorithm

The first advance approach to the best first search is known as A-search algorithm. An algorithm is simply defined as a best first search plus specific function. This specific function represents the actual distance (levels) between the initial state and the current state and is denoted by g(n). A notice will be mentioned here that the same steps that are used in the best first search are used in an A algorithm but in addition to the g(n) as follow;

$f(n) = h(n) + g(n)$ where $h(n)$ is the heuristic function that computes the heuristic value for each state n, and $g(n)$ is the generation function that computes the actual distance (levels) between initial state to current state n.

Example:



Find the path from **a** to **k** using A-search algorithm

Step	Current Queue/Frontier	Visited/Closed Set
1	[a]	[]
2	[b4, c3, d5]	[a]
3	[b4+1, c3+1, d5+1]	[a]
4	[c4, b5, d6]	[a]
5	[f5, g4, b5, d6]	[a, c4]
6	[f5+2, g4+2, b5, d6]	[a, c4]
7	[f7, g6, b5, d6]	[a, c4]
8	[b5, g6, d6, f7]	[a, c4]
9	[e6, f7, g6, d6, f7]	[a, c4, b5]
10	[e6+2, f7+2, g6, d6, f7]	[a, c4, b5]
11	[g6, d6, f7, e8, f9]	[a, c4, b5]
12	[n1, k2, d6, f7, e8, f9]	[a, c4, b5, g6]
13	[n1+3, k2+3, d6, f7, e8, f9]	[a, c4, b5, g6]
14	[n4, k5, d6, f7, e8, f9]	[a, c4, b5, g6]
15	[k5, d6, f7, e8, f9]	[a, c4, b5, g6, n4]
16	**Stop: Goal (k) is found**	**-**

A* Search Algorithm (A-star)

The second advance approach to the best first search is known as A*-search algorithm. A* algorithm is simply defined as a best first search plus specific function. This specific function represents the actual distance (levels) between the current state and the goal state and is denoted by g(n).

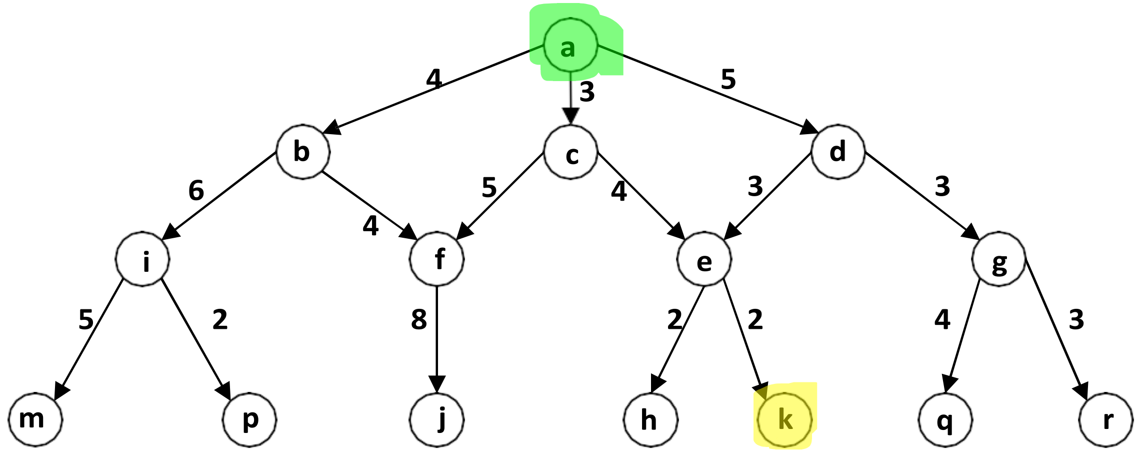
$f(n) = h(n) + g(n)$ where $h(n)$ is the heuristic function that computes the heuristic value for each state n, and $g(n)$ is the generation function that computes the actual distance (levels) between current state n to goal state.

Algorithm (Pseudo-code)

```
Function A* Search Algorithm
Begin
Open: = [Initial state];    %initialize
Closed: = [ ];
While open <> [ ] do  %states remain
Begin
Remove leftmost state from open, call it X;
If X = goal then return the path from initial to X
Else Begin
Generate children of X; For each child of X do Begin
Add the distance between current state to goal state to the heuristic
value for each child  %make the g(n)
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.
```

Example:



Find the path from **a** to **k** using A*-search algorithm

Open	Closed
[a]	[ ]
[b4, c3, d5]	[a]
[b4+4, c3+2, d5+2]	[a]
[c5, d7, b8]	[a]
[f5, e4, d7, b8]	[a, c5]
[f5+3, e4+1, d7, b8]	[a, c5]
[e5, d7, f8, b8]	[a, c5]
[h2, k2, d7, f8, b8]	[a, c5, e5]
[h2+2, k2+0, d7, f8, b8]	[a, c5, e5]
[k2, h4, d7, f8, b8]	[a, c5, e5]

Stop, the goal (k) is found.

Heuristic Search Methods with Heuristic Function

Hill climbing

For each state $f(n) = h(n)$ where $h(n)$ is the heuristic function that computes the heuristic value for each state n.

Best First Search

For each state $f(n) = h(n)$ where $h(n)$ is the heuristic function that computes the heuristic value for each state n.

A-search algorithm

$f(n) = h(n) + g(n)$ where $h(n)$ is the heuristic function that computes the heuristic value for each state n, and $g(n)$ is the generation function that computes the actual distance (levels) between initial state to current state n.

A*-search algorithm

$f(n) = h(n) + g(n)$ where $h(n)$ is the heuristic function that computes the heuristic value for each state n, and $g(n)$ is the generation function that computes the actual distance (levels) between current state n to goal state.

Problems with Hill Climbing Search

Hill Climbing is a simple search algorithm. It always moves to the neighbor state that has a better heuristic value. However, it has some problems.

1) Local Minima (First Hill)

This problem **stops the search process**.

A local minimum happens when the algorithm reaches a state that is better than its neighbors, but it is not the goal state. Since all nearby states are worse, the algorithm thinks it cannot improve more, so it stops.

The goal may still exist in the search space, but the algorithm cannot reach it because it does not go back to try other paths.

Why does this happen?

Because Hill Climbing does not use backtracking. It only looks at nearby states.

How can we solve it?

- Use Backtracking
- Use Random Restart Hill Climbing
- Use Simulated Annealing

2) Plateau Problem

This problem also **stops the search process**.

A plateau happens when the algorithm reaches a state where two or more neighbor states have the same heuristic value. All choices look equal, so the algorithm does not know which direction to choose.

Because of this, the search may stop even though the goal exists.

Why does this happen?

Because the heuristic values of neighbor states are equal.

How can we solve it?

- Allow sideways moves
- Choose one direction (for example, always choose the left one)
- Use random selection

3) Ridge Problem

This problem **does not stop the search**, but it may give a non-optimal solution.

A ridge happens when the best path to the goal is not directly reachable by simple local moves. The algorithm moves step by step, but the best path may require a more complex direction.

So, the algorithm may reach the goal, but with higher cost (not the best solution).

Why does this happen?

Because Hill Climbing only checks limited directions at each step.

How can we solve it?

- Apply more than one rule at each step
- Use more advanced search algorithms

	Blind Search	Heuristic Search
1	In term of complexity: it is less complex.	In term of complexity: it is more complex.
2	In term of memory capacity: usually need more memory capacity.	In term of memory capacity: usually need less memory capacity.
3	In term of run time consuming: usually consumes more run time.	In term of run time consuming: usually consumes less run time.
4	Guarantee for solution.	Guarantee for solution, except Hill Climbing (not always).
5	Usually does not find the optimal solution path.	Usually finds the optimal solution path or nearly the optimal solution path.
6	It does not have a guider in search behavior.	It has a guider in search behavior (Heuristic Function).
7	It is not efficient in game playing.	It is efficient in game playing such as Minmax or Alpha-Beta procedures.