# AI notes

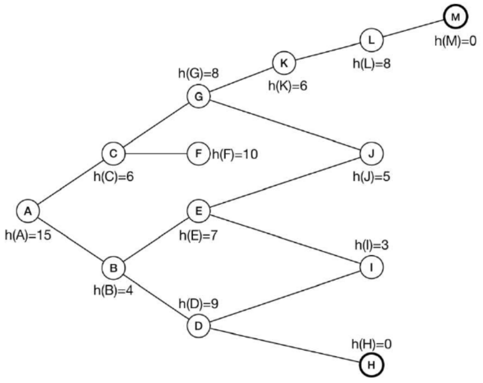
## Heuristic search

### Greedy Best-first Graph Search (GBFS)

GBFS is a search algorithm that opens all connected nodes and goes to the cheapest node based on the heuristic value of that node. Every time new nodes are discovered they are added to the (search database) or in other words a table that contains all known nodes. GBFS will then always go to the cheapest node in that search database. Lets do an example on the search path on this Heuristic search tree with GBFS:

Our stating state will be A, and our goal states will be M and H.

Question: Will we get to M or H using GBFS?



Let's start by discovering all connected nodes to A and add then to a table:

| | | |
|---|---|---|
| A- | C | B |
| 15 | 6 | fi |

We will use – to indicate that the node is closed. We have closed A because it has the highest value in our tree so that would mean that after being there, we should never go back based on the logic of GBFS. Ps we close nodes after they have been explored!

Ok next! the node with the lowest heuristic value is B, so let's go to B and discover which nodes there are connected to it and added them to our table.

| | | |
|---|---|---|
| C | E | D |
| 6 | 7 | 9 |

Our lowest value is not C so we will go to C and discover the nodes connected there and again add them to our table. ok I think you get my point, so let me just write the path out for you from start to finish:

A->B->C->E->I->J->G->K->L->M

So the answer is M!

## A* search Algorithm

For A* the way it works is it also takes the cheapest way to the goal state, but the cost is determined be adding the step cost to get to the goal state plus the heuristic value of the node. With a step cost of 1 all the nodes have these values:

| | Goal H | Goal M |
|---|---|---|
| **A** | 18 | 20 |
| B | 6 | 10 |
| C | 10 | 10 |
| D | 10 | 16 |
| E | 10 | 12 |
| F | 15 | 15 |
| G | 13 | 11 |
| ***H*** | 0 | Na |
| I | 5 | 9 |
| J | 9 | 9 |
| K | 12 | 8 |
| L | 15 | 9 |
| ***M*** | Na | 0 |

So in our case with the same tree as before the path would be as follows:

A->B->D->H

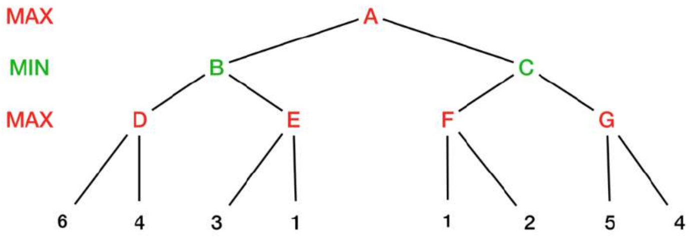
## Admissible

Question is our Heuristic function h admissible?

False because the value of D is higher than 1 and is not equal or lower than the step cost. To be admissible the heuristic value of each node of the path to the goal state should be lower or equal to the number of steps it costs to get the goal state.

## Adversarial search (minimax)

Adversarial search using minimax is done by starting from left to right and at the end of a branch. Start at the end of a branch and find either the minimum or maximum of the terminal states values that are connected to the final non-terminal node. VISUAL EXAMPLE:



In this case we start at D(x) and find the max value between our terminal states 6 and fi. In this case that would be 6, so D is now D(6). We do the same with E. Now for be with take the minimum value of our to connected nodes, which in this case would be 3. This is done all through the tree and the final result should look like this.

D(6) – E(3) – B(3) -> A(3,x)

F(2) – G(5) – C(2) -> A(3,2)

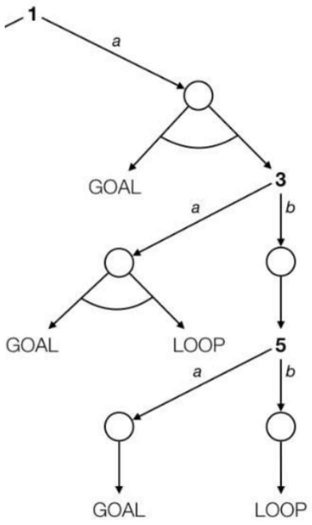
A(3)

## Alpha-Beta

We can optimize this process by doing what is called an Alpha-Beta search. Alpha-Beta is a variation of the previous minimax search, but that we prune branches. Pruning is when we don't explore branches that cannot yield a result that can have a effect on our pre-existing ones. Example of this is when we explore the right side of our tree we can see that F is going to be 2, then when we open G the first one is 5 that means that G can only be equal <=5. This means that we don't need to explore fi so we can prune that.

## Non-deterministic search

For a Non-deterministic search we want to insure that the path we chose **ONLY** results in a goal state, and does not risk a loop. Example:



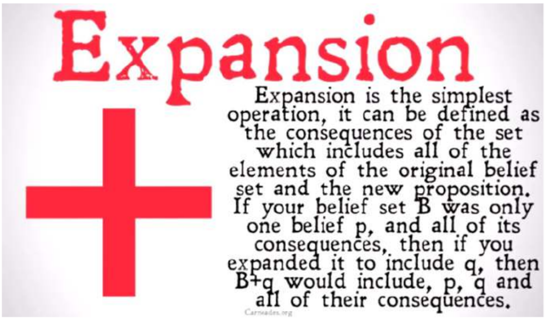
For this the non-deterministic path would be:

$\Pi(1) = a, \Pi(3) = b, \Pi(5) = a$

## Belief logic

### Operators

- **Expansion:** Adding beliefs to some set of beliefs which do not contradict something already there

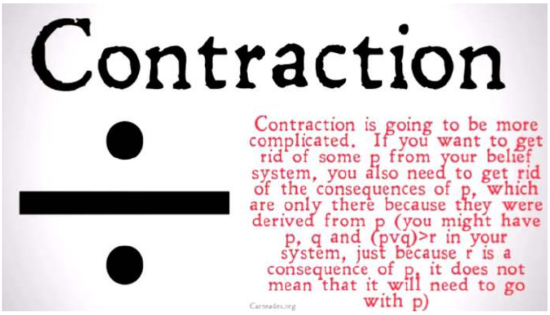


Expansion

Expansion is the simplest operation, it can be defined as the consequences of the set which includes all of the elements of the original belief set and the new proposition. If your belief set B was only one belief p, and all of its consequences, then if you expanded it to include q, then B+q would include, p, q and all of their consequences.

Carneades.org

- **Contraction:** Removing beliefs and their consequences from a set.

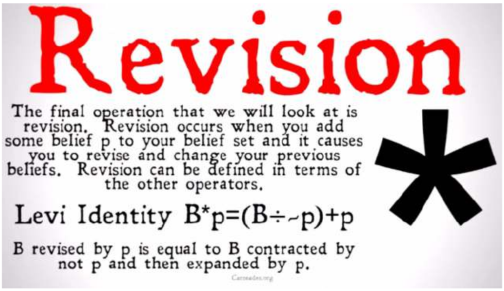


Contraction

Contraction is going to be more complicated. If you want to get rid of some p from your belief system, you also need to get rid of the consequences of p, which are only there because they were derived from p (you might have p, q and (pvq)>r in your system, just because r is a consequence of p, it does not mean that it will need to go with p)

Carneades.org

- **Revision:** Adding one belief which contradicts some other belief in the system and remove them from the set.



## Revision

The final operation that we will look at is revision. Revision occurs when you add some belief p to your belief set and it causes you to revise and change your previous beliefs. Revision can be defined in terms of the other operators.

Levi Identity $B^{*}p=(B\div{\sim}p)+p$

B revised by p is equal to B contracted by not p and then expanded by p.

Carneades.org