Search

Search Space:

- A search space is a graph with partual solutions or states as nodes and possible transitions as arcs
- They may contain cycles
- They are often approximated by a tree: Easier algorithms, requires less memory, less efficient due to more repititions
- Blind search methods looks for goals systematically but do not account for the quality of partual solutions (Using Heuristics)
- Different search strategies can be modelled with the notion of an agenda. A list of nodes which we plan to search next

```
search(Agenda, Goal) :-
    next(Agenda, Goal, Rest),
    Goal(Goal).

search(Agenda, Goal) :-
    next(Agenda, Current, Rest),
    Children(Current, Children),
    add(Children, Rest, NewAgenda),
    search(NewAgenda, Goal).
```

How 'Add' is defined determines wether this is a bredth first or a depth first search. Given an agenda which represents a set of nodes at the frontier of the search process. It will return a Goal node that is accessible from the nodes on the agenda

```
search_df([Goal|Rest], Goal) :-
    goal(Goal).
search_df([Current|Rest], Goal) :-
    Children(Current, Children),
    Append(Children, Rest, NewAgenda),
    search_df(NewAgenda, Goal).

search_bf([Goal|Rest], Goal) :-
    goal(Goal).
search_bf([Current|Rest], Goal) :-
    Children(Current, Children),
    Append(Rest, Children, NewAgenda),
    search_bf(NewAgenda, Goal).
```

Depth-First:

- Agenda = Stack

- Incomplete - May get trapped in an infinite branch
- No shortest-path property
- Requires O(BxN) memory where N is the depth of the tree and B is the number of branches at each node

Breadth-First Search:

- Agenda = Queue
- Complete - Will find all solutions
- First solution is found along the shortest path
- Requires O(B$ ^n $) memory

Iterative Deepening:

- Combines good features of df and bf
- Repeats work but has better memory complexity.

Best-First Search:

```
search_bstf([Goal|Rest], Goal) :-
    goal(Goal).

search_bstf([Current|Rest], Goal) :-
    children(Current, Children),
    add_bstf(Children, Rest, NewAgenda),
    search_bstf(NewAgenda, Goal).
```

- Agenda = Priority Queue
- Behaviour depends on the heuristic
- Certain guarantees can be made if the heuristic satisfies certain properties

A Search:

A search is a best first search algorithm that aims at minimising the total cost along a path from start to goal.

$f(n) = g(n) + h(n) f(n)$: Estimate of total cost along path through n$g(n)$: Actual cost to reach n from the sta

A heuristic is globally optimistic or admissable if the estimated cost of reaching a goal is always less than the actual cost from any node n.

$$h(n) \leq h^*(n)$$

A heuristic is monotonic or locally optimistic if it never decreases by more than the cost c of an edge from a node n1 to a child n2.

$$h(n_1) - h(n_2) \leq c$$

Distance Heuristics:

Minkowsksi Distance:

$$D(X,Y) = \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{\frac{1}{p}} p = 1 : \text{Manhatten} p = 2 : \text{Euclidean} p = \infty : \text{Chebyshev}$$

A* Algorithm:

An A algorithm with an admissable heuristic. It always reaches the goal along the cheapest path first, breadth-first search is a special case of this.

Monotonicity makes the search more efficient:

- The first time a node is put on the agenda it is reached along the cheapest path
- Its called monotincity becauses f-values are never decreasing along a path
- In the abscene of monotinicity, the agenda may contain multiple copies of the same node along multiple paths