Search Techniques

We will explore 3 different search algorithms (out of very many variants). The task is to get from an original position to a target position via nodes connected by links to other nodes, minimizing some total cost along the way. That cost could be:

- the total number of steps (one step = transition along a link from a node to its neighbor)
- total distance covered (each link between a pair of nodes has a distance cost known beforehand)
- total time (each link has a known time-cost)

The search techniques are:

- **Lowest-path-cost uninformed search**:
  - In this situation, you have no information about the position of the target (hence "uninformed")
  - With this technique, you are choosing to concentrate on the node with the minimum path cost so far, and choose its neighbors as candidates to look at (put into the frontier)
- **Nearest-to-target "greedy" informed search**:
  - In this situation, you can get an estimate (guess) of the distance (cost) from any node to the target
  - At any node, we'll use that guess to find the nodes in the frontier that are the closest to the target, and choose those to explore
- **A* total path cost informed search**:
  - In this one, for each candidate node, we'll use the sum of its actual distance (cost) from the origin plus the guessed cost to the target as the selection criterion for which node in the frontier to concentrate on.
  - For this search, the estimated (guessed) cost from any node to the target must never exceed that actual cost of getting there. It must be optimistic. For instance, when walking from one location to a target in Manhattan, a valid optimistic estimate would be the straight-line distance from the place to the target. This is usually less than the actual distance you have to walk (since most of us can't fly) – it is never longer than the actual distance.

Some cost terminology:

- For each node N, **g(N)** is the total cost to get to N from the origin along the particular path taken to get to N (which may not be the minimum cost path).
- For the informed searches, for each node N, **h(N)** is the guessed (estimated) cost from N to the target. For A* search, this is an optimistic estimate, which must never be greater than the actual cost to get to the target (but may be equal to it).

How the searches work:

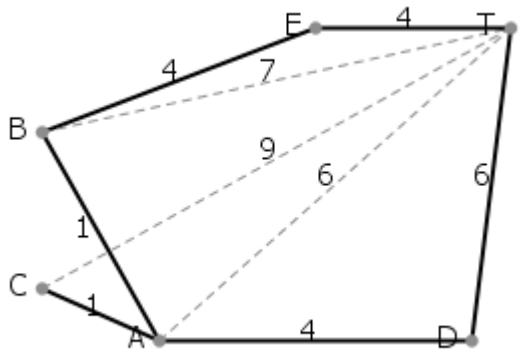
- There are 3 named collections of nodes: ***unexplored***, ***frontier***, and ***explored***. At any one time, each node is a member of exactly one of those collections. At the beginning of a search, every node is in the *unexplored* group. In general, a node moves from the *unexplored* group to the *frontier group* and then to the *explored* group.
- As a first step, the origin node is placed into the *frontier* group and removed from the *unexplored* group.
- Then the following dance takes place:
    - A node is chosen from the *frontier* group. Each of the three searches uses a different choosing algorithm (see below). The *frontier* group is best stored in a priority queue, with the choosing algorithm encoded in the queue's comparison function.
    - If the node is the target, then we're done in finding the target and the search ends. There may be some post-search work to calculate and return the total cost, or to provide the path taken from the origin to the target.
    - If the node is not the target, but is already in the *explored* group, then ignore the node, and go on to the next best node in the *frontier*. This may happen because there has been more than one path to this node, and a shorter path has already been explored, and this is an equivalent cost or longer path. In either case, ignore this node because it has already been processed.
    - If the node is not the target, the node is removed from the *frontier* and put into the *explored* group. Then each of the node's neighbors is examined:
        - If the neighbor is in the *unexplored* group, its current cost is calculated (and assigned to the node), and it's removed from *unexplored* and added to the *frontier*.
        - If the neighbor is in *explored*, ignore it
    - If the *frontier* is empty, we've failed to find the target (look over your shoulder, suspect sabotage).

The frontier choice algorithms:

Every node, when it is retrieved from the *unexplored* group, has its cost calculated, which the node then keeps as one of its properties. Each of the different search techniques calculates the node's cost differently. Since the *frontier* is housed in a priority queue whose order is controlled by the costs of the nodes, each search will end up choosing nodes differently – and that is what controls the behavior of the search.

- **Uninformed search**: each node's cost is g(N). Since the node (B) will be retrieved because it is a neighbor of a node (A) that we've just retrieved from the *frontier*, we can calculate B's cost as A's cost plus the cost of moving from A to B.
- **Greedy search**: each node's cost is h(N). Remember that h(N) is the guessed (and optimistic) cost of moving from this node to the target.
- **A* search**: each node's cost is g(N)+h(N)

Sample Graph:



In this graph, we're trying to get from node A to the target node T. The dark lines are actual links between the nodes, each labelled with its cost. The dashed lines are the estimated costs to get to the target T from A, B and C.

Working through the Uninformed Search:

Notation for nodes below: (node-name; cost; path-to-node). Therefore (X;15;F,G,Q) means that we're looking at node X, which has a cost of 15, and the path to get to X is F->G->Q->X.

When we choose a node from the frontier, we'll always choose the one with the least cost. If there's more than one node with the least cost, we'll choose one of them at random.

ACTION	NODE EXAMINED	UNEXPLORED	FRONTIER	EXPLORED
		A,B,C,D,E,T		
Start: place A into frontier		B,C,D,E,T	(A;0)	
Choose A from frontier	(A;0)	B,C,D,E,T		
Put A's neighbors into frontier, and put A into explored		E,T	(B;1;A), (C;1;A), (D;4;A)	A
Choose B from frontier	(B;1;A)	E,T	(C;1;A), (D;4;A)	A
Put B's neighbors into frontier		T	(C;1;A), (D;4:A) (E;5;A,B)	A,B
Choose C	(C;1;A)	T	(D;4:A) (E;5;A,B)	A,B
Put C's neighbors into frontier (none)		T	(D;4:A) (E;5;A,B)	A,B,C
Choose D	(D;4;A)	T	(E;5:A,B)	A,B,C
Put D's neighbors into frontier			(E;5,A,B), (T;10;A,D)	A,B,C,D
Choose E	(E;5;A,B)		(T;10;A,D)	A,B,C,D
E's neighbor is T, already in the frontier, but replace it with the shorter path through E			(T;9;A,B,E)	A,B,C,D,E
Choose T	(T;9;A,B,E)			
Is target, have winner! Path length: 9 Path: A,B,E,T (shortest path found, more steps than Greedy or A*)	T			

Working through the Greedy Search:

ACTION	NODE EXAMINED	UNEXPLORED	FRONTIER	EXPLORED
		A,B,C,D,E,T		
Start: place A into frontier		B,C,D,E,T	(A;6)	
Choose A from frontier	(A;6)	B,C,D,E,T		
Put A's neighbors into frontier, and put A into explored		E,T	(B;7;A), (C;9;A), (D;6;A)	A
Choose D from frontier	(D;6;A)	E,T	(B;7;A), (C;9;A)	A
Put D's neighbors into frontier		E	(B;7;A), (C;9;A), (T;0;A,D)	A,D
Choose T from frontier	(T;0;D)	E	(B;7;A), (C;9;A)	A,D
Is target, have winner! Path length: 10 Path: A,D,T (fewest steps, but not shortest path)	T			

Working through the A* Search:

ACTION	NODE EXAMINED	UNEXPLORED	FRONTIER	EXPLORED
		A,B,C,D,E,T		
Start: place A into frontier		B,C,D,E,T	(A;6)	
Choose A from frontier	(A;6)	B,C,D,E,T		
Put A's neighbors into frontier, and put A into explored		E,T	(B;8;A), (C;10;A), (D;10;A)	A
Choose B from frontier	(B;8;A)	E,T	(C;10;A), (D;10;A)	A
Put B's neighbors into frontier		T	(C;10;A), (D;10;A), (E;9;A,B)	A,B
Choose E	(E;9;A,B)	T	(C;10;A), (D;10;A)	A,B
Put E's neighbors into frontier			(C;10;A), (D;10;A), (T;9;A,B,E)	A,B,E
Choose T	(T;9;A,B,E)			
Is target, have winner! Path length: 9 Path: A,B,E,T (shortest path found, more steps than Greedy, fewer steps than Uninformed)	T			