A common search algorithm, searching for a goal state:

```plaintext
fringe := make-queue(initial-node)
loop
  if empty?(fringe) then return FAIL
  else
    X := remove-front(fringe)
    if satisfies-goal(state(X)) then return X
    else
      fringe := insert(fringe, expand(X)) /*important operation*/
  end loop
```

A* is a special case for this search where each node n is entered in the fringe with priority $f(n) = h(n) + g(n)$. Where $h(n)$ is a estimated cost of reaching the goal from n and $g(n)$ is the cost of getting to n from the start state.

**H(n) is called heuristic function. H(n) has following properties.**
It never over-estimates the cost of reaching to the goal. (Admissible)
It is consistent or monotonic. Never decreasing.

$$F(n') = g(n') + h(n') \geq g(n) + h(n) = f(n)$$

Straight-line heuristic, as discussed in the class for robot navigation problem, is consistent.
At node 2, we have $h = 3$ and $g = 2$. We can see that there is no way to move distance 1 (increasing $g$ by 1) and reducing $h$ by more than 1.

**Properties that can be proven by h being monotonic:**
Search is optimal if $h(n)$ is monotonic
Every monotonic heuristic is also admissible.

**Some example of good heuristics:**
For robot navigation problem a good heuristic was discussed in the class. A straight line distance between goal and n.
Ideas of 8 puzzle?

- $H_1 = \text{Number of dislocated blocks}$
- $H_2 = \text{Manhattan distance of blocks}$

An example showing a solution for a 3-puzzle using $H_1$. 

![Diagram of 3-puzzle solution using H1 heuristic]