

## Search

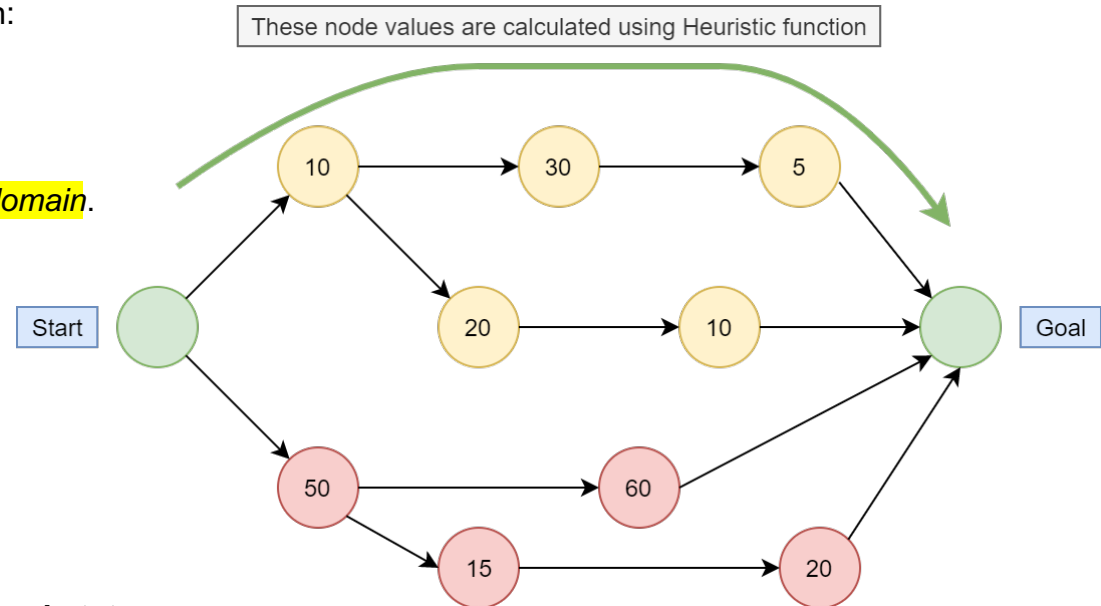
• A lot of AI boils down to search—searching through:

- **states**
- **assignments of values**
- orderings of a **route**

• **Uninformed or unsupervised search**

does not require of **any knowledge of the problem domain**.

It is a **brute-force** approach.



• Terminology:

• We begin in some **starting state**, searching for a **goal state**.

• We have some method of **transitioning** from the **current** state to 1 or more **successor** states.

• Unsupervised search offers no method of selecting one transition over another for a given state.

There's no **fixed** metric to tell us if we're **getting closer** to the goal until we're there.

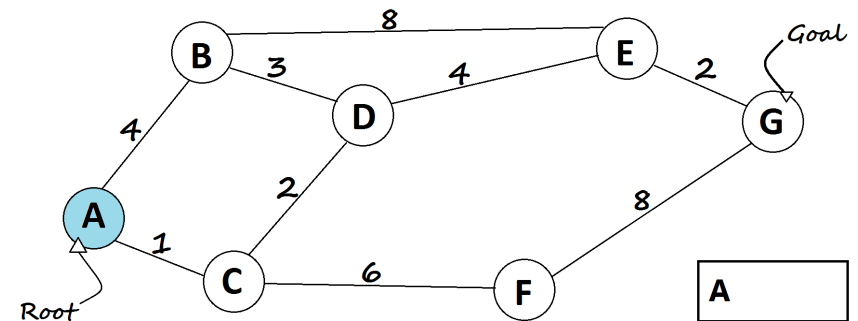
• One partial solution:

Use some measurement (derived from purely **local** information) to make a selection from offered successor states.

Search methods which use this solution are said to employ a **greedy algorithm**

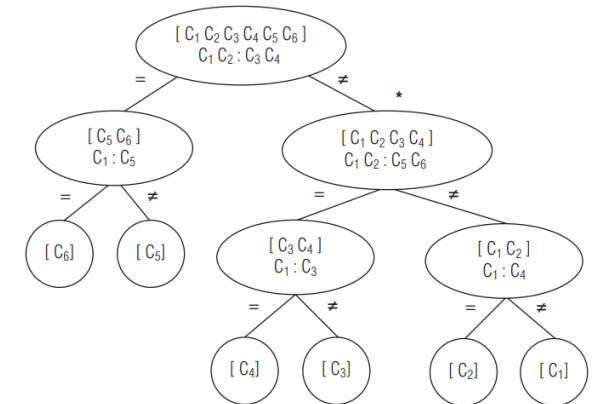
## Evaluating Search Methods

- We obviously need some way of determining whether we're at the **goal** state
- The **path length** between **two states** is the **number of transitions necessary** to move between them.
- If each transition has its own **independent** cost that may differ from 1, the path cost is the **sum of all transitions** on the path
- Note that in this case the **lowest** cost path to the goal **may not be the shortest** (lowest number of transitions)
- Depending on the problem, state transitions **may** –or– **may not** be **reversible**
- We can produce a **state-space graph** showing the possible transitions.
- The maximum or average **number of transitions** existing out of a particular state is the **branching factor**
- If we select only transitions that avoid cycles, a search tree results



## False Coin Problem

- Each state is a node (the list of which coin might be false, and what should be done at each), and the path chosen on the result of each comparison.
- As every possible outcome is accounted for, and each path terminates with exactly 1 solution, this is a map of the search space that must be dealt with.
- The total state-space map of a problem contains every state the problem might be in, and all transitions between states; obviously it can become quite complex
- Sometimes we want to find out if any path from a start state exists to any solution; other times we want to find a shortest or lowest-cost or in some way optimal path
- Note that it may be convenient to represent a state space graph as having more than 1 node for a specific state; see fig. 2.2, p. 48



- A sample search tree showing a solution to the 6-coin False Coin problem is on p. 47.

One of these coins is fake



- One method of finding a solution (in cases where we just want to know if any solutions exist) is to just generate all possible states, testing each to see if it's a goal.

### Move Solver

- One example is the [N-queens problem](#)

- In chess, the queen can move horizontally along rows, vertically along files, or along either diagonal, and attacks every square it can move to.

- The N-queens problem is: On an NxN board, place N queens such that **no queen is attacking any other**.

- We could just generate every permutation of N queens on an NxN board and pick out the solution(s)

- For N = 10 (so a 10x10 board), there are 100 ways to place the first queen, 99 ways to place the second, etc., so total placements, or in general  $(n^2/n)$

- We can reduce the size of this by observing, for example, that each row can contain only 1 queen, so begin by placing 1 queen in each

row and only move queens along rows

- Terminology:

- An algorithm is correct if it can find a valid solution.

- It is complete if it can find every solution; either every solution that exists, or every solution reachable from a given start state.

- It is optimal if it finds the best (lowest-cost, nearest, whatever) solution.

- It is optimally efficient if it finds the solution at least as fast as any other algorithm (in big-O form).

- It is nonredundant if any state rejected as a possible solution is not proposed again.

- It is informed if it is able to limit its proposals in some way rather than blindly generating every possible state (for example, every

possible placement of N queens on the board)

## Strategies

- Exhaustive enumeration consists of generating all possibilities
- But this can lead to wasted effort
- If the first 2 queens we place are attacking each other, there's no point in placing the other  $N-2$ .
- While generating a state, we should verify that the partially-constructed solution satisfies all constraints
- If not, we try another; if no further progress can be made, we must backtrack, undoing part of what we have done so far, and making another attempt; if none can be found from there, we backtrack another step, and so on.
- Backtracking to solve 4-Queens, p 50-52