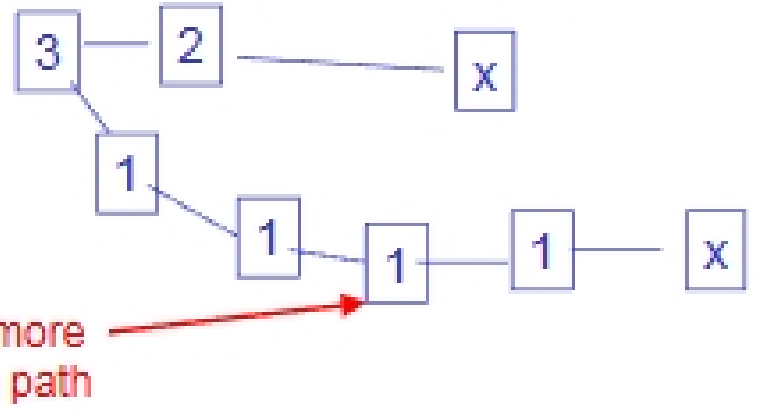
Optimal searching

Best-first search review

- Advantages
  - Takes advantage of domain information to guide search
  - Greedy advance to the goal
- Disadvantages
  - Considers cost to the goal from the current state
  - Some path can continue to look good according to the heuristic function



Branch & Bound

- Use current cost (past cost) to a node
- Pick best (lowest) cost.
- If *f* is our evaluation function for node *n*, *f*(*n*)= *g*(*n*) [g= cost 'gone so far'
  g ≥ 0
- B&B: sort queue in order of lowest *f*, & make sure not to pursue identical paths with higher costs than known costs

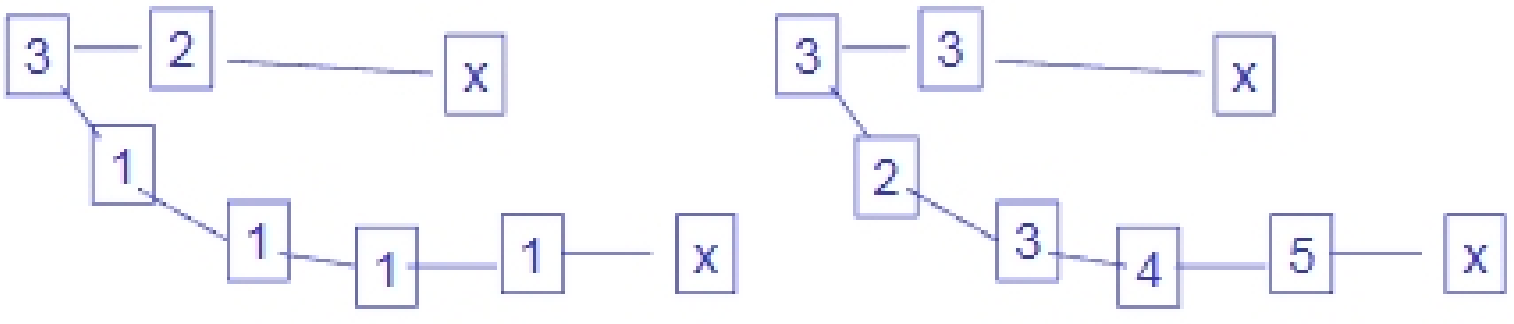
The A* Algorithm: combining past with future

- Now Consider the overall cost of the solution.

f(n) = g(n) + h(n) where g(n) is the path cost to node

g= distance gone so far h= future estimated cost

think of f(n) as an estimate of the cost of the best solution *going through the node n*



UCS, BFS, Best-First, and A*

- $f = g + h$ → A* Search
- $h = 0$ → Uniform cost search
- $g = 1, h = 0$ → Breadth-First search
- $g = 0$ → Best-First search

Admissible Heuristics

- This is not quite enough, we also require h be *admissible*:
  - a heuristic h is admissible if h(n) < h*(n) for all nodes n,
  - where h* is the actual cost of the optimal path from n to the goal
- Examples:
  - travel distance straight line distance must be shorter than actual travel path
  - tiles out of place each move can reorder at most one tile distance of each out of place tile from the correct place each move moves a tile at most one place toward correct place