CS 235: Artificial Intelligence

Week 3

Heuristic (Informed) Search

Dr. Moumita Roy
CSE Dept., IIITG

Reference: http://ai.stanford.edu/~latombe/cs121/2011/schedule.htm

Recall that the ordering of FRINGE defines the search strategy

Search Algorithm #2

SEARCH#2

1. INSERT(initial-node,FRINGE)
2. Repeat:
   a. If empty(FRINGE) then return failure
   b. N ← REMOVE(FRINGE)
   c. s ← STATE(N)
   d. If GOAL?(s) then return path or goal state
   e. For every state s' in SUCCESSORS(s)
      i. Create a node N' as a successor of N
      ii. INSERT(N',FRINGE)

Best-First Search

- It exploits state description to estimate how "good" each search node is
- An evaluation function f maps each node N of the search tree to a real number
  $f(N) \geq 0$
  [Traditionally, f(N) is an estimated cost; so, the smaller f(N), the more promising N]
- Best-first search sorts the FRINGE in increasing f
  [Arbitrary order is assumed among nodes with equal f]
- The strategy is identical to that for uniform cost search; except the use of f instead of g to order the priority queue.

How to construct f?

- Typically, f(N) estimates:
  - either the cost of a solution path through N

    Then f(N) = g(N) + h(N), where
    - g(N) is the cost of the path from the initial node to N
    - h(N) is an estimate of the cost of a path from N to a goal node
  - or the cost of a path from N to a goal node

    Then f(N) = h(N)

A* search algorithm

Greedy best first search

Heuristic function

- But there are no limitations on f. Any function of your choice is acceptable.

  But will it help the search algorithm?

Heuristic Function

- The heuristic function $h(N) \geq 0$ estimates the cost to go from STATE(N) to a goal state

  Its value is **independent of the current search tree**; it depends only on STATE(N) and the goal test GOAL?

- Example:

5		8
4	2	1
7	3	6

STATE(N)

1	2	3
4	5	6
7	8	

Goal state

$h_1(N)$ = number of misplaced numbered tiles = 6

Other Examples

5		8
4	2	1
7	3	6

STATE(N)

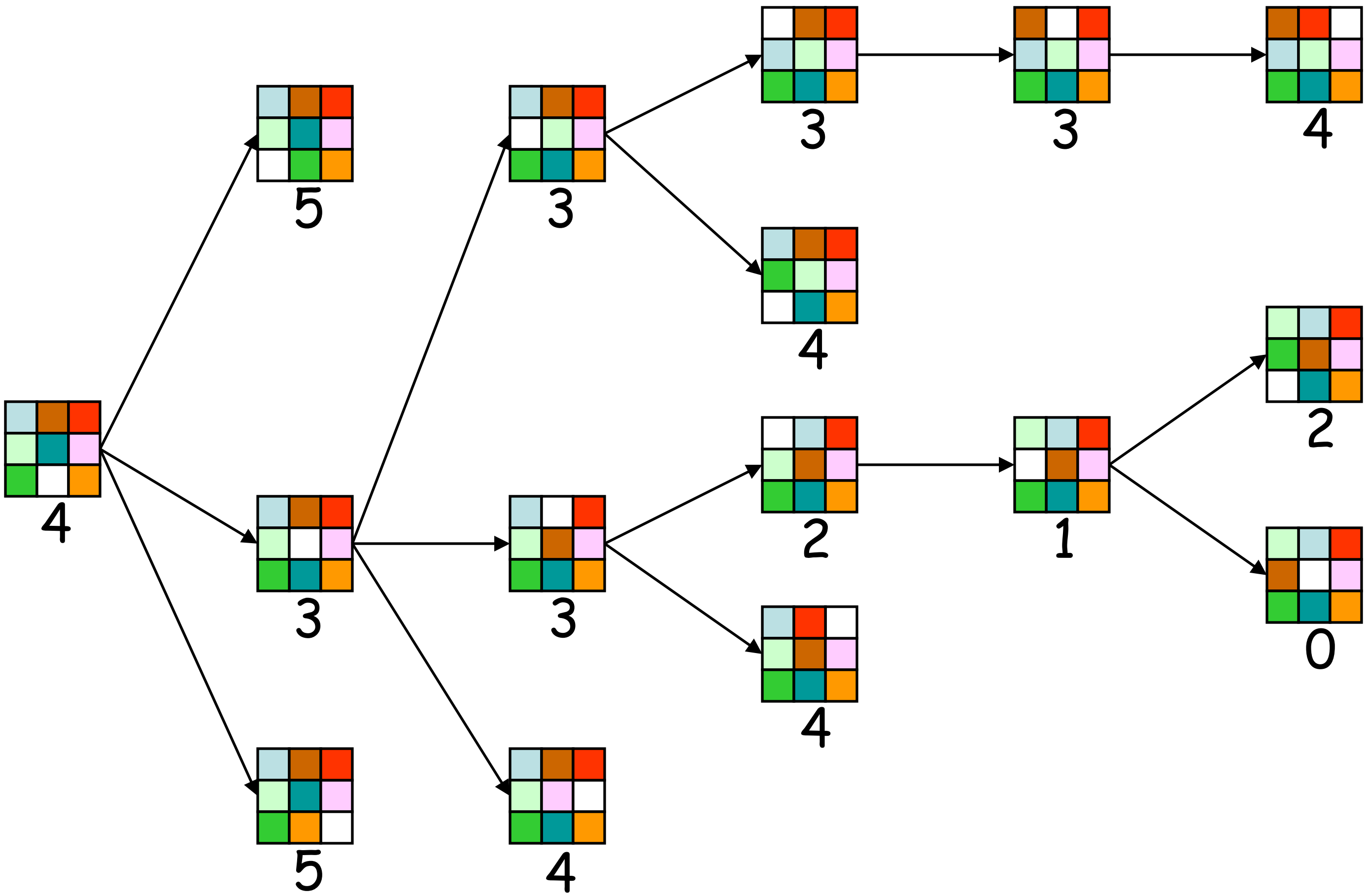
1	2	3
4	5	6
7	8	

Goal state

- $h_1(N)$ = number of misplaced numbered tiles = 6
- $h_2(N)$ = sum of the (Manhattan) distance of every numbered tile to its goal position = 2 + 3 + 0 + 1 + 3 + 0 + 3 + 1 = 13

8-Puzzle

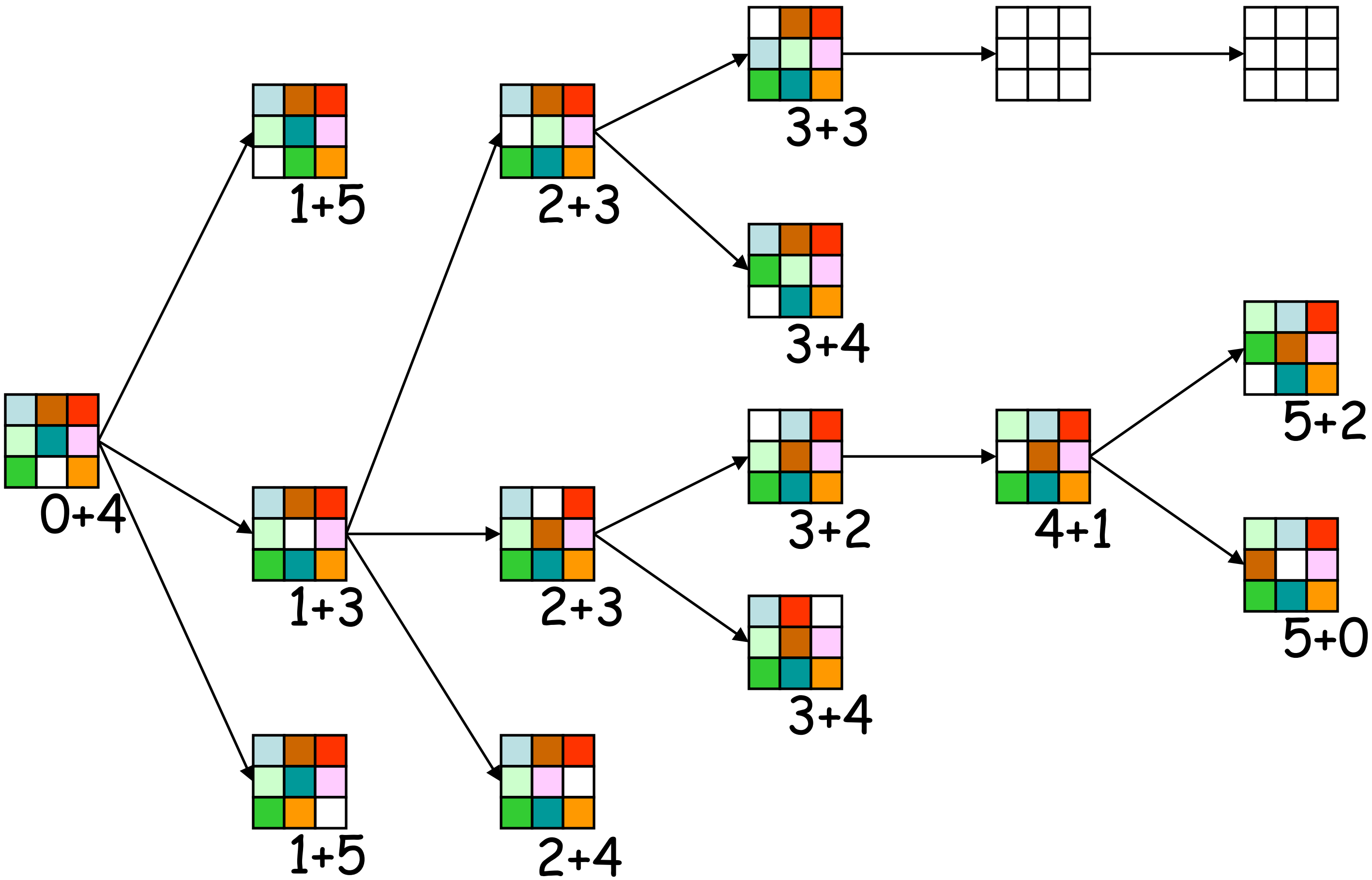
f(N) = h(N) = number of misplaced numbered tiles



The white tile is the empty tile

8-Puzzle

$f(N) = g(N) + h(N)$
with $h(N)$ = number of misplaced numbered tiles



Heuristic Function

Heuristic

- Use our domain knowledge about the problem to choose some (not all) successors of the current state
- To speed up the searching process
- Heuristic function takes up a state and return the assessment of that state
- Finding right function is not always easy

Heuristic Function and AI search

- Heuristic function estimates how close a state is to the goal state
- It is just an estimation of path cost from a state to goal state (not actual value)
- In state space search, heuristic function helps to reduce the number of nodes expanded during search

Problem relaxation

- Standard approach to create a heuristic is problem relaxation

- Add some new actions for the problem so that search is not required to find the solution cost in the relaxed problem

Problem relaxation with example

5		8
4	2	1
7	3	6

STATE(N)

1	2	3
4	5	6
7	8	

Goal state

- A problem with fewer restriction on action is called relaxed problem
- One way: pick up any misplaced tiles and place it in the appropriate position (tiles can move anywhere)
- Another way: relaxation on moving tiles along the x-axis and y-axis (tiles can move adjacent squares)

Problem relaxation with example

5		8
4	2	1
7	3	6

STATE(N)

1	2	3
4	5	6
7	8	

Goal state

- $h_1(N)$ = number of misplaced numbered tiles = 6
- $h_2(N)$ = sum of the (Manhattan) distance of every numbered tile to its goal position = 2 + 3 + 0 + 1 + 3 + 0 + 3 + 1 = 13

Cost of optimal solution to the relaxed problem is an admissible heuristic for the original problem (h_1 and h_2 both are admissible heuristic)

Admissible Heuristic

- It never overestimates the cost for reaching goal node from any node
- Let $h^*(N)$ be the cost of the optimal path from N to a goal node
- The heuristic function $h(N)$ is admissible if for all N:

$$0 \leq h(N) \leq h^*(N)$$

$h(N) < h^*(N)$ [underestimated]; $h(N) > h^*(N)$ [overestimated]

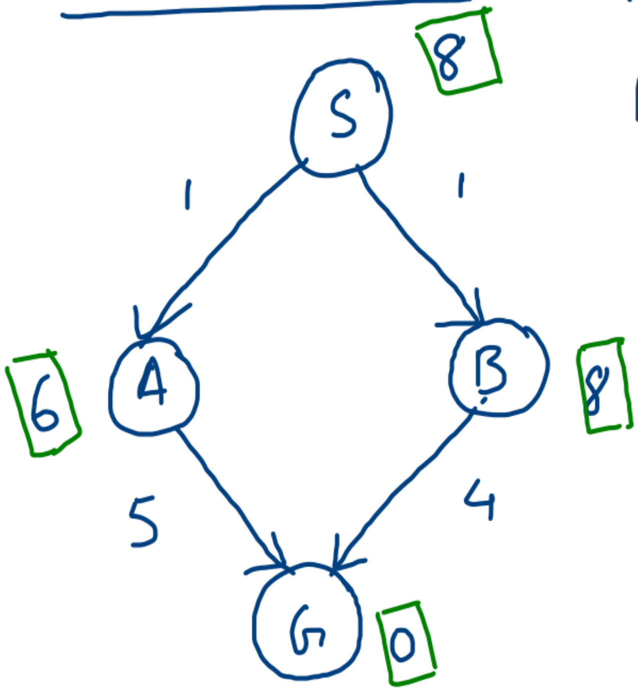
- An admissible heuristic function is always optimistic !

G is a goal node → $h(G) = 0$

Admissible vs. Non-admissible heuristic

- **Non-admissible heuristic:**
  - Pessimistic heuristic because they overestimate the cost
  - Breaking optimality by trapping good plan in fringe
- **Admissible heuristic:**
  - Optimistic heuristic they can only underestimate the cost
  - Can only slow down the search by assigning lower cost to a bad plan so that it will explored first; but it will find the optimal path gradually.

Inadmissible



$f(N) = g(N) + h(n)$

1. $OL = \{ S^{8} \}$

   $CL = \{ \}$

2. $OL = \{ A^{7}, B^{9} \}$

   $CL = \{ S^{8} \}$

3. $OL = \{ B^{9}, G^{6} \}$

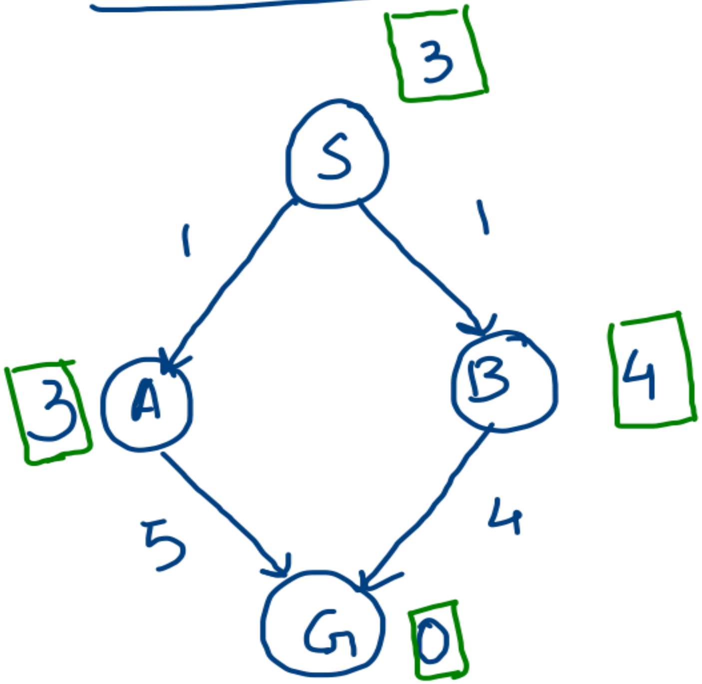
   $CL = \{ S^{8}, A^{7} \}$

4. $OL = \{ B^{9} \}$

   $CL = \{ S^{8}, A^{7}, G^{6} \}$

Break optimality

Admissible



1. $OL = \{S^3\}$

   $CL = \{\ \}$

2. $OL = \{A^4, B^5\}$

   $CL = \{S^3\}$

3. $OL = \{B^5, G^6\}$

   $CL = \{S^3, A^4\}$

4. $OL = \{G^{\cancel{6}\ 5}\}$

   $CL = \{S^3, A^4, B^5\}$

unnecessary expansion of A; but find optimal solution gradually

Dominance

For two admissible heuristics h_1 and h_2

If $h_2(n)>=h_1(n)$ for all n, then h_2 dominates h_1

h_2 is more informed that $h_{1.}$

h_2 is better for search.

[Better heuristic means we have to explore fewer node before find the solution]

Composite Heuristic

- maximum of two admissible heuristics is also admissible
- Suppose, we have designed two or more heuristics and unsure about that any of them dominates all others
- We can use maximum of them as composite heuristic

$$h(n) = \max\{h_1\,(n), \ldots, h_m(n)\ \}$$

Consistent Heuristic

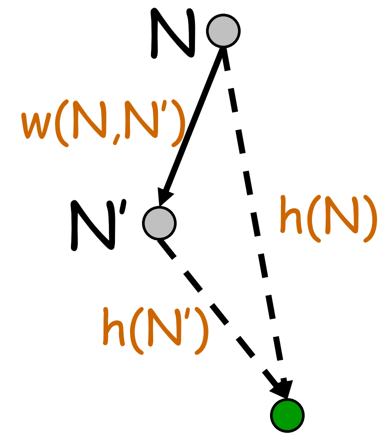
A heuristic h is consistent (or monotone) if

1) for each node N and each successor N' of N:

$$h(N) \leq w(N,N') + h(N')$$

2) for each goal node G:

$$h(G) = 0$$



(triangle inequality)

A consistent heuristic is also admissible; but opposite may or may not hold

Consistent Heuristic

- It will be stricter than the admissible one.
- one consequence, f-value never decreases along the path

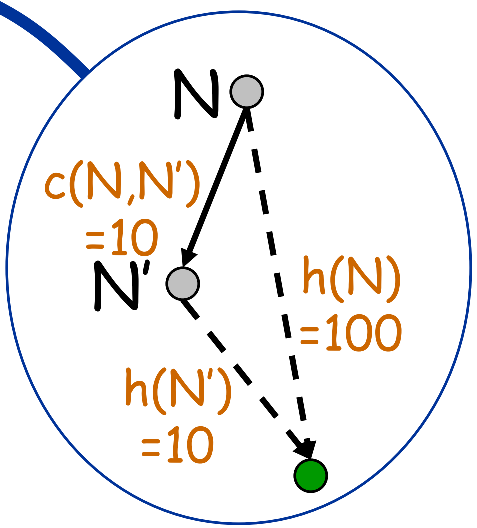
$$
\begin{aligned}
f(N') &= g(N')+h(N') \\
&= g(N)+w(N,N')+h(N') \\
&>= g(N)+h(N) \\
&= f(N)
\end{aligned}
$$

$$f(N') >= f(N)$$

→ Intuition: a consistent heuristic becomes more precise as we get deeper in the search tree

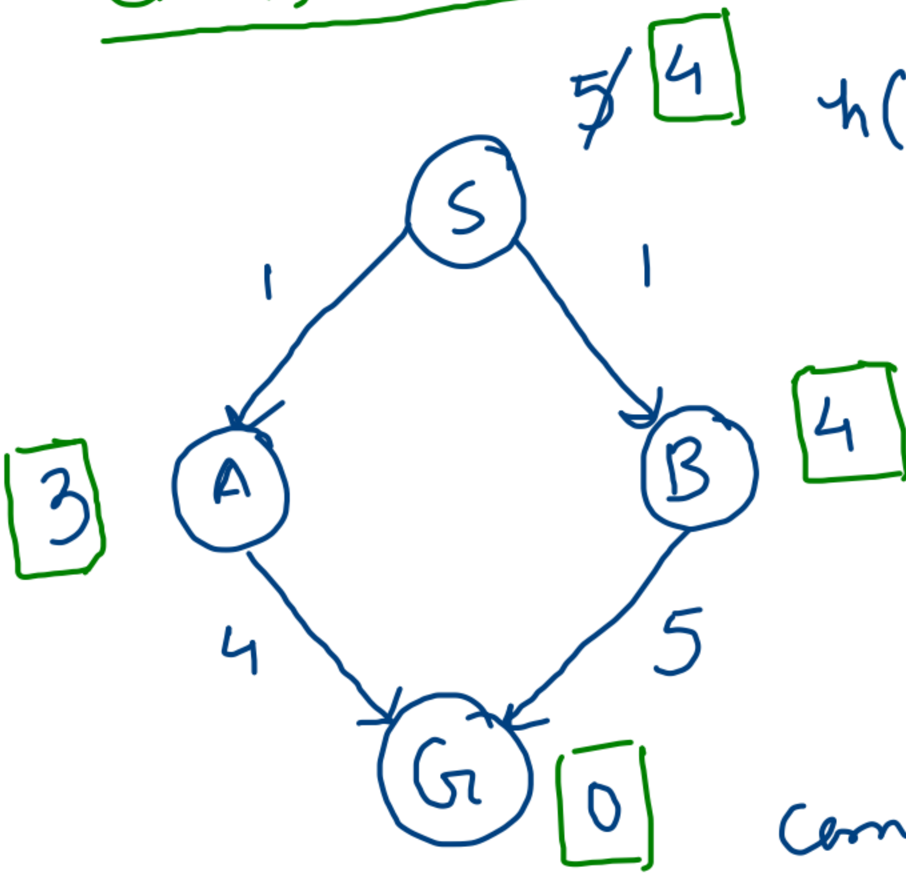
Consistency Violation

If h tells that N is 100 units from the goal, then moving from N along an arc costing 10 units should **not** lead to a node N' that h estimates to be 10 units away from the goal



(triangle inequality)

Consistent and admissible



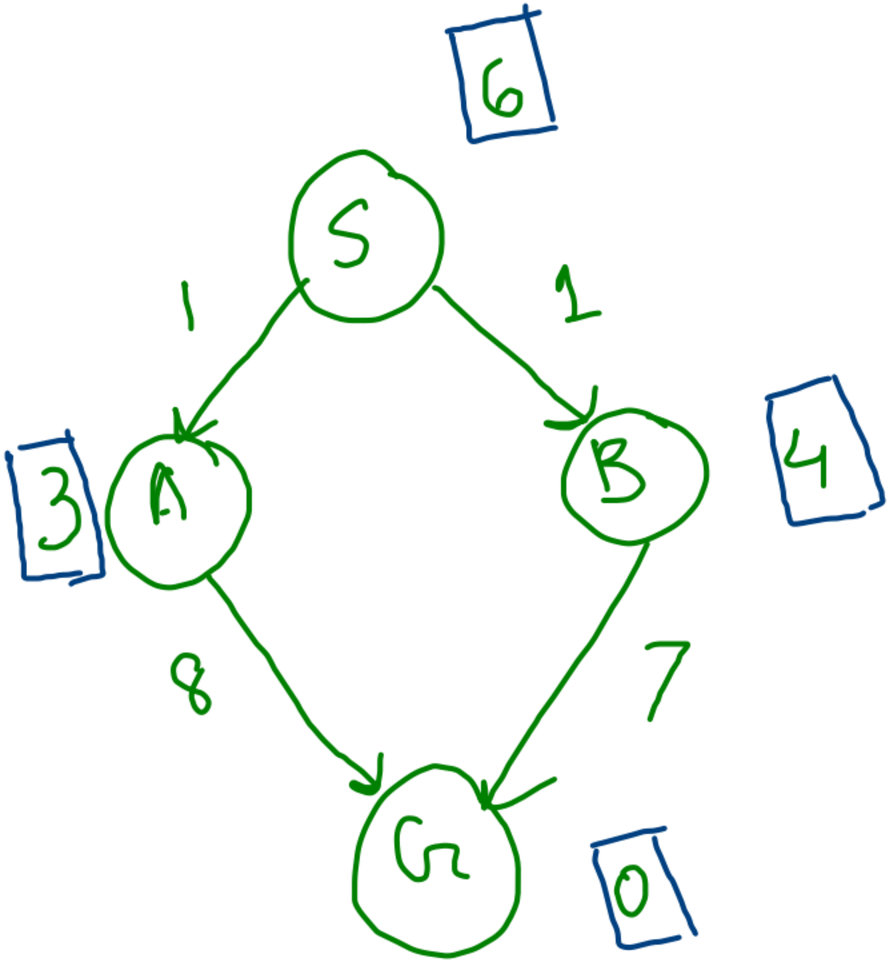
$$h(N) \leq w(N, N') + h(N')$$

$$f(S) = 4 + 0 = 4$$

$$f(B) = 4 + 1 = 5$$

$$f(G) = 0 + 6 = 6$$

Come closer to goal state; evaluation function increase

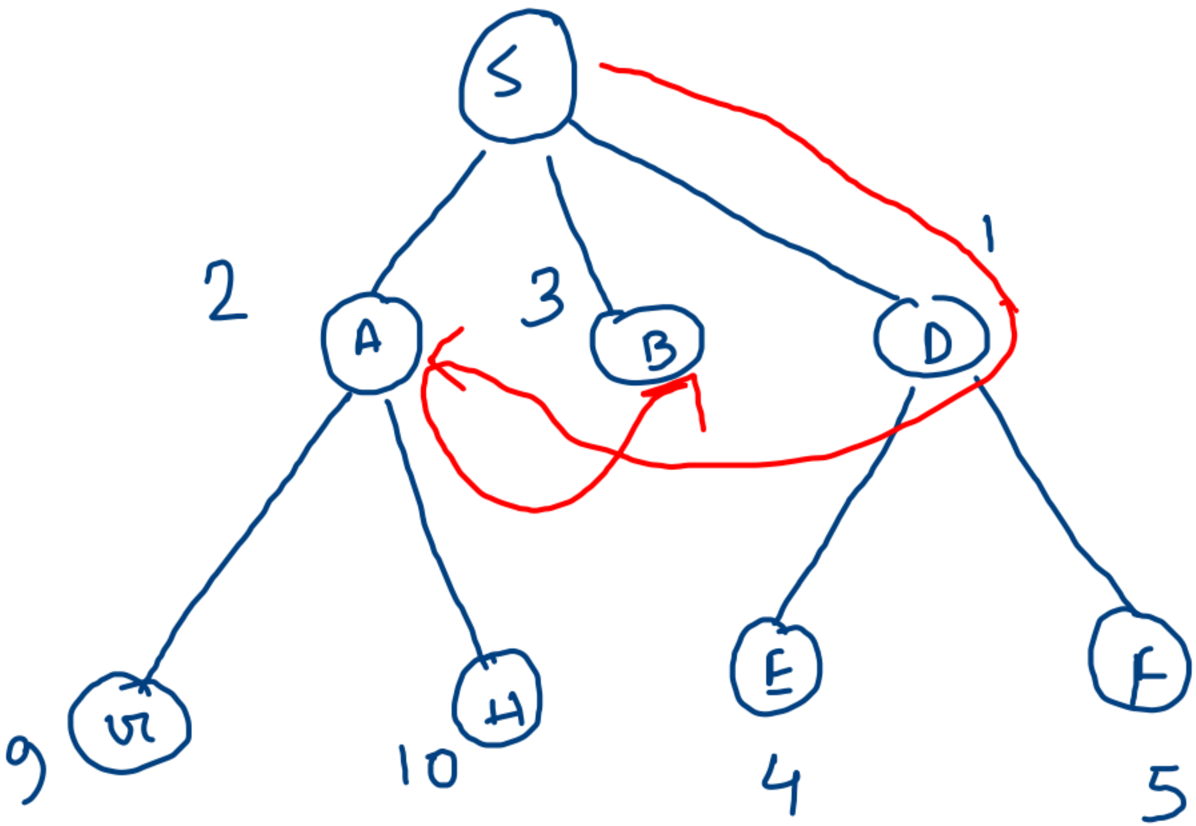


Admissible / not Consistent

Best First Search

- It is a way to combining the advantage of both breadth first search and depth first search
- DFS is good if it allows a solution to be found without all competing branches having to be expanded
- BFS is good as it does not trap into a dead end
- Combining by following a single path at a time but switch the path whenever a competing path looks more promising than the current one

Example of Best First Search



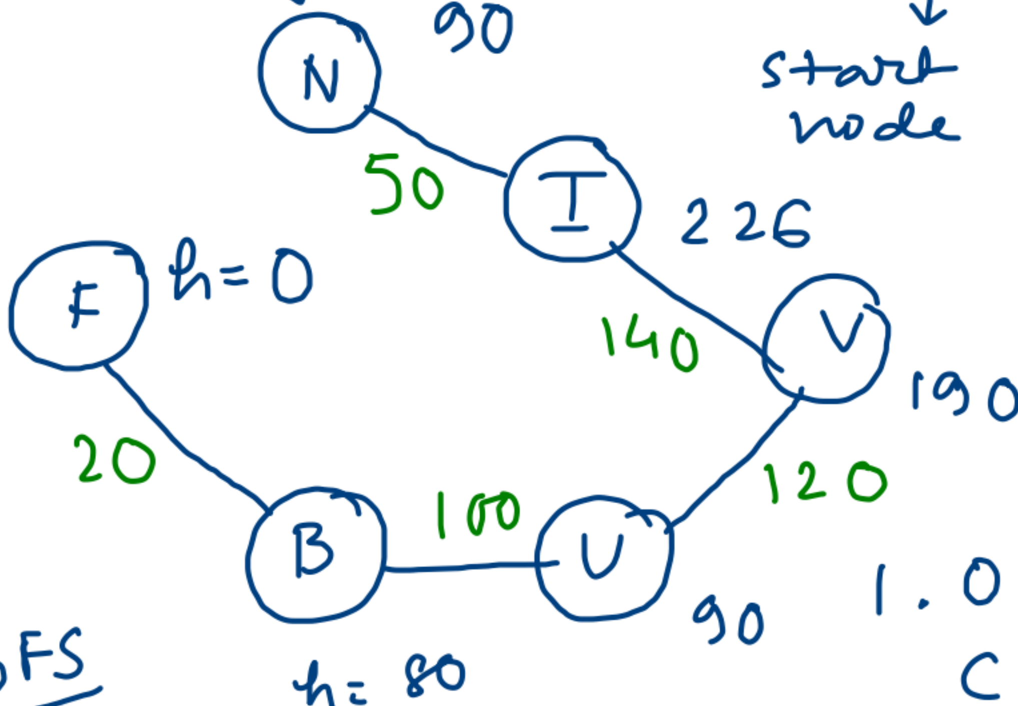
Greedy best-first search

- Evaluation function f(n)=h(n), estimated cost from n to goal
- In route finding problem, $h_{SLD}(n)$=straight line distance/air distance from a node n to destination city
- Heuristic can't be computed from the problem description itself
- In addition, the problem specific domain knowledge is required to understand the correlation between the actual road distance and straight line distance
- The strategy expands the node that appears to be closest to goal (completely heuristic dependant)
- It may lead to the dead end in case of route finding problem

Route finding problem from I to F

I → start node

F → goal node



Straight line distance between two cities (h

Apply Greedy BFS

1. OL={ I^{226} }
   CL={ }

2. OL={ N^{90}, V^{190} }
   CL= { I^{226} }

3. OL={ V^{190}, I^{226} }
   CL={ I^{226}, N^{90} }

Infinite loop (if allow revisit) otherwise dead end

h → estimated one

Evaluation

- If the state space is infinite, in general the search is not complete
- If the state space is finite and we do not discard nodes that revisit states, in general the search is not complete
- If the state space is finite and we discard nodes that revisit states, the search is complete, but in general is not optimal
- The worst case time and space complexity is $O(b^m)$, m is maximum depth of search space
- However, the complexity can be reduced substantially with good heuristic.

A* Search
(most popular algorithm in AI)

1) $f(N) = g(N) + h(N)$, where:
   - $g(N)$ = cost of best path found so far to N
   - $h(N)$ = heuristic function

2) for all arcs: $w(N,N') \geq \varepsilon > 0$

➔ Best-first search is then called *A* search*

Search algorithm (A*)

1. **Initialize:** Set OPEN = {s}, CLOSED = { } Set g(s) = 0 and f(s)=h(s)
2. **Fail:** If OPEN = { }, Terminate & fail
3. **Select:** Select the **minimum cost state, n,** from OPEN and save n in CLOSED
4. **Terminate:** If $n \in G$, terminate with success

Search algorithm (A*)

5. **Expand:**

Generate the successors of n using successor function.

For each successor, m:

If $m \notin [OPEN \cup CLOSED]$

Set $g(m) = g(n) + w(n,m)$

Set $f(m)=g(m)+h(m)$

and insert m in OPEN

If $m \in [OPEN \cup CLOSED]$

Set $g(m) = \min \{g(m), g(n) + w(n,m)\}$

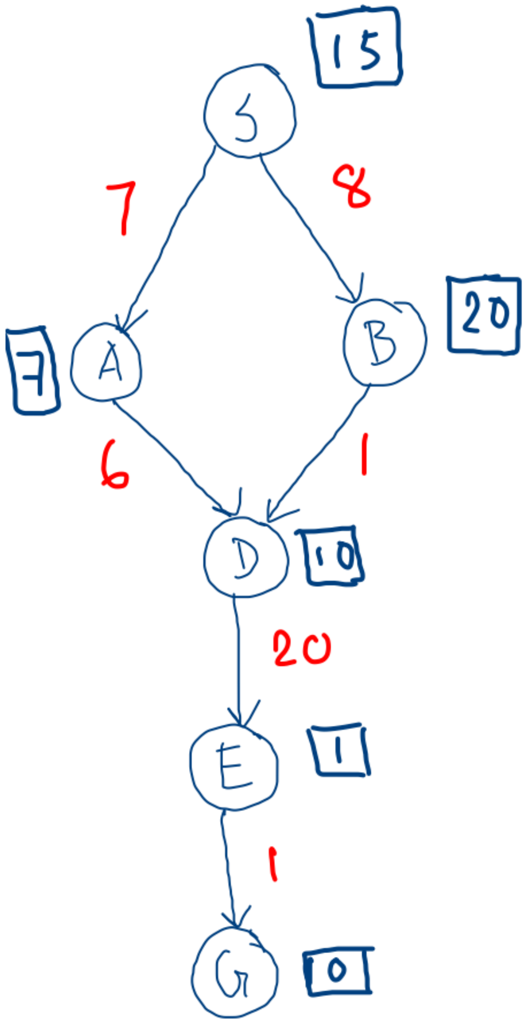
Set $f(m)=g(m)+h(m)$

If **$f(m)$ has decreased** and

$m \in CLOSED$, move it to OPEN

Evaluation

- A* search is not optimal if the heuristic is overestimated (break optimality)
- A* search is optimal with admissible heuristic if node reopening is allowed.
- Node reopening leads to the unnecessary node expansion (re-visit)
- A* search is optimal with consistent heuristic; In this case, node reopening is not needed (always expand node with optimal path)



Admissible heuristic

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

1. $OL = \{S^{15}\}$ $CL = \{\ \}$
2. $OL = \{A^{14}, B^{28}\}$ → due to accurate heuristic, good plan trapped

   $CL = \{S^{15}\}$
3. $OL = \{B^{28}, D^{23}\}$

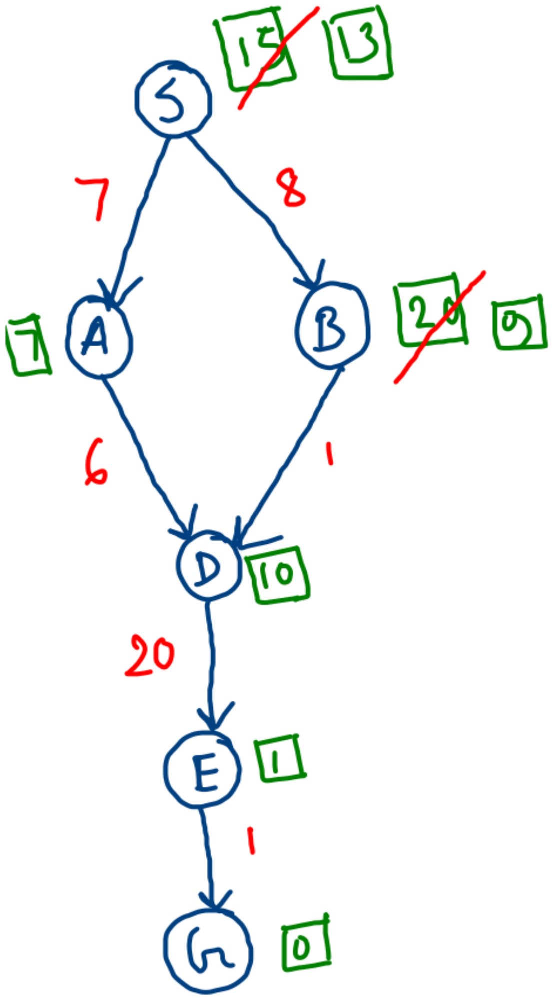
   $CL = \{S^{15}, A^{14}\}$ → expand node in bad plan
4. $OL = \{B^{28}, E^{34}\}$

   $CL = \{S^{15}, A^{14}, D^{23}\}$ → move from closed to open
5. $OL = \{E^{34}, D^{19}\}$ → now we have a good plan

   $CL = \{S^{15}, A^{14}, \cancel{D^{23}}, B^{28}\}$ discard previous path
6. $OL = \{E^{\cancel{34}\ 30}\}$

   $CL = \{S^{15}, A^{14}, B^{28}, D^{19}\}$
7. $OL = \{G^{30}\}$ → we can reach optimal goal in next step

   $CL = \{S^{15}, A^{14}, B^{28}, D^{19}, E^{30}\}$



Consistent heuristic

$$h(n) \le w(n, n') + h(n')$$

1. $OL = \{S^{13}\}$, $CL = \{\ \}$
2. $OL = \{A^{14}, B^{17}\}$
   $CL = \{S^{13}\}$

no node-reopening is needed

3. $OL = \{B^{17}, D^{23}\}$
   $CL = \{S^{13}, A^{14}\}$
4. $OL = \{D^{\cancel{23}\ 19}\}$
   $CL = \{S^{13}, A^{14}, B^{17}\}$ good plan
5. $OL = \{E^{30}\}$
   $CL = \{S^{13}, A^{14}, B^{17}, D^{19}\}$ goal in next step
6. $OL = \{G^{30}\}$
   $CL = \{S^{13}, A^{14}, B^{17}, D^{19}, E^{30}\}$

Completeness

- A* expands all nodes with f(n)<C* (C* is the cost of optimal solution path)

- It may expand some nodes with f(n)=C*

- A* search is complete if there is finitely many nodes with cost less than or equal to C* (additionally step cost exceeds some finite value and b is finite)

Complexity

- Time/space complexity is exponential
- With good heuristic, it can reduce significantly
- Main problem is storage; need to store all generated nodes

A* Search (optimally efficient)

- A* search is optimally efficient for any given consistent heuristic
- No other optimal algorithm is guaranteed to expand fewer nodes than A*
- Any algorithm that does not expand all node with $f(n)<C^*$ runs the risk of missing the optimal solution.

Conclusion

- Design good heuristic for A* search
- Use different variants of A* (if possible some bound on memory requirement)

We can think about a solution (not necessarily optimal) for large-scale AI problem.

Move to the local search algorithm

Local Search

- Light-memory search method
- No search tree; only the current state is represented!
- Only applicable to problems where the path is irrelevant (e.g., 8-queen)
- Many similarities with optimization techniques

Search problems

Blind search

Heuristic search:
best-first (A* and Greedy BF)

Construction of heuristics

Variants of A*

Local search