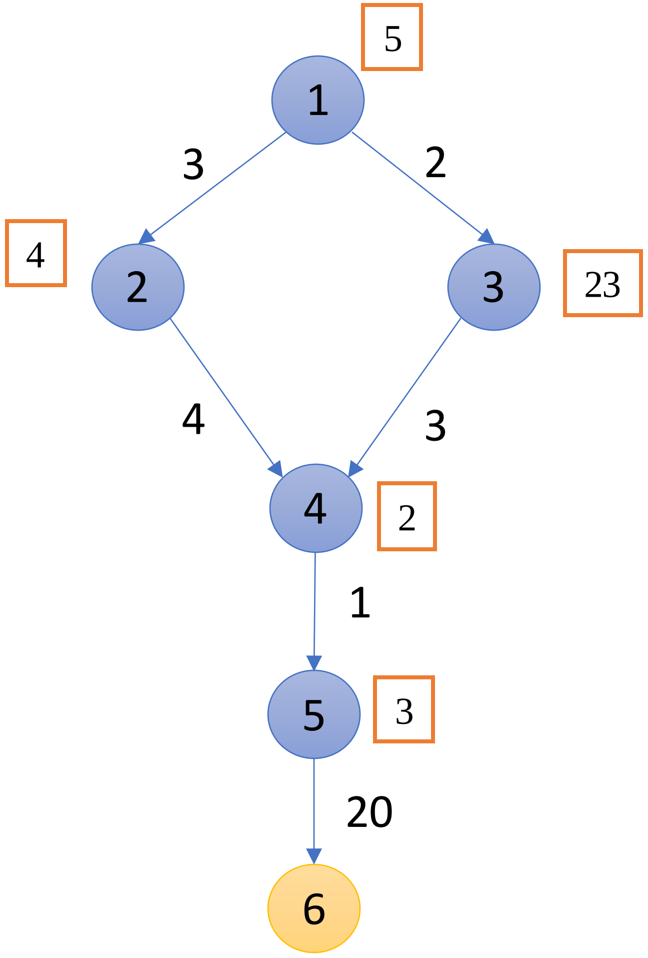
A* Analysis

20/01/2025

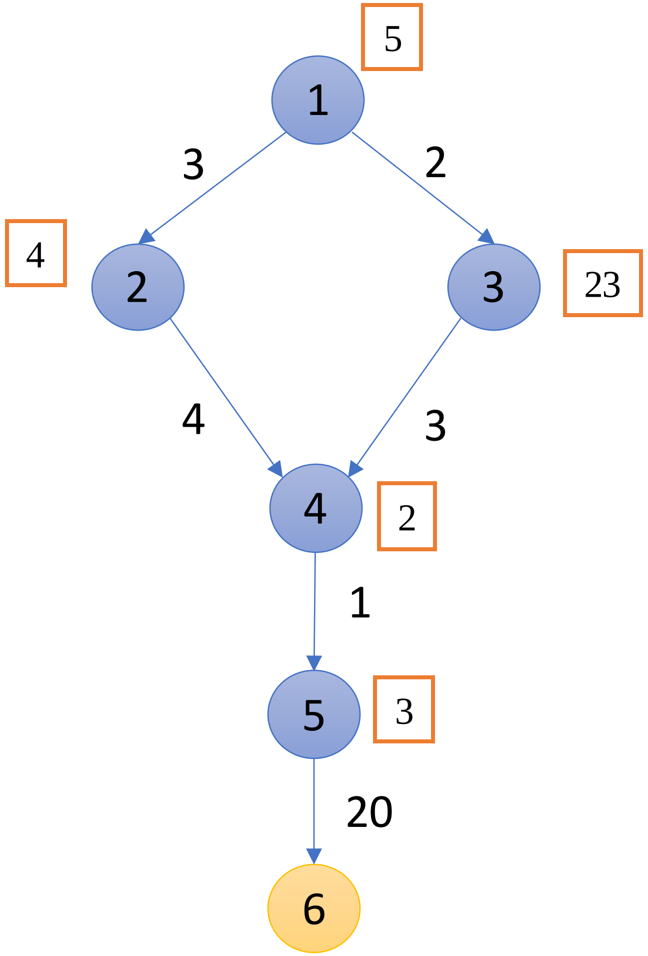
Koustav Rudra

Algorithm A*



OPEN SET	SELECT	GOAL	EXPANDED	CLOSED
[1(5)]	1(5)	N	[2(7),3(25)]	[1(5)]
[2(7),3(25)]	2(7)	N	[3(25),4(9)]	[1(5),2(7)]
[3(25),4(9)]	4(9)	N	[3(25),5(11)]	[1(5),2(7),4(9)]
[3(25),5(11)]	5(11)	N	[3(25),6(28)]	[1(5),2(7),4(9),5(11)]
[3(25),6(28)]	3(25)	N	[6(28),4(7)]	[1(5),2(7),~~4(9)~~,5(11),3(25)]

Algorithm A*



OPEN SET	SELECT	GOAL	EXPANDED	CLOSED
[3(25),6(28)]	3(25)	N	[6(28),4(7)]	[1(5),2(7),~~4(9)~~,5(11),3(25)]
[6(28),4(7)]	4(7)	N	[6(28),5(9)]	[1(5),2(7),~~4(9)~~,~~5(11)~~,3(25),4(7)]
[6(28),5(9)]	5(9)	N	[6(26)]	[1(5),2(7),~~4(9)~~,~~5(11)~~,3(25),4(7),5(9)]
[6(26)]	6(26)	Y		

A* Analysis

20/01/2025

Koustav Rudra

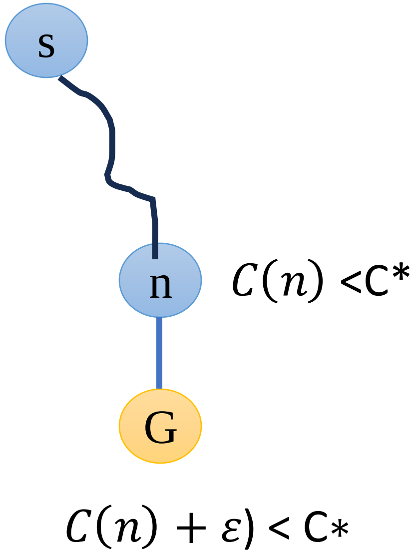
Algorithm A*: Benefit

- Reduces number of expanded nodes
- Performs the lookahead and tells us promising paths
- What about optimality?

Uniform Cost Search

- Claim: If C(n)<C* (optimal cost) then n must be expanded
- Let algorithm A does not expand n

- For the class of algorithms without any heuristics
  - All states that have cost $< C^*$ will have to be expanded
  - Always expands the minimum cost node in your frontier
  - When we find the goal
    - All the states that we have in the frontier have cost higher than the goal state



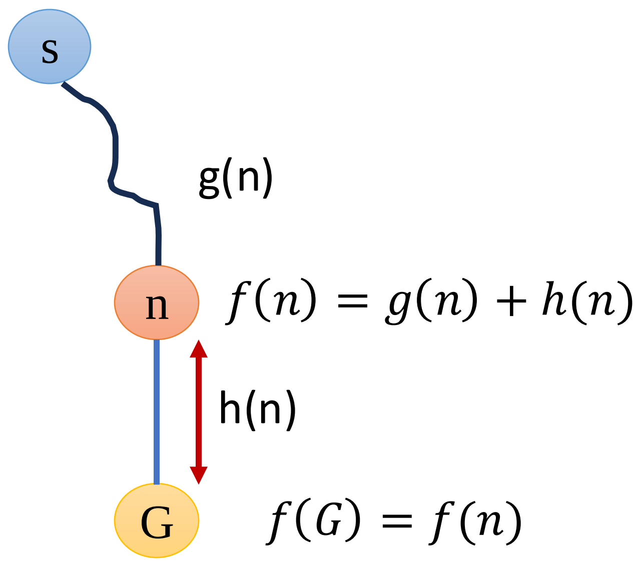
Algorithm A*: Benefit

- Claim: $f(n) < C^*$ then n must be expanded

- The heuristic function underestimates
  - $h(n) \leq f^*(n)$
    - Cost of reaching goal from n
  - All costs are +ve

- If we do not expand n, we can't find the goal

- If we have a state whose cost is less than C*
  - Then every algorithm which guarantees finding optimal solution have to expand it



Algorithm A*

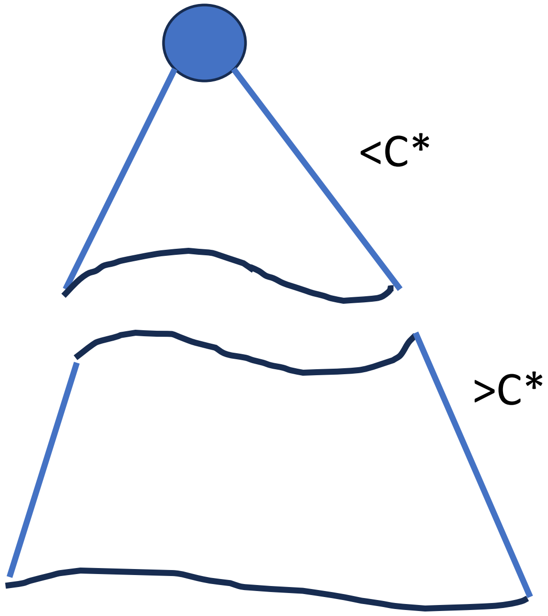
- Evaluate the INITIAL state
  - If it is GOAL return it
  - Else CURRENT← INITIAL

- Loop until the solution is found or no new operators could be applied to CURRENT:
  - Select an operator that has not been applied to the current state [CURRENT] and apply it to produce new state [NEW]

  - Evaluate NEW:
    - If it is GOAL return it
    - Else If NEW > CURRENT, CURRENT← NEW
    - Else go to Loop

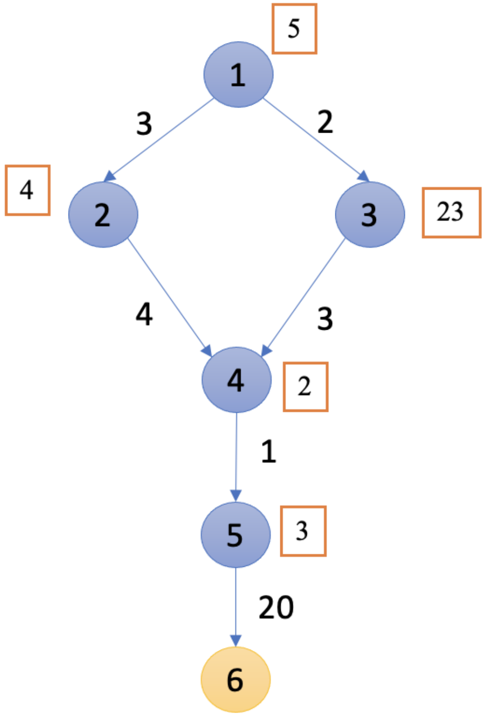
A*: Result

- What is an admissible heuristic?
  - If it always underestimates
  - We always have $h(n) \leq f^*(n)$, where $f^*(n)$ denotes minimum distance to a goal state from n
- For finite state spaces, A* will always terminate



A*: Result

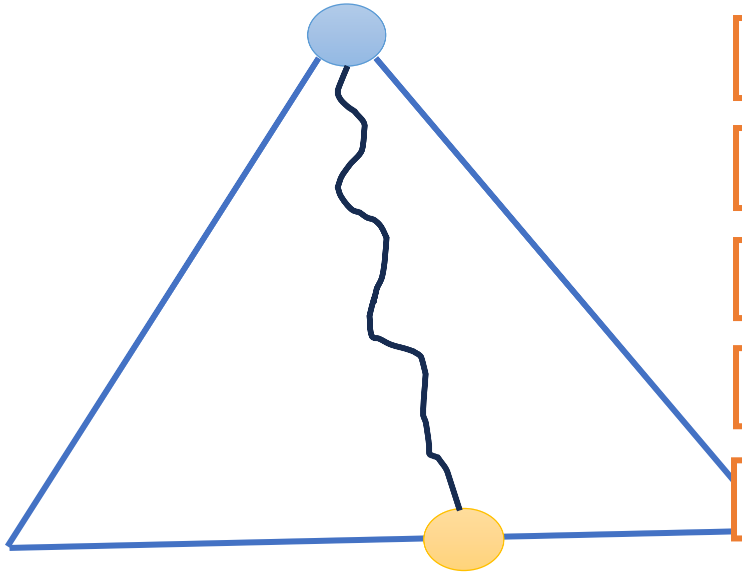
- At any time before A* terminates, there exists in OPEN a state **n**
  - That is on an optimal path from s to a goal state, with
  - $f(n) \leq f^*(s)$



OPEN SET	SELECT	GOAL	EXPANDED	CLOSED
[3(25),5(11)]	5(11)	N	[3(25),6(28)]	[1(5),2(7),4(9),5(11)]
[3(25),6(28)]	3(25)	N	[6(28),4(7)]	[1(5),2(7),~~4(9)~~,5(11),3(25)]
[6(28),4(7)]	4(7)	N	[6(28),5(9)]	[1(5),2(7),~~4(9)~~,~~5(11)~~,3(25),4(7)]
[6(28),5(9)]	5(9)	N	[6(26)]	[1(5),2(7),~~4(9)~~,~~5(11)~~,3(25),4(7),5(9)]
[6(26)]	6(26)	Y		

A*: Result

- At any time before A* terminates, there exists in OPEN a state **n**
  - That is on an optimal path from s to a goal state, with
  - $f(n) \leq f^*(s)$



OPEN SET	SELECT	GOAL	EXPANDED	CLOSED
[3(25),5(11)]	5(11)	N	[3(25),6(28)]	[1(5),2(7),4(9),5(11)]
[3(25),6(28)]	3(25)	N	[6(28),4(7)]	[1(5),2(7),~~4(9)~~,5(11),3(25)]
[6(28),4(7)]	4(7)	N	[6(28),5(9)]	[1(5),2(7),~~4(9)~~,~~5(11)~~,3(25),4(7)]
[6(28),5(9)]	5(9)	N	[6(26)]	[1(5),2(7),~~4(9)~~,~~5(11)~~,3(25),4(7),5(9)]
[6(26)]	6(26)	Y		

A*: Result

- If there is a path from s to a goal state, A* terminates
  - Even if the state space is infinite
  - Need good estimate about heuristic

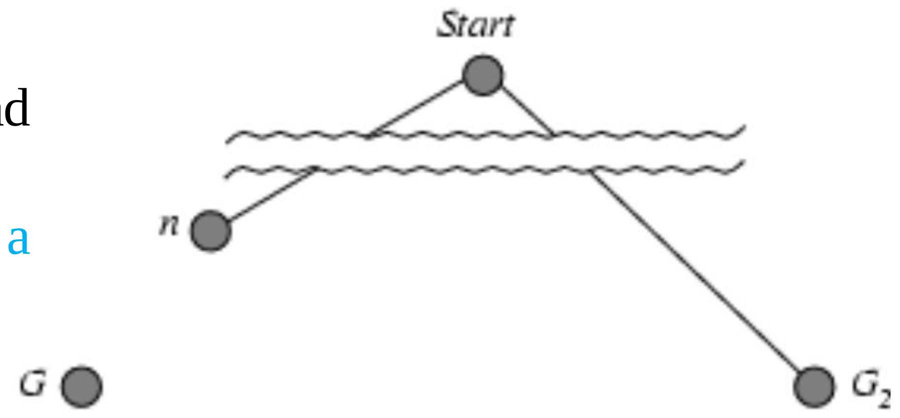
- What is the worst case scenario?
  - Heuristic is difficult to estimate

- Approximation algorithm

A*: Result

- Algorithm A* is admissible
  - If there is a path from s to a goal state,
  - A* terminates by finding an optimal path

A* Optimality



- Suppose some suboptimal goal path G2 has been generated and is in OPEN
- Let n be an unexpanded node in OPEN such that n is on a shortest path to an optimal goal G

$h(G_2) = h(G) = 0$

$f(G_2) = g(G_2)$ $\quad$ $\mathrm{f}(G) = g(G)$

$g(G_2) > g(G)$ $\quad$ G_2 is suboptimal

$f(G_2) > f(G)$

$h(n) \leq h^*(n)$ $\quad$ h is admissible, h^* is minimal distance

$g(n) + h(n) \leq g(n) + h^*(n)$

$\mathrm{f}(n) < f^*(G)$ $\quad$ $f(G_2) > \mathrm{f}(n)$ $\quad$ A* will never select G_2 for expansion

A*: Result

- Algorithm A* is admissible
  - If there is a path from s to a goal state,
  - A* terminates by finding an optimal path

- If A1 and A2 are two versions of A* such that A2 is more informed than A1
  - A1 expands at least as many states as does A2

  - $\forall_n h_2(n) > h_1(n)$
  - h_2 is tighter than h_1
  - Claim: $f(n) < C^*$ then n must be expanded

A*: Result

- We have two good heuristic functions h1 and h2 but do not know which one is more informed
  - Which one to use?
  - $\forall_n \max\{h_1(n), h_2(n)\}$

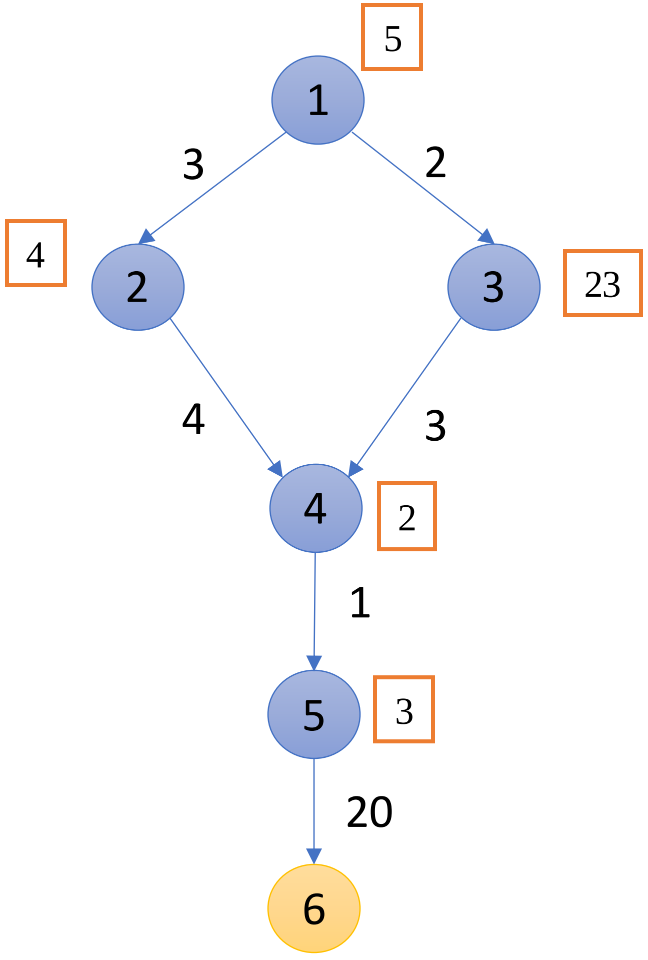
- How much effort should we put in computing heuristics?
  - If state space is complex we may invest time

A* Analysis

20/01/2025

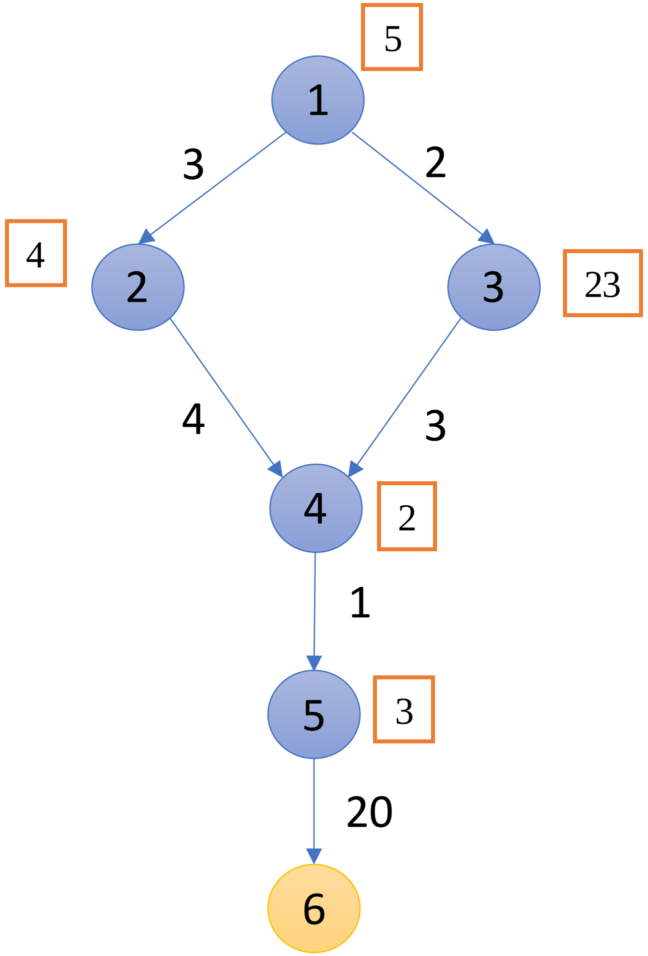
Koustav Rudra

Algorithm A*



OPEN SET	SELECT	GOAL	EXPANDED	CLOSED
[1(5)]	1(5)	N	[2(7),3(25)]	[1(5)]
[2(7),3(25)]	2(7)	N	[3(25),4(9)]	[1(5),2(7)]
[3(25),4(9)]	4(9)	N	[3(25),5(11)]	[1(5),2(7),4(9)]
[3(25),5(11)]	5(11)	N	[3(25),6(28)]	[1(5),2(7),4(9),5(11)]
[3(25),6(28)]	3(25)	N	[6(28),4(7)]	[1(5),2(7),~~4(9)~~,5(11),3(25)]

Algorithm A*

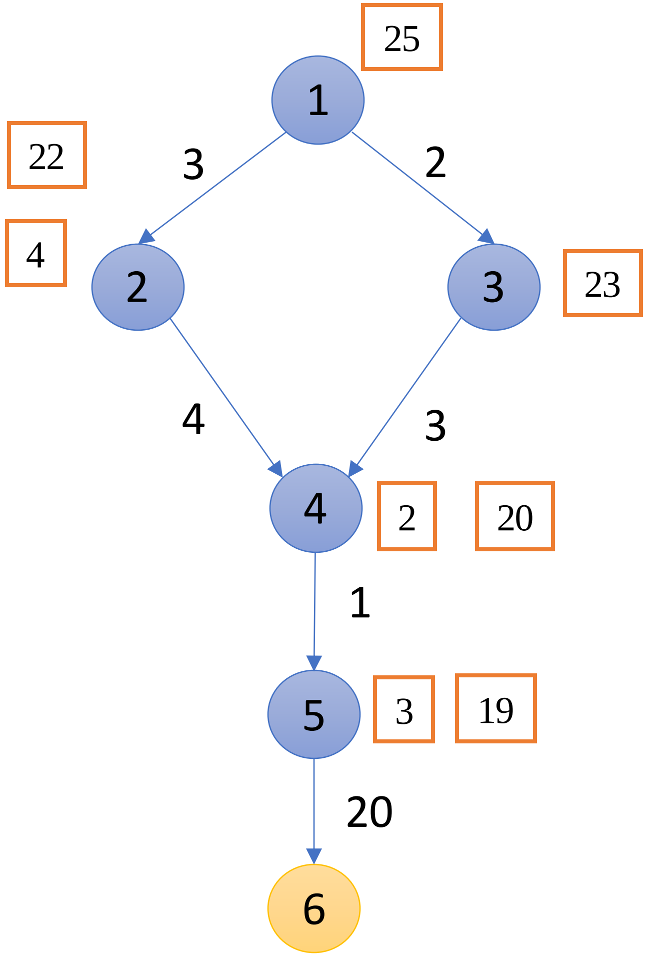


OPEN SET	SELECT	GOAL	EXPANDED	CLOSED
[3(25),6(28)]	3(25)	N	[6(28),4(7)]	[1(5),2(7),~~4(9)~~,5(11),3(25)]
[6(28),4(7)]	4(7)	N	[6(28),5(9)]	[1(5),2(7),~~4(9)~~,~~5(11)~~,3(25),4(7)]
[6(28),5(9)]	5(9)	N	[6(26)]	[1(5),2(7),~~4(9)~~,~~5(11)~~,3(25),4(7),5(9)]
[6(26)]	6(26)	Y		

Monotone Heuristics

- An admissible heuristic function, h(), is monotonic if for every successor m of n:
  - $h(n) - h(m) \leq C(n, m)$

Algorithm A*



OPEN SET	SELECT	GOAL	EXPANDED	CLOSED
[1(25)]	1(25)	N	[3(25),2(7)] [3(25),2(25)]	[1(25)]
[3(25),2(25)]	3(25)	N	[2(25),4(7)] [2(25),4(25)]	[1(25),3(25)]
[2(25),4(25)]	2(25)	N	[4(25)]	[1(25),3(25),2(25)]
[4(25)]	4(25)	N	[5(9)] [5(25)]	[1(25),3(25),2(25),4(25)]
[5(25)]	5(25)	N	[6(26)]	[1(25),3(25),2(25),4(25), 5(25)]
[6(25)]	6(25)	Y		

Monotone Heuristics

- An admissible heuristic function, h(), is monotonic if for every successor m of n:
  - $h(n) - h(m) \leq C(n, m)$

- If the monotone heuristic is satisfied, then A* has already found an optimal path to the state it selects for expansion

  - f(m) = g(m) + h(m)
  - f(m) = g(n) + C(n,m) + h(m)
  - $f(m) \geq g(n) + h(n)$
  - $f(m) \geq f(n)$

f(n) is non-decreasing along any path

If *h(n)* is consistent/monotone, A* using GRAPH-SEARCH is optimal

Monotone Heuristics

- An admissible heuristic function, h(), is monotonic if for every successor m of n:
  - $h(n) - h(m) \leq C(n, m)$

- If the monotone heuristic is satisfied, then A* has already found an optimal path to the state it selects for expansion

- How to convert a non-monotonic heuristic to monotonic one?

Pathmax

- Converts a non-monotonic heuristic to a monotonic one

  - During generation of the successor, m, of n we set:
    - $h'(m) = \max\{h(m), h(n) - C(n,m)\}$

    - Use $h'(m)$ as heuristic at m

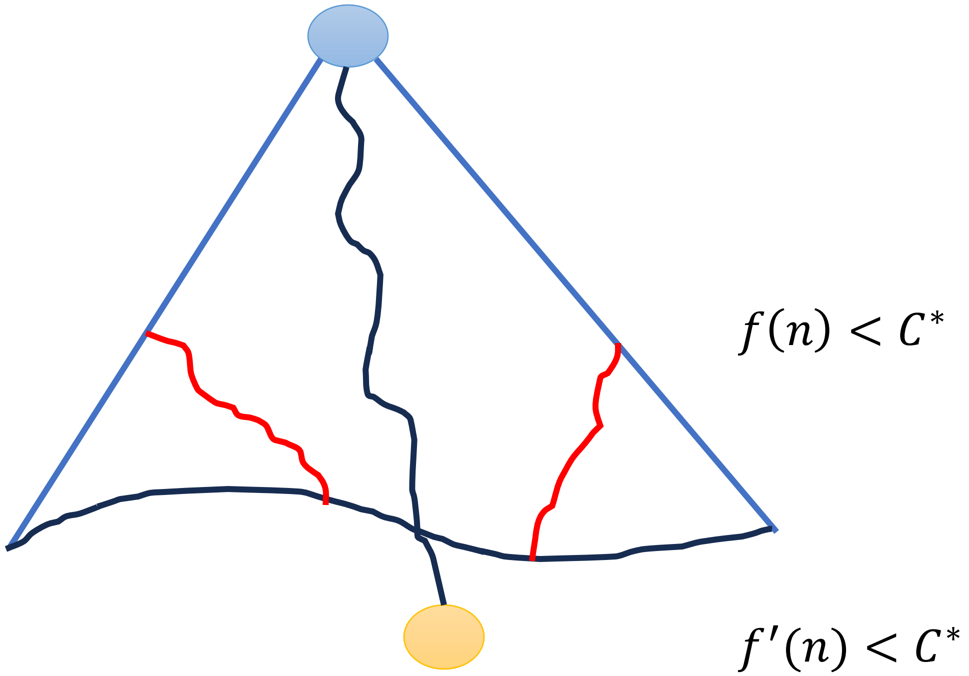
A*: Result

- Heuristic Function is weak
  - Lots of states having cost < C*
  - Need to explore lots of states
  - Less pruning

- Heuristic function is zero
  - Uniform cost search

- Heuristic function is overestimating
  - More pruning
  - Less number of states to explore
  - Does that always help?

A*: Result



When to go for overestimating?

Are we satisfied with sub-optimal solution?

In some cases we may get 10-50 times faster solution

Inadmissible Heuristic

- Advantages
  - In many cases, inadmissible heuristics can cause better pruning, and
  - Significantly reduce the search time

- Drawbacks
  - A* may terminate with a suboptimal solution

Algorithm A*: Result

- Is A* good?
  - Uses too much memory

- Why should we concern about the shift of nodes from CLOSED to OPEN?
  - We want to expand a node only once
  - Derive a set of states should be expanded by any admissible algorithm
  - Any optimal algorithm has to expand the nodes $f(n)<C^*$

Algorithm A*: Result

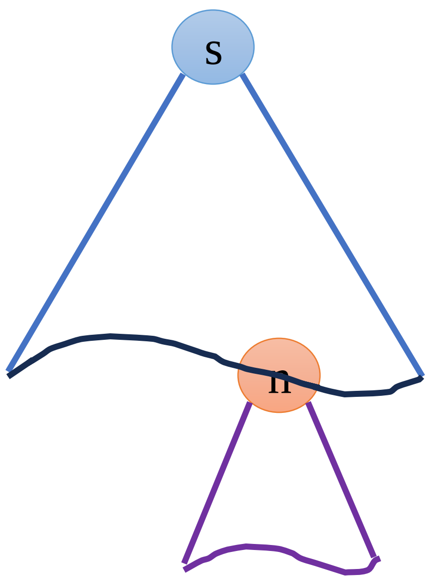
- Complexity would be linear in terms of expanded nodes
  - If we can ensure every node is expanded only once

  - If S is the set of nodes which must be visited by any admissible algorithm
    - Then the complexity is linear in S
    - $S = \{n | f(n) < C^*\}$

  - Any algorithm which is admissible or any algorithm which guarantees to give us the optimal solution
    - Will have to expand this set of states

  - Algorithm that is linear in the size of S is asymptotically optimal
    - Any algorithm that is admissible and guarantees optimal solution will have to explore |S| nodes

Iterative Deepening A* (IDA*)



- Set C = f(s)

- Perform DFBB with cut-off C
  - Expand a state, n, only if its f-value is less than or equal to C
  - If a goal is selected for expansion then return C and terminate

- Update C to the minimum f-value which exceeded C among states which were examined

- Go to Step 2

Iterative Deepening A* (IDA*)

- In the worst case, only one new state is expanded in each iteration
  - If A* expands N states, then IDA* can expand:
    - $1 + 2 + 3 + \cdots + N = O(N^2)$

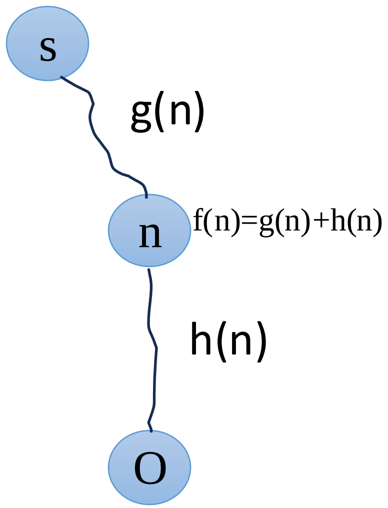
- Space Complexity:
  - Linear

- IDA* is asymptotically optimal

Variations

BEST-FIRST Tree Search

- Initialize: Set OPEN={s}, CLOSED = {}, f(s) = h(s)
- Fail:
  - If OPEN={}, Terminate with failure
- Select: Select the minimum cost state, n, from OPEN and save in CLOSED
- Terminate:
  - If n∈G, terminate with success

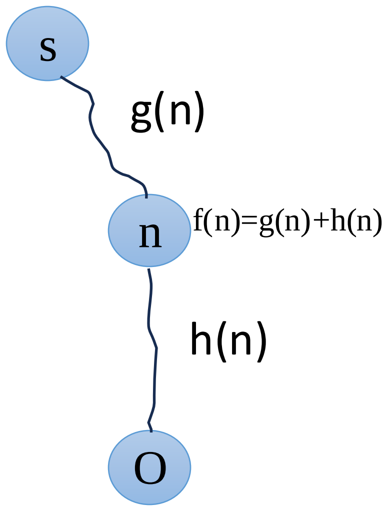


BEST-FIRST Tree Search

BEST-FIRST Tree Search with pruning

- Expand:
  - For each successor, m, of n:
    - If m∉[OPEN∪CLOSED]
      - Set $f(m) = h(m)$
      - Insert m in OPEN
- Loop:
  - Go to step 2

BEST-FIRST Tree Search [pruning]

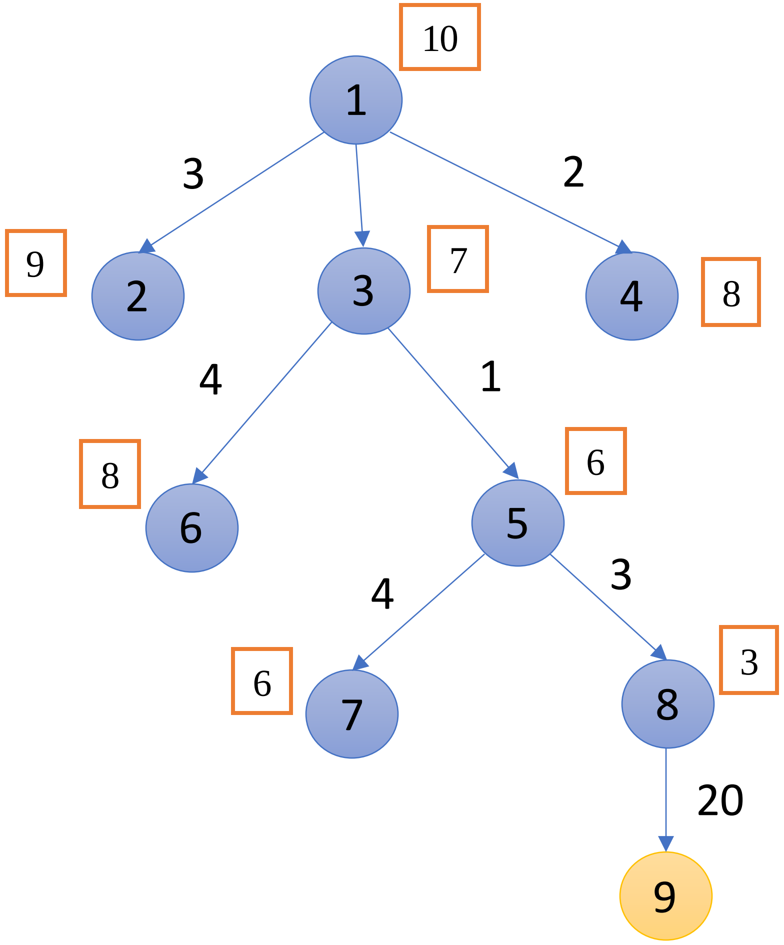


- Initialize: Set OPEN={s}, CLOSED = {}, f(s) = h(s), CB
- Fail:
  - If OPEN={}, Terminate with failure
- Select: Select the minimum cost state, n, from OPEN and save in CLOSED
- Terminate:
  - If n∈G and f(n) < CB, CB = f(n), Go to Step 2
  - Else terminate

BEST-FIRST Tree Search [pruning]

- Expand:
  - If f(n) $\leq$ CB
    - For each successor, m, of n:
      - If m$\notin$[OPEN$\cup$CLOSED]
        - Set $f(m) = h(m)$
        - Insert m in OPEN
- Loop:
  - Go to step 2

BEST-FIRST Tree Search



OPEN SET	SELECT	GOAL	EXPANDED	CLOSED
[1(10)]	1(10)	N	[2(9),3(7),4(8)]	[1(10)]
[2(9),3(7),4(8)]	3(7)	N	[2(9),4(8),6(8),5(6)]	[1(10),3(7)]
[2(9),4(8),6(8),5(6)]	5(6)	N	[2(9),4(8),6(8),7(6),8(3)]	[1(10),3(7),5(6)]
[2(9),4(8),6(8),7(6),8(3)]	8(3)	N	[2(9),4(8),6(8),7(6),6(0)]	[1(10),3(7),5(6),8(3)]
[2(9),4(8),6(8),7(6),9(0)]	9(0)	Y		

Thank You