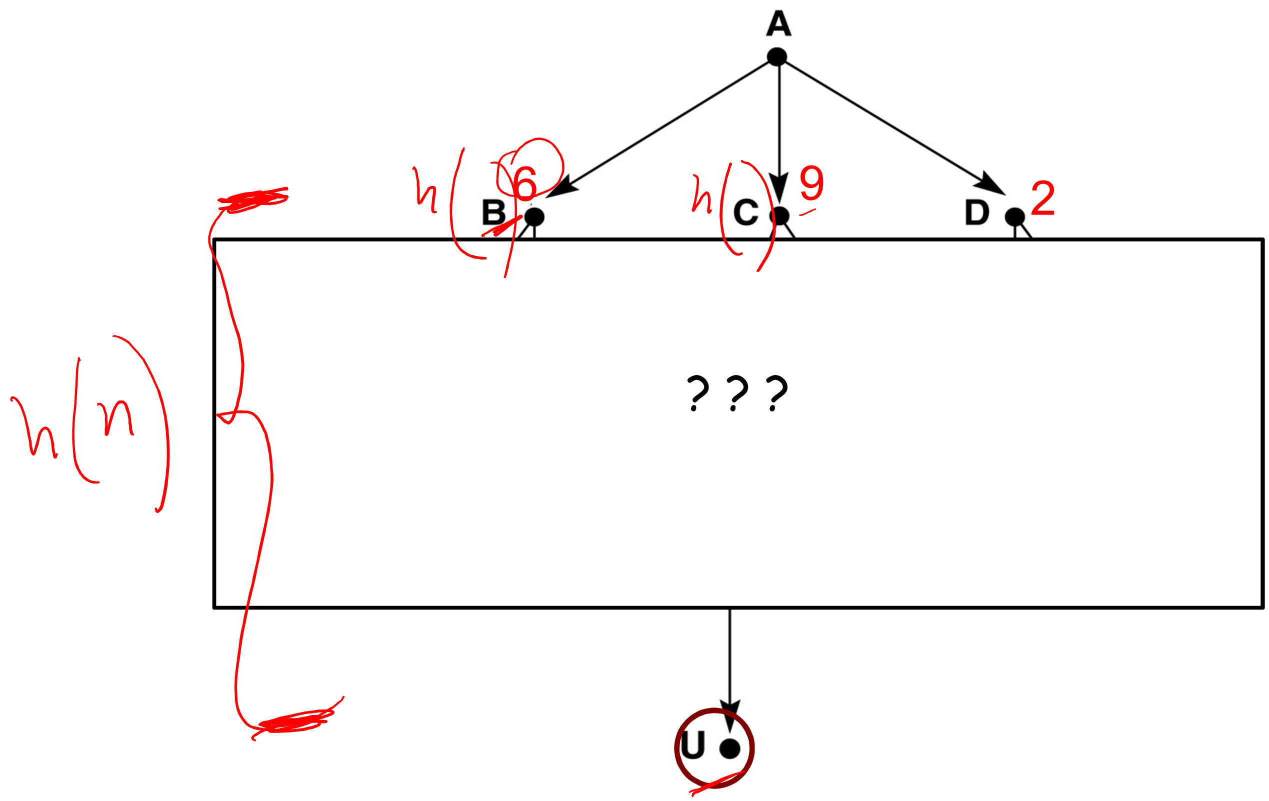
Artificial Intelligence: State Space Search } part 3 Informed Search Greedy Best First Search and Algorithms A and A* } video #5

- Russell & Norvig – Sections 3.5.1, 3.5.2, 4.1.1

Today

1. State Space Representation
2. State Space Search
   a) Overview
   b) Uninformed search
      1. Breadth-first Search and Depth-first Search
      2. Depth-limited Search
      3. Iterative Deepening
      4. Uniform Cost
   c) Informed search
      1. Intro to Heuristics
      2. Hill climbing
      3. Greedy Best-First Search
      4. Algorithms A & A*
      5. More on Heuristics
   d) Summary

h(n)



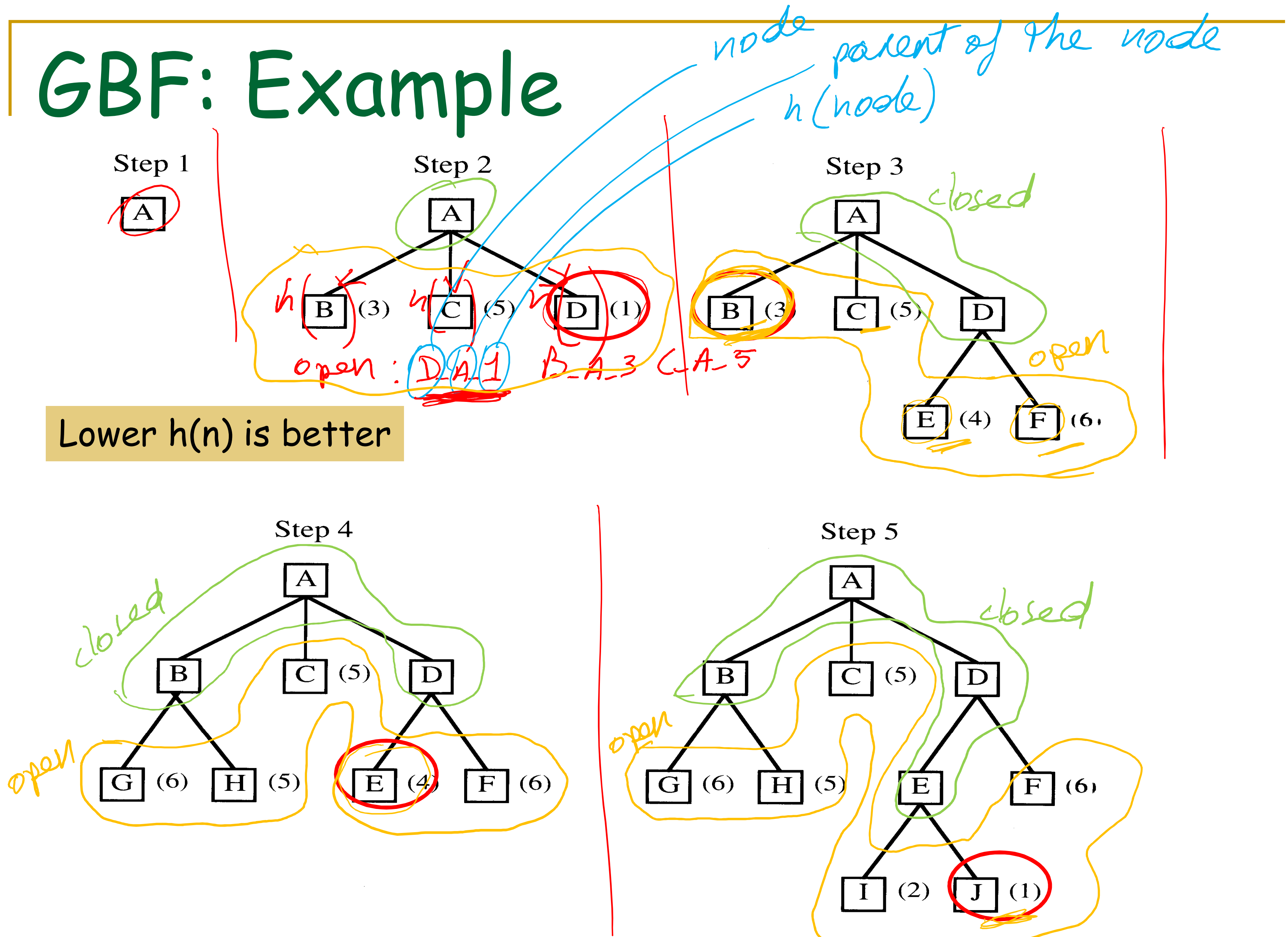
- **h(n)** = estimate of the lowest cost from *n* to *goal*

Greedy Best-First Search

- problem with hill-climbing:
  - no open list
  - --> can't backtrack
  - one move is selected and all others are forgotten
- solution to hill-climbing:
  - use "open" as a priority queue h(n)
  - this is called best-first search

- Best-first search:
  - Insert nodes in *open* list so that the nodes are sorted in ascending **h(n)**
  - Always choose the next node to visit to be the one with the best h(n) -- regardless of where it is in the search space
    lowest

GBF: Example



Lower h(n) is better

source: Rich & Knight, Artificial Intelligence, McGraw-Hill College 1991.

Notes on GBF

- If you have a good $h(n)$, best-first can find a solution very quickly
- The solution may not be the optimal one (lowest cost) but there is a good chance of finding it quickly

GBF Search: Example

node parent h(node)

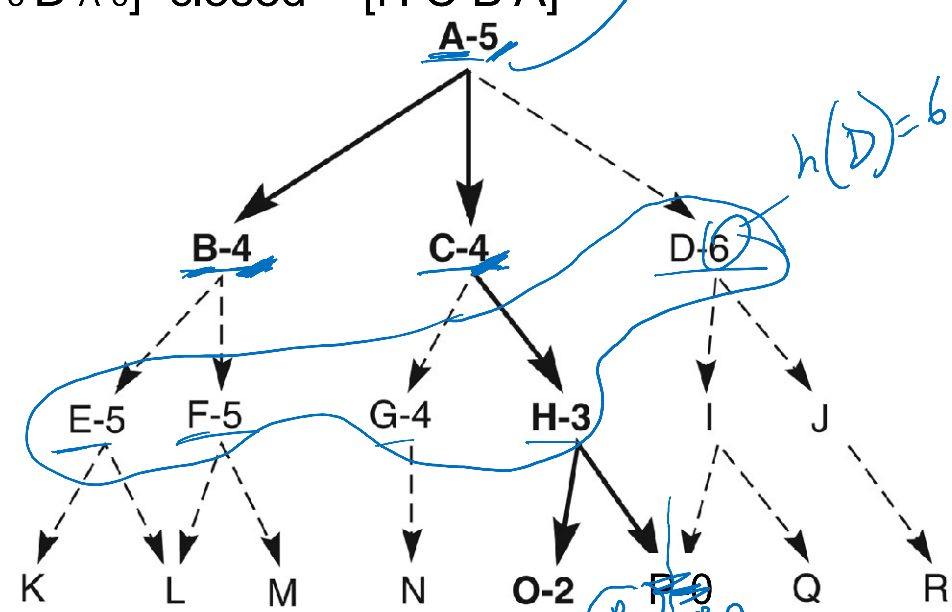
1. open = [A-null-5] closed = []
2. open = [B-A-4 C-A-4 D-A-6] *(arbitrary choice)* closed [A]
3. open = [C-A-4 E-B-5 F-B-5 D-A-6] closed = [B A]
4. open = [H-C-3 G-C-4 E-B-5 F-B-5 D-A-6] closed = [C B A]
5. open = [P-H-0 O-H-2 G-C-4 E-B-5 F-B-5 D-A-6] closed = [H C B A]
6. goal P found

solution path: A C H P

priority queue sorted by h(n)

Lower h(n) is better

h(A)=5



h(D)=6

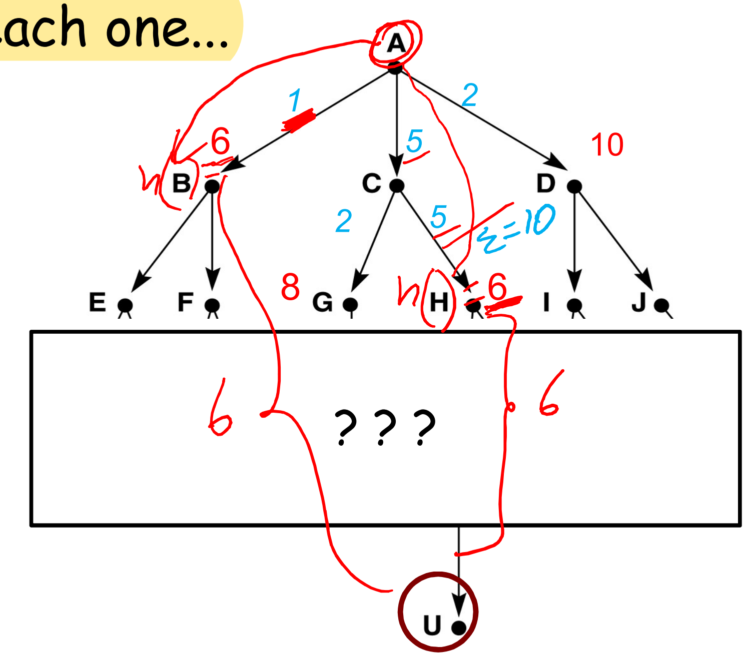
P-0 P-0

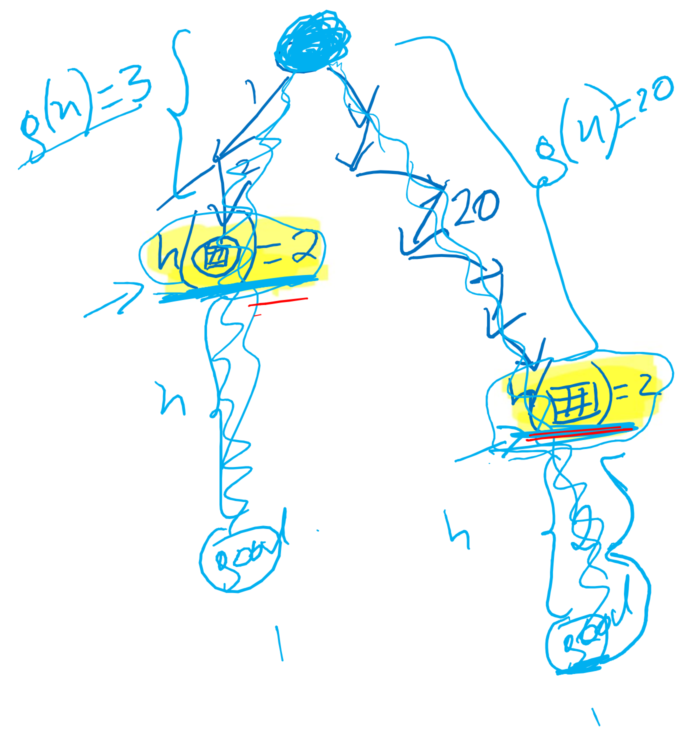
h(P)=0 // goal state

source: adapted from G. Luger (2005)

Problem with GBF search

- if 2 nodes have the same *h(n)*, no preference to the closest/least costly to reach one...





- Solution:
  - Maintain a cost count – *g(n)*
  - i.e. give preference to nodes with least expensive paths from root to n
  - i.e. combine *h(n)* and *g(n)*

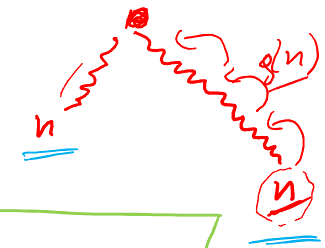
Today

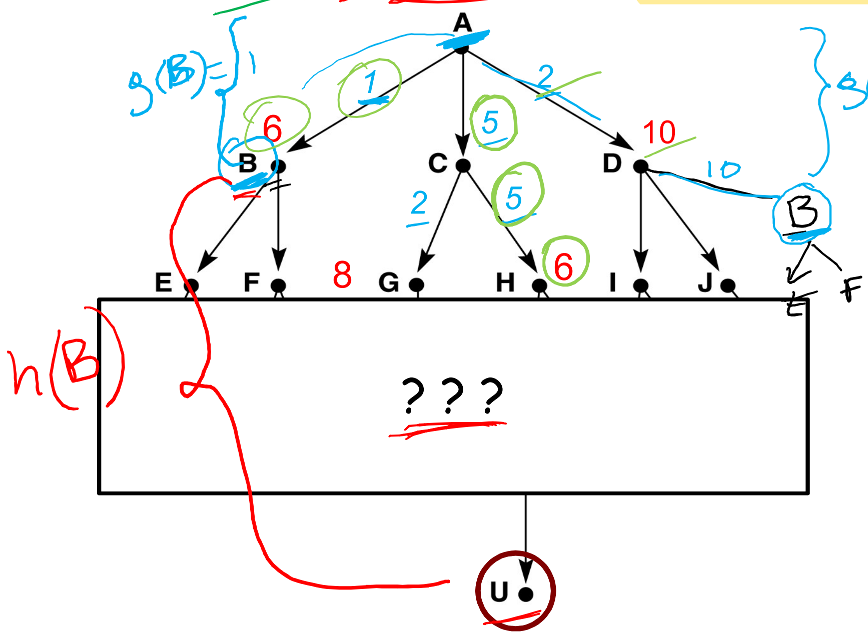
1. State Space Representation
2. State Space Search
   a) Overview
   b) Uninformed search
      1. Breadth-first Search and Depth-first Search
      2. Depth-limited Search
      3. Iterative Deepening
      4. Uniform Cost
   c) Informed search
      1. Intro to Heuristics
      2. Hill climbing
      3. Greedy Best-First Search
      4. Algorithms A & A*
      5. More on Heuristics
   d) Summary

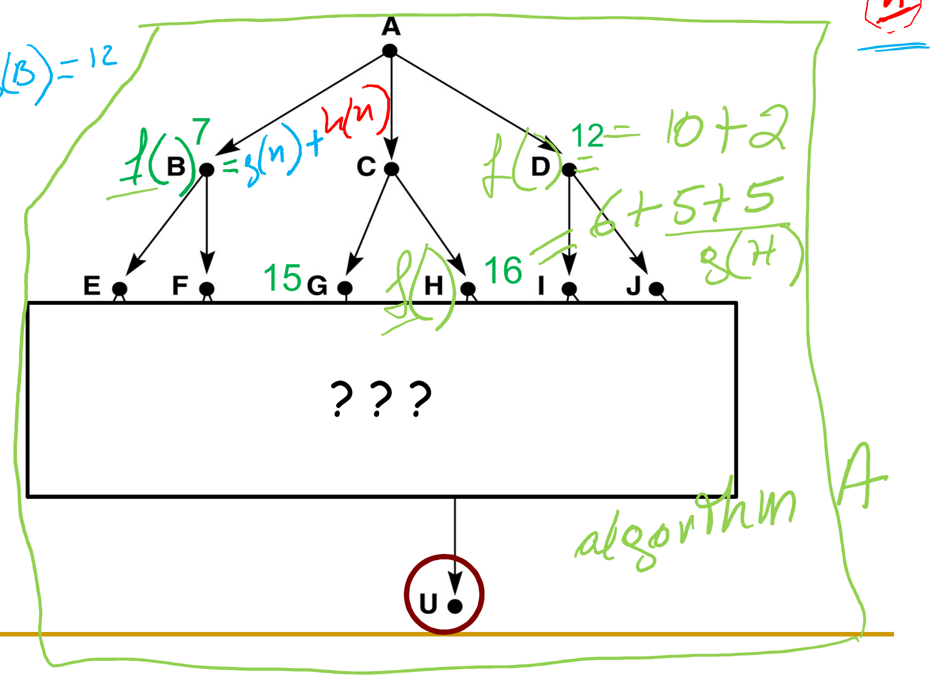
f(n) = h(n) + g(n)

- Modified evaluation function **f**:
  - $f(n) = \boxed{g(n) + h(n)}$
  - f(n) estimate of total cost along path through n
  - g(n) actual cost of path from start to node n
  - h(n) estimate of cost to reach goal from node n

from the start to the goal







Algorithms A and A*

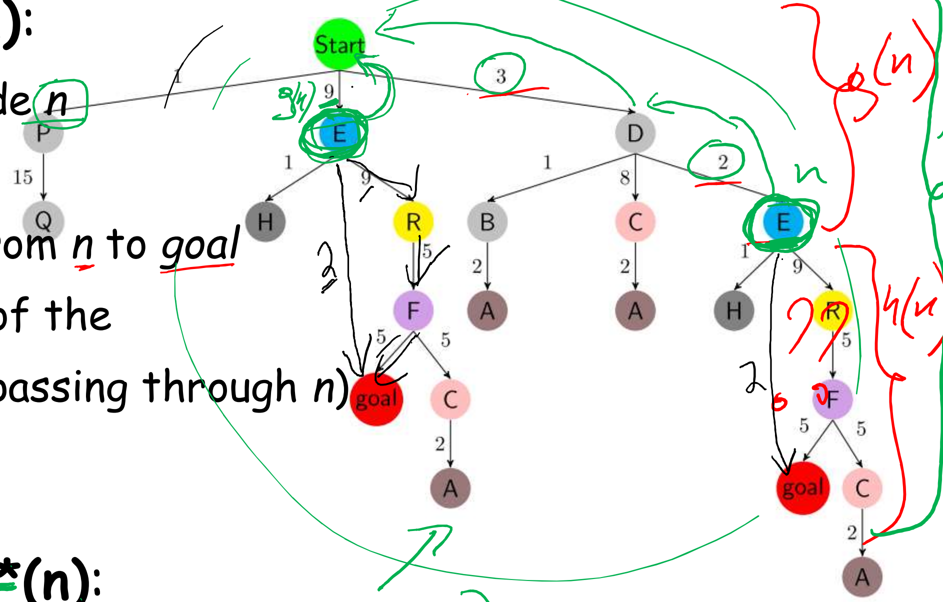
≠? (see next slides)

- similarly to Greedy Best first search:
  - keep an OPEN list as a priority queue
- But
  - OPEN is sorted by lowest f(n) = h(n) + g(n)

g(n)*, h(n)* and f*(n)

$g(n)$

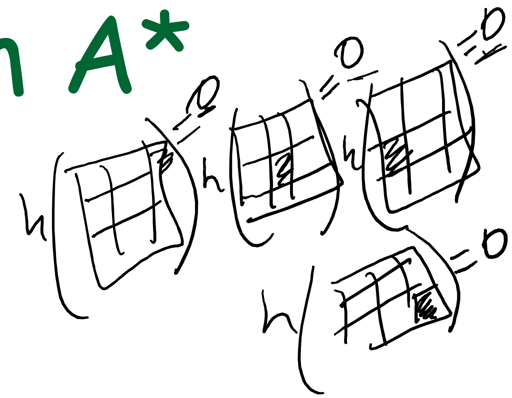
- We know that **f(n) = g(n) + h(n)**:
  - **g(n)** actual ~~current~~ cost from *start* to node *n*
    (maybe not be the lowest cost)
  - **h(n)** estimate of the lowest cost from *n* to *goal*
  - --> **f(n)** estimate of the lowest cost of the solution path (from *start* to *goal* passing through *n*)



- Let us define **f*(n) = g*(n) + h*(n)**:
  - **g*(n)** cost of lowest cost path from *start* to node *n* $g^*(n) = 3 + 2 = 5$
  - **h*(n)** actual lowest cost from *n* to *goal*   // unknown... what $h(n)$ is trying to estimate
  - --> **f*(n)** actual cost of lowest cost of the solution path (from *start* to *goal* passing through *n*)

✓ for ex in the graph above $h^*(E) = 2$

Algorithm A vs Algorithm A*



- IF
  - $g(n) \geq g^*(n)$ $\forall n$

    // ie. if the cost from the root to n is considered
- AND

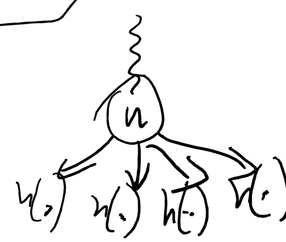
  our estimate actual cost
  - $\boxed{h(n) \leq h^*(n)}$ for all n $\forall n$

    $h^*(n)$

    // ie. h(n) never overestimates the true lowest cost from n to the goal
- THEN
  - algorithm A is called $\boxed{\text{algorithm } A^*}$

uniform cost
- uses $g(n)$
- uses $h(n)$ where $\boxed{h(n)=0 \ \forall n}$

what's the big deal?



--> algorithm A* is admissible

--> i.e. it guarantees to find the lowest cost solution path from the initial state to the goal

Algorithm A* vs GBF search

- given the same h(n):
  - *A* guarantees to find the lowest cost solution path*
  - *GBF does not*

- so is A* always "better" in real life?
  - not necessarily
  - computing g(n) can take time to compute
  - if client is not looking for the optimal (lowest cost) solution
  - a good-enough solution faster (i.e. GBF search) might be preferable

Today

1. State Space Representation
2. State Space Search
   a) Overview
   b) Uninformed search
      1. Breadth-first and Depth-first
      2. Depth-limited Search
      3. Iterative Deepening
      4. Uniform Cost
   c) Informed search
      1. Intro to Heuristics
      2. Hill climbing
      3. Greedy Best-First Search
      4. Algorithms A & A*
      5. More on Heuristics
   d) Summary

Up Next

1. State Space Representation
2. State Space Search
   a) Overview
   b) Uninformed search
      1. Breadth-first and Depth-first
      2. Depth-limited Search
      3. Iterative Deepening
      4. Uniform Cost
   c) Informed search
      1. Intro to Heuristics
      2. Hill climbing
      3. Greedy Best-First Search
      4. Algorithms A & A*
      5. More on Heuristics next video
   d) Summary