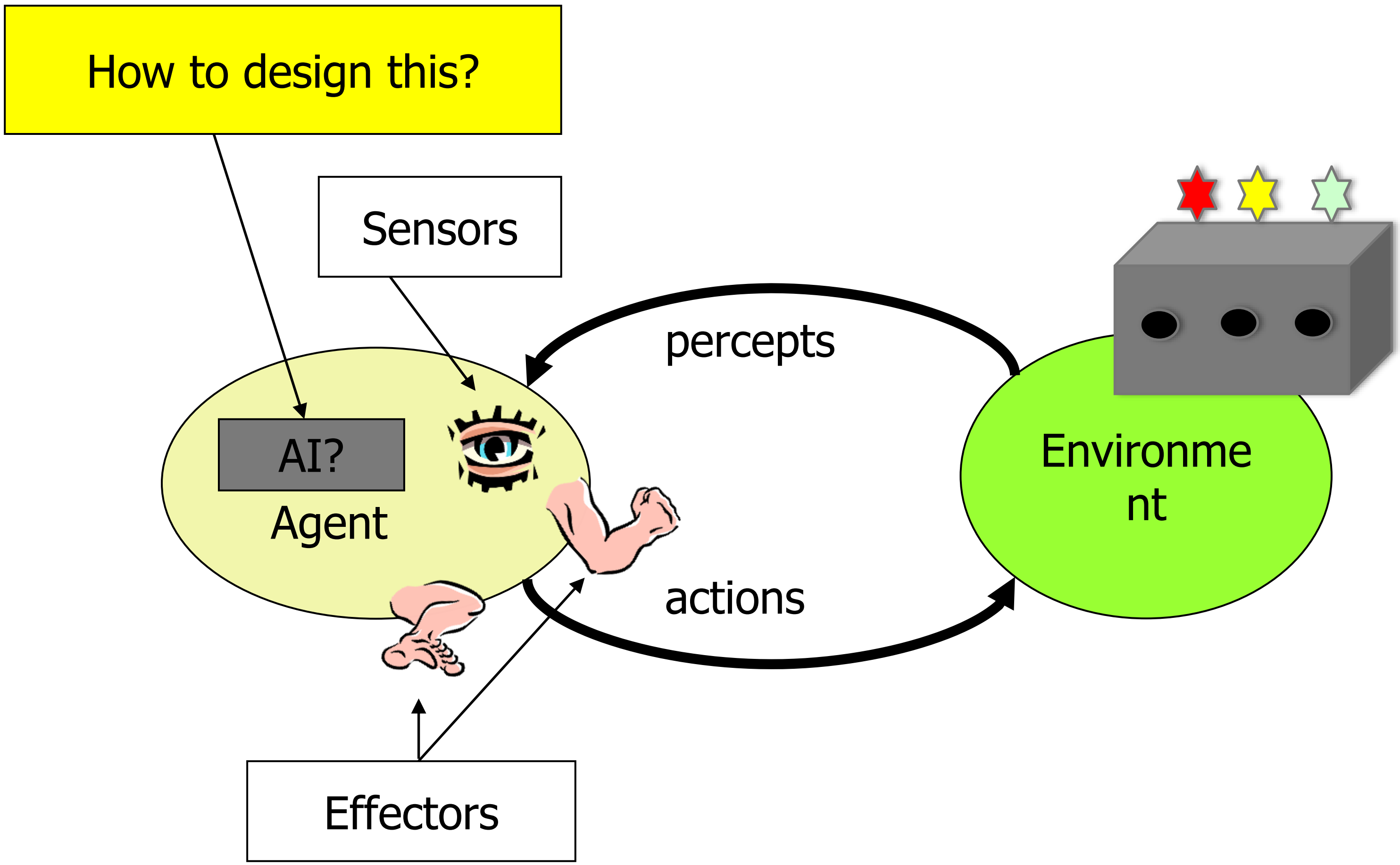
CSCI 561 - Foundation for Artificial Intelligence

DISCUSSION SECTION (WEEK 2)

PROF WEI-MIN SHEN SHEN@ISI.EDU

WHAT IS "PROBLEM SOLVING"? WHAT IS "SEARCH"?

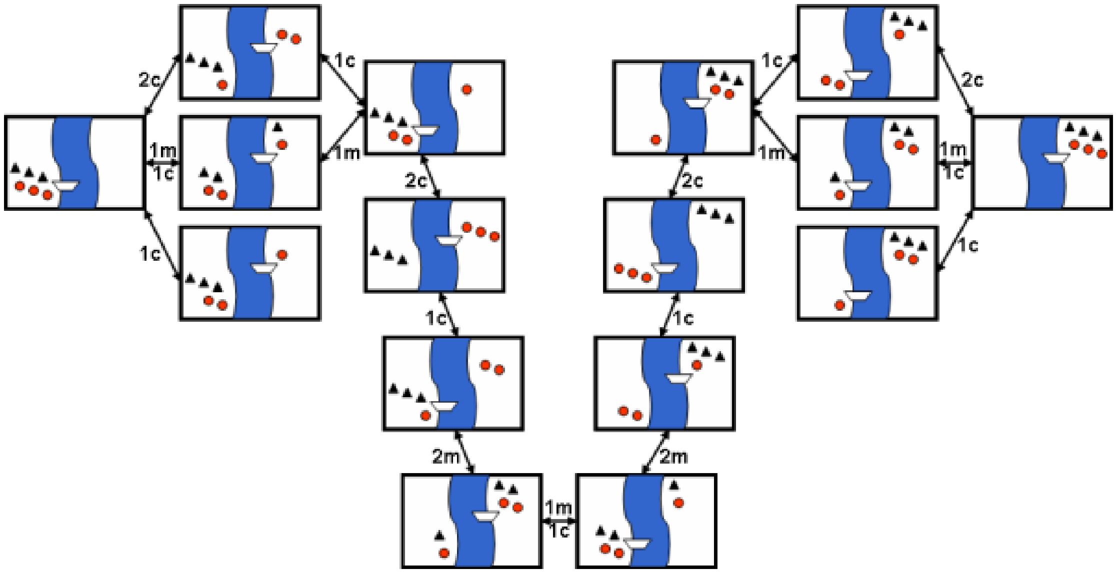


ESSENTIALS OF SEARCH

- **How to represent a “problem”?**
  - How to construct a Search Tree/Graph?
    - Nodes, Goals, Initials, Links
- **How to find a solution “systematically” or “optimally” in your representation?**
  - Use the uninformed algorithms you learned
  - Use the informed algorithms you learned

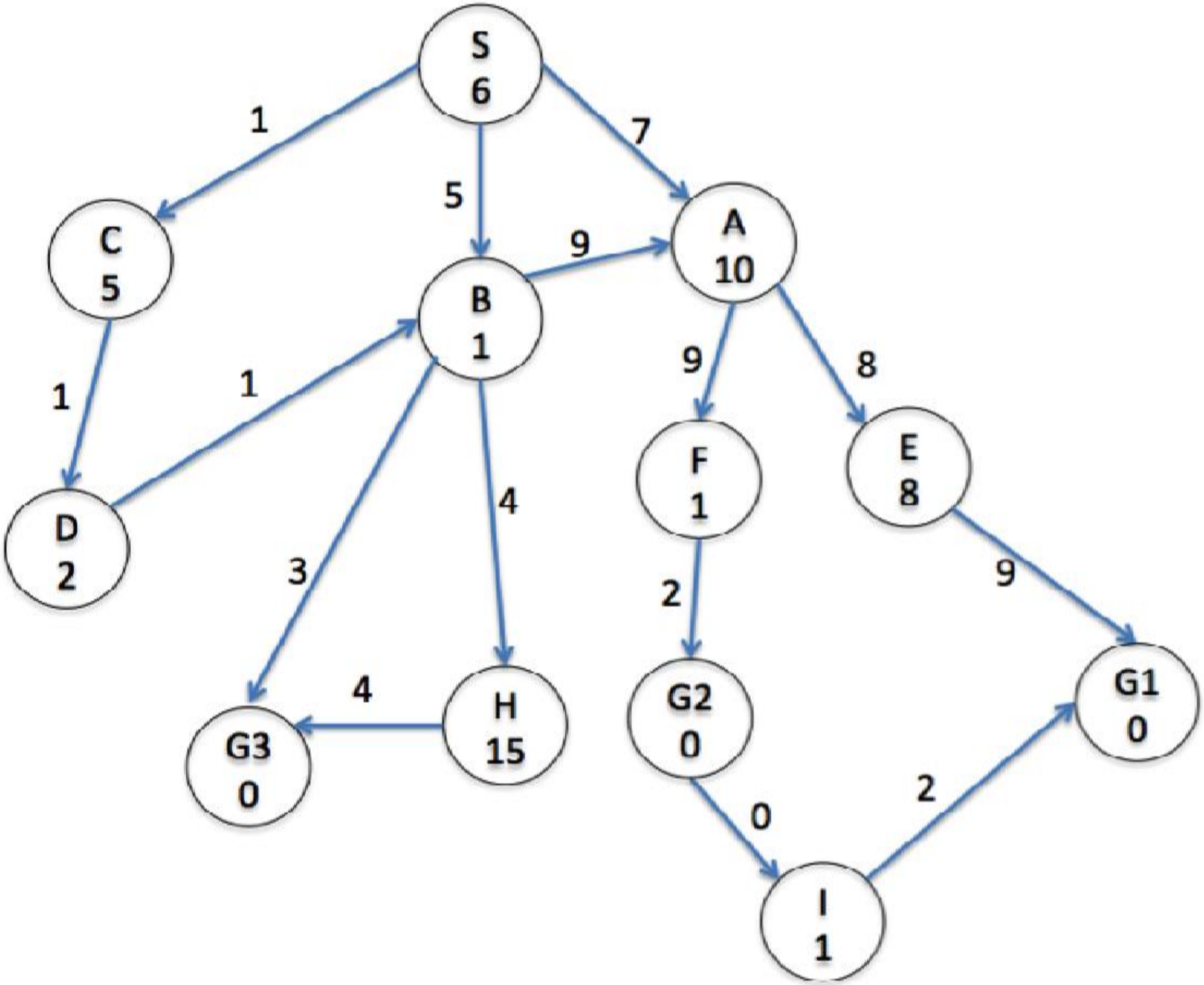
MISSIONARIES AND CANNIBALS

Did you find that there was much search involved in finding a solution?



Why do people have a hard time solving this problem?

SEARCH GRAPH



GRAPH SEARCH

```
function GRAPH-SEARCH(problem) return a solution or failure
  frontier ← MAKE-QUEUE(MAKE-NODE(problem.INITIAL-STATE))
  explored_set ← empty
  loop do
        if EMPTY?(frontier) then return failure
        node ← REMOVE-FIRST(frontier)
        if problem.GOAL-TEST applied to node.STATE succeeds
              then return SOLUTION(node)
        explored_set ← INSERT(node, explored_set)
        for each new_node in EXPAND(node, problem) do
              if NOT(MEMBER?(new_node, frontier)) and
                NOT(MEMBER?(new_node, explored_set))
                then frontier ← INSERT(new_node, frontier)
```

GRAPH SEARCH

function GRAPH-SEARCH(*problem*) **return** a solution or failure
  frontier ← MAKE-QUEUE(MAKE-NODE(problem.INITIAL-STATE))
  explored_set ← empty
  loop do
    if EMPTY?(*frontier*) **then return** failure
    node ← REMOVE-FIRST(*frontier*)
    if problem.GOAL-TEST applied to node.STATE succeeds
      then return SOLUTION(*node*)
    explored_set ← INSERT(*node, explored_set*)
    for each *new_node* **in** EXPAND(*node*, *problem*) **do**
      if NOT(MEMBER?(*new_node, frontier*)) and
        NOT(MEMBER?(*new_node, explored_set*))
        then *frontier* ← INSERT(*new_node, frontier*)

How to modify this algorithm to become the following algorithms? (important!)

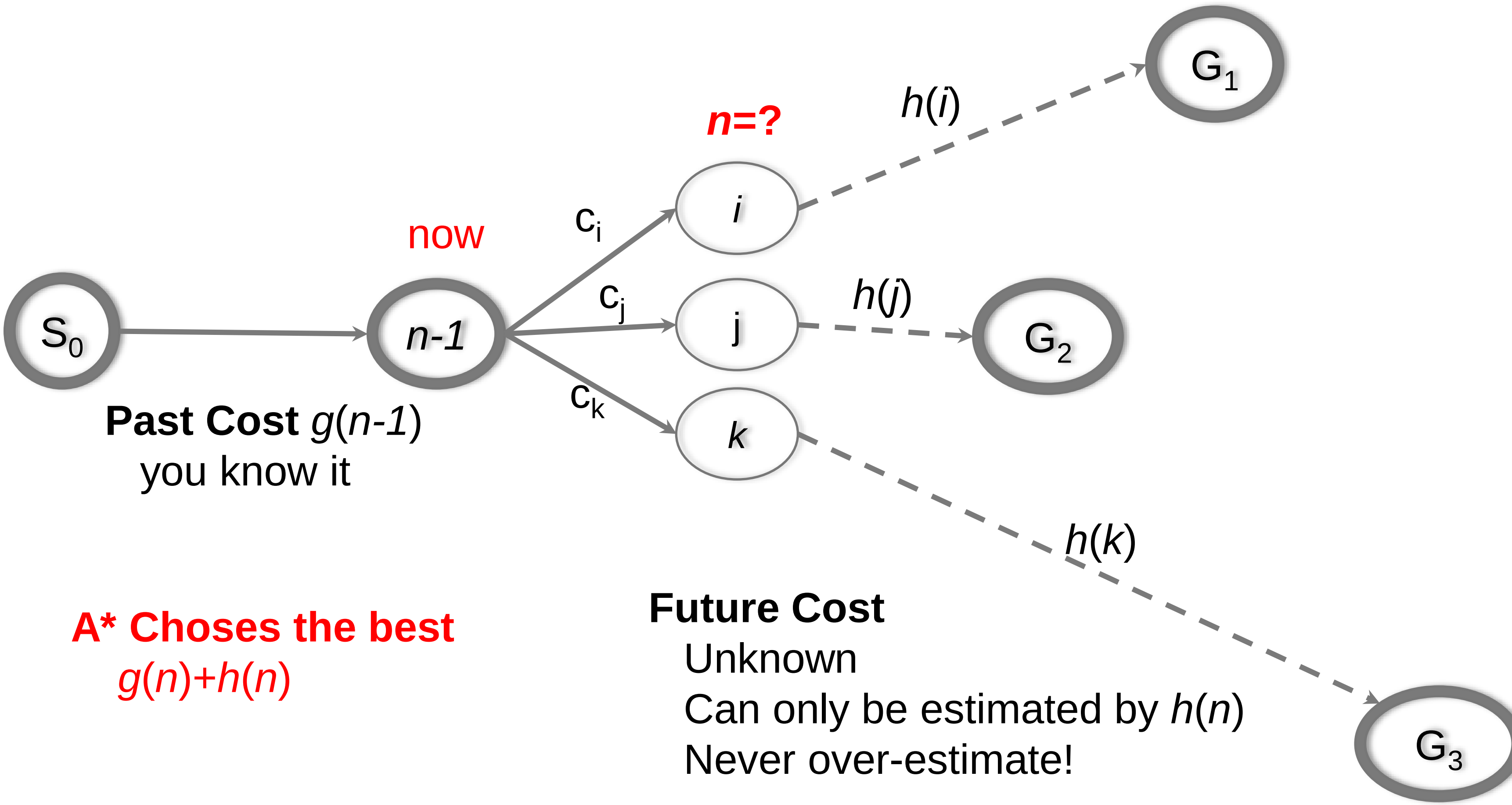
BFS

DFS

UCS

A*

A* = BEST-FIRST (PAST + ESTIMATED FUTURE)



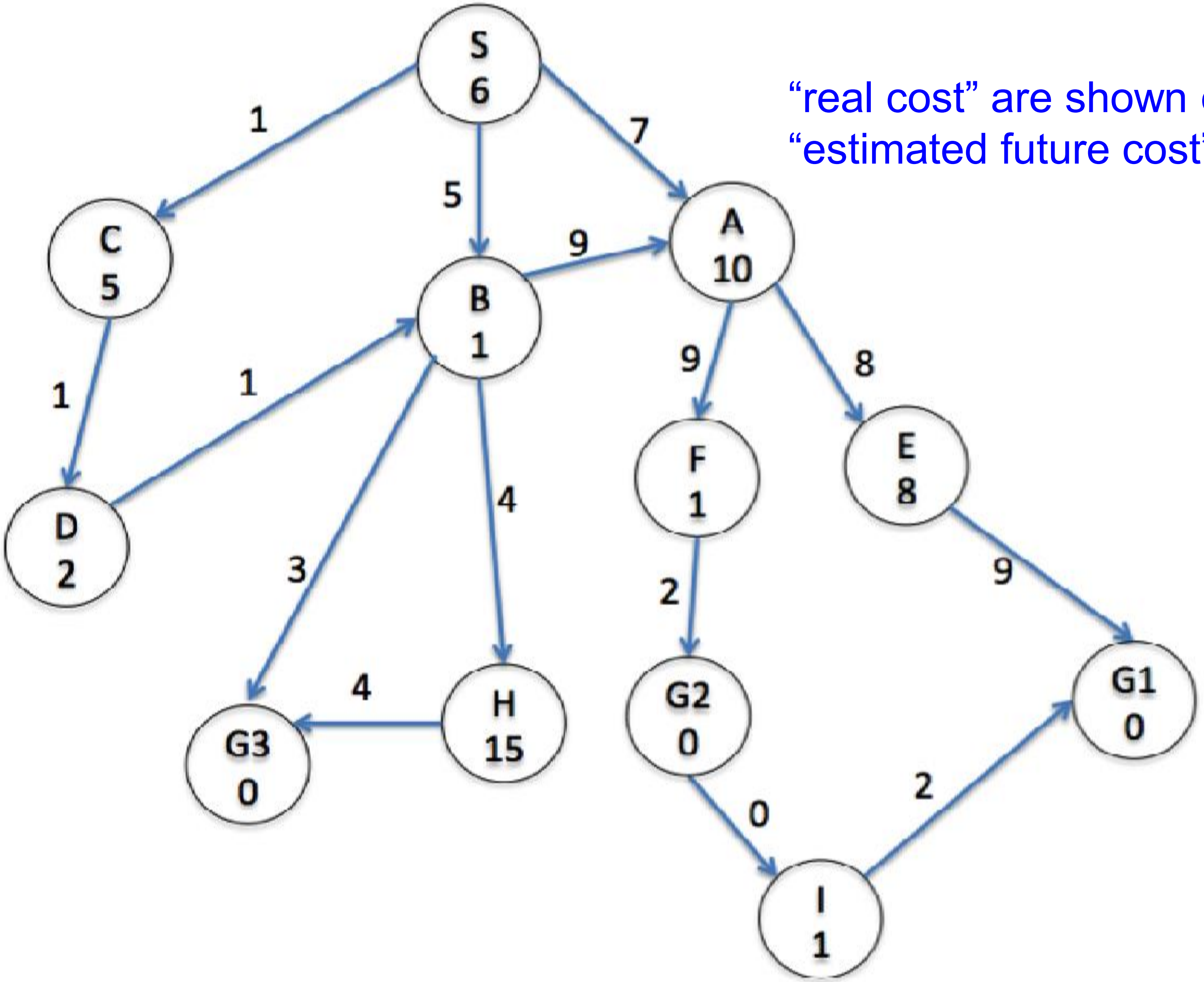
Past Cost $g(n\text{-}1)$
you know it

A* Choses the best
$g(n)+h(n)$

Future Cost
Unknown
Can only be estimated by $h(n)$
Never over-estimate!

Note: Uniform-cost search uses only $g(n)$, no $h(n)$.

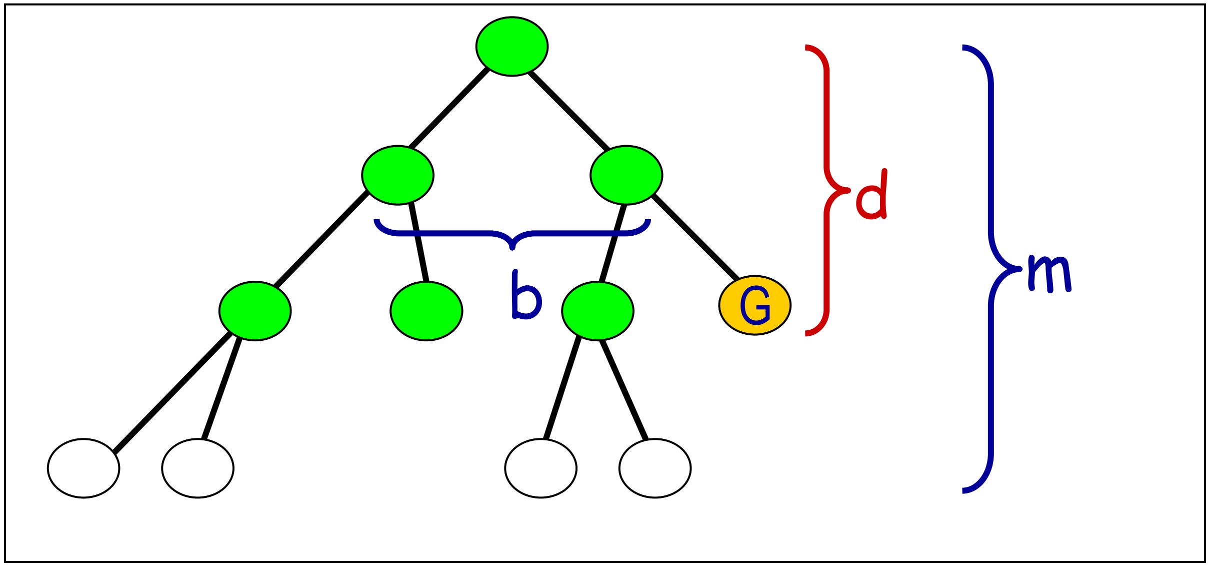
Is it good to have h(x)=0 for all x?



"real cost" are shown on edge
"estimated future cost" are inside circle

TIME COMPLEXITY OF BREADTH-FIRST SEARCH

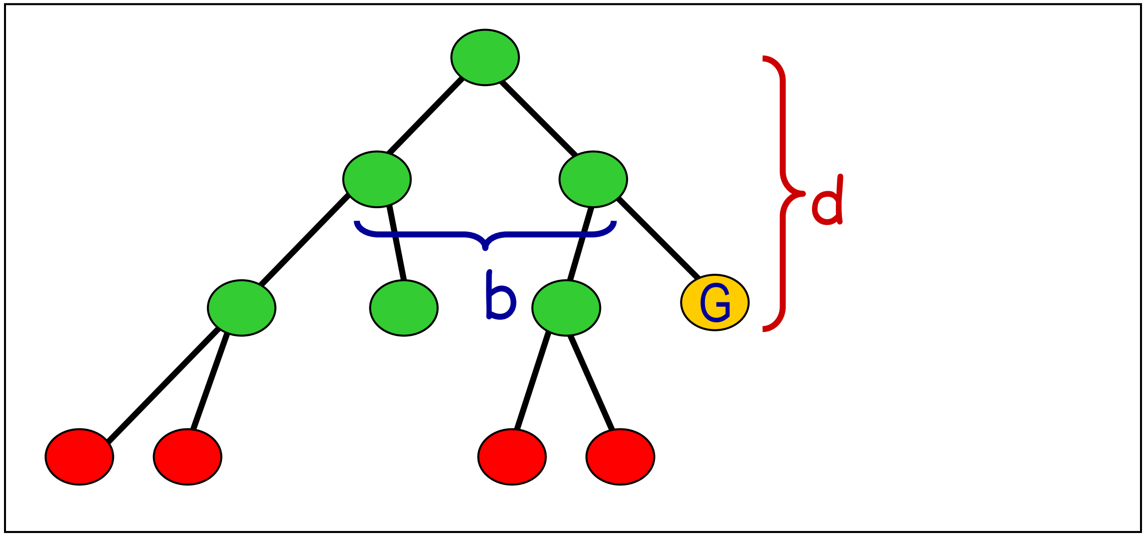
- Illustrates when goal check is done when node **is selected for expansion**
- If a goal node is found on depth **d** of the tree, all nodes up till that depth are created and examined (note: and the children of nodes at depth d are created and queued, but not yet examined).

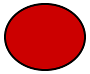


- Thus: $O(b^{d+1})$

SPACE COMPLEXITY OF BREADTH-FIRST SEARCH

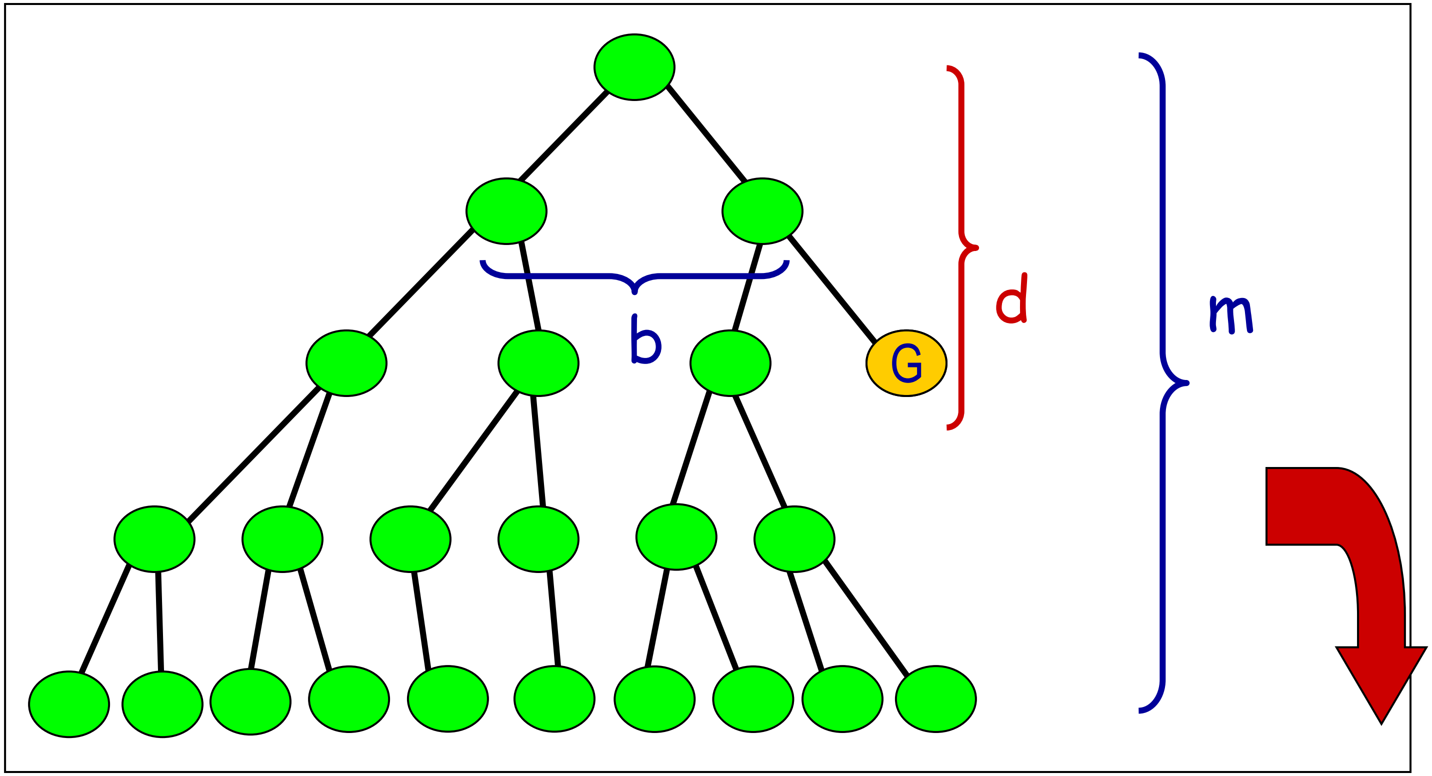
- Illustrates when goal check is done when node **is selected for expansion**
- Largest number of nodes in FRONTIER is reached on the level d+1 just beyond the goal node.



- QUEUE contains all  nodes. (Thus: 4) .
- In General: $b^{d+1} - b \sim b^{d+1}$

TIME COMPLEXITY OF DEPTH-FIRST SEARCH

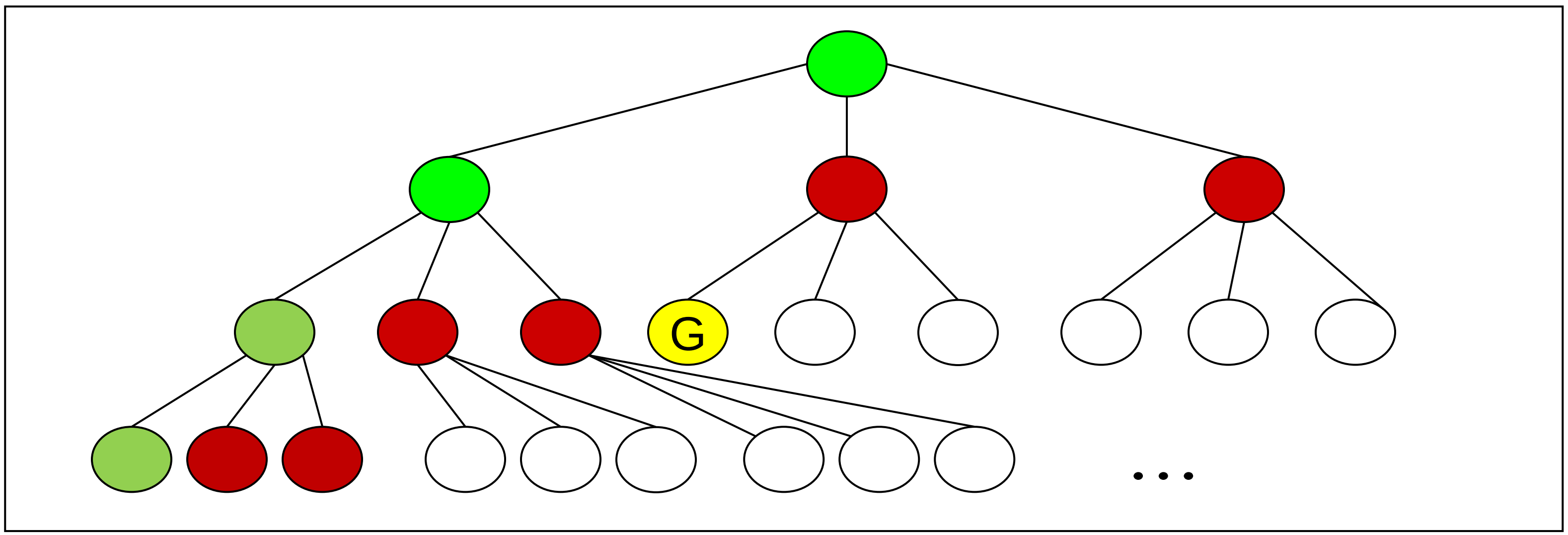
- In the worst case:
  - the (only) goal node may be on the right-most branch,

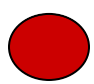


- Time complexity = $b^m + b^{m-1} + \ldots + 1 = O(b^m)$

SPACE COMPLEXITY OF DEPTH-FIRST

- Largest number of nodes in FRONTIER is reached in bottom left-most node.
- Example: m = 3, b = 3 :



- FRONTIER contains all  nodes. Thus: 6.
- In General FRONTIER contains ((b-1) * m)
- Order: O(m*b)

SEARCH IN AI APPLICATIONS

Is search involved in these AI applications? If so, in what part or parts of the application?

- Building a driverless car that will drive down a roadway. (Leave aside the search involved in route planning.)
- Building a system like Siri.
- Text-to-speech synthesis

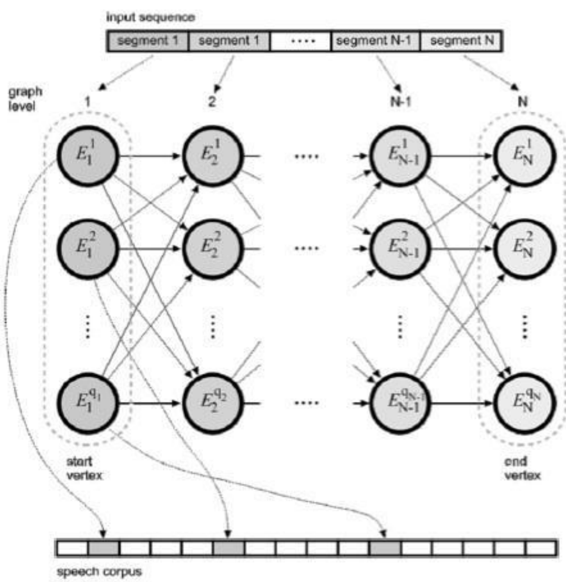
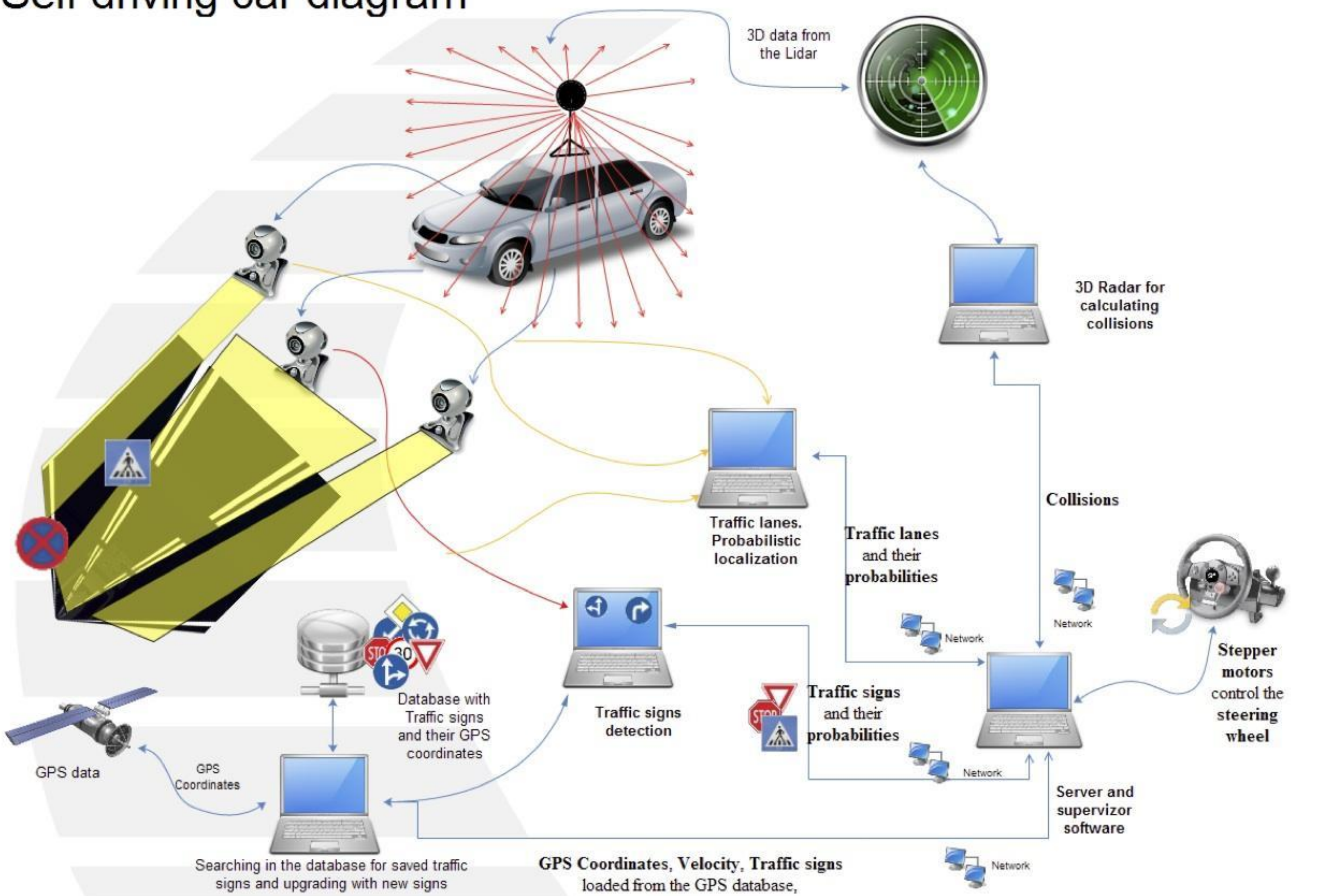


Figure 1. Structure of the graph for finding the optimal speech unit sequence; E_1^i are the graph initial-level vertices, E_N^i are the graph final-level vertices.

Self driving car diagram



WHAT YOU SHOULD KNOW

- What is the difference between uninformed and informed search? Which ones are optimal?
- What are the advantages and disadvantages of depth-first search?
- Be familiar with the differences between search strategies shown in Figure 3.21

Criterion	Breadth-First	Uniform-Cost	Depth-First	Depth-Limited	Iterative Deepening	Bidirectional (if applicable)
Complete?	Yes[a]	Yes[a,b]	No	No	Yes[a]	Yes[a,d]
Time	$O(b^d)$	$O(b^{1+\lfloor C^*/\epsilon \rfloor})$	$O(b^m)$	$O(b^\ell)$	$O(b^d)$	$O(b^{d/2})$
Space	$O(b^d)$	$O(b^{1+\lfloor C^*/\epsilon \rfloor})$	$O(bm)$	$O(b\ell)$	$O(bd)$	$O(b^{d/2})$
Optimal?	Yes[c]	Yes	No	No	Yes[c]	Yes[c,d]

WANT MORE?

BigO and complexity:

https://apelbaum.wordpress.com/2011/05/05/big-o/