# COMPSCI 761:
# ADVANCED TOPICS IN ARTIFICIAL INTELLIGENCE

# INFORMED SEARCH II

Anna Trofimova, July 2022

# TODAY

- A* Search
- Heuristic properties

# RECAP: UNIFORM-COST SEARCH

- The simplest search method that is guaranteed to find a minimum cost path is Lowest-Cost-First Search or Uniform-Cost Search
  - is similar to breadth-first search, but it selects a path with the lowest cost.
  - the frontier is implemented as a priority queue ordered by the **cost function**.

$$cost(\langle n_0, \ldots, n_k \rangle) = \sum_{i=1}^{k} cost(\langle n_{i-1}, n_i \rangle)$$

# RECAP: UNIFORM-COST SEARCH

- Expand root first, then expand least-cost unexpanded node
- Implementation: priority queue  = insert nodes in order of increasing path cost - (lowest path cost g(n)).
- Reduces to Breadth First Search when all actions have same cost
- Finds the cheapest goal provided path cost is monotonically increasing along each path (i.e. no negative-cost steps)

# RECAP: PROPERTIES OF UNIFORM-COST SEARCH

| Algorithm | Completeness | Admissibility | Space | Time |
|---|---|---|---|---|
| **Uniform-Cost Search** | guaranteed, if b is finite and cost ≥ ε | guaranteed | $O(b^{[1 + C / \varepsilon]})$ | $O(b^{[1 + C / \varepsilon]})$ |

- Complete? Yes, if b is finite and if step cost ≥ ε  with ε > 0
- Time?  $O(b^{[1 + C / \varepsilon]})$ where C* = cost of the optimal solution and assume  every action costs at least ε
- Space? $O(b^{[1 + C / \varepsilon]})$
- Optimal? Yes – nodes expanded in increasing order of g(n)

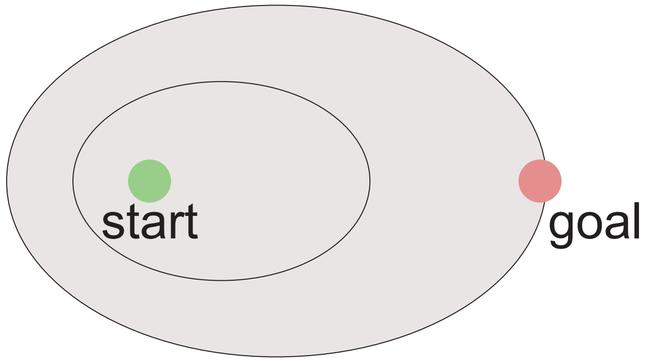
# RECAP: GREEDY SEARCH

- Informed search uses information external to the problem description to guide its selection of nodes to expand
- One source of external information are heuristics, i.e., estimates of how far a node is from its nearest goal state.
- Specifically, heuristics are used to help select nodes.
- The (best-first) greedy search algorithm only uses the heuristic.
- Greedy search tends to find goal nodes quickly but not necessarily the best goal nodes.

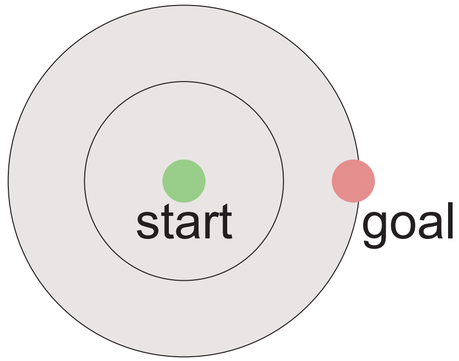
# RECAP: WHAT WE WOULD LIKE TO HAVE

Guide search **towards the goal** instead of **all over the place**

**Informed**



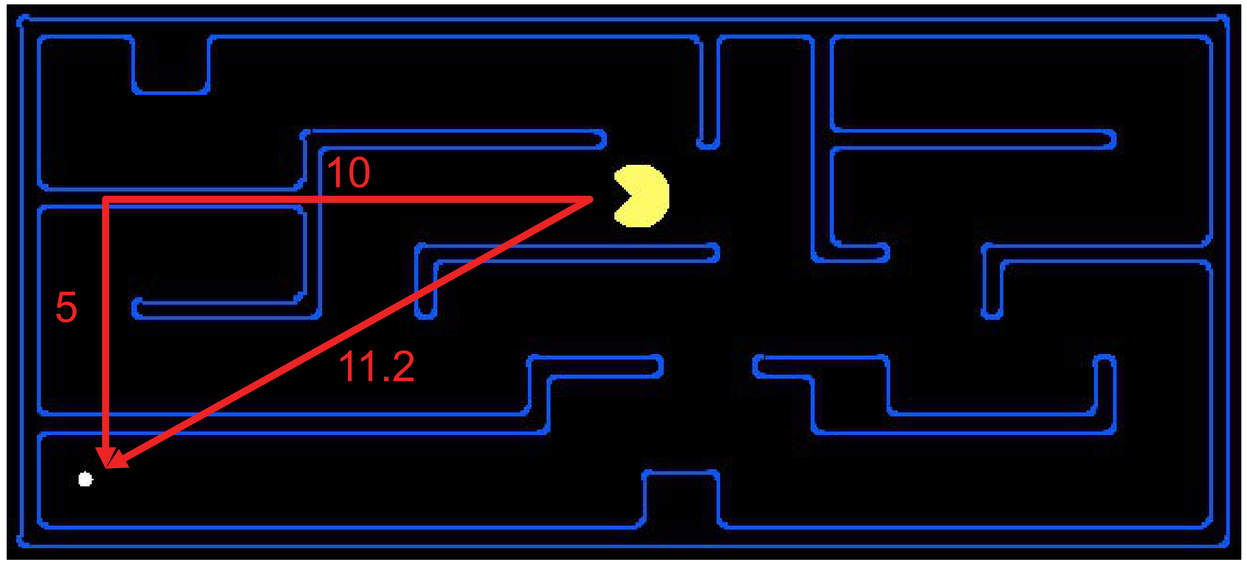

**Uninformed**



# RECAP: INFORMED SEARCH HEURISTIC

A heuristic is:

- A function that ***estimates*** how close a state is to a goal
- Designed for a particular search problem
- Examples: Manhattan distance, Euclidean distance for path finding



# RECAP: GREEDY BEST-FIRST SEARCH

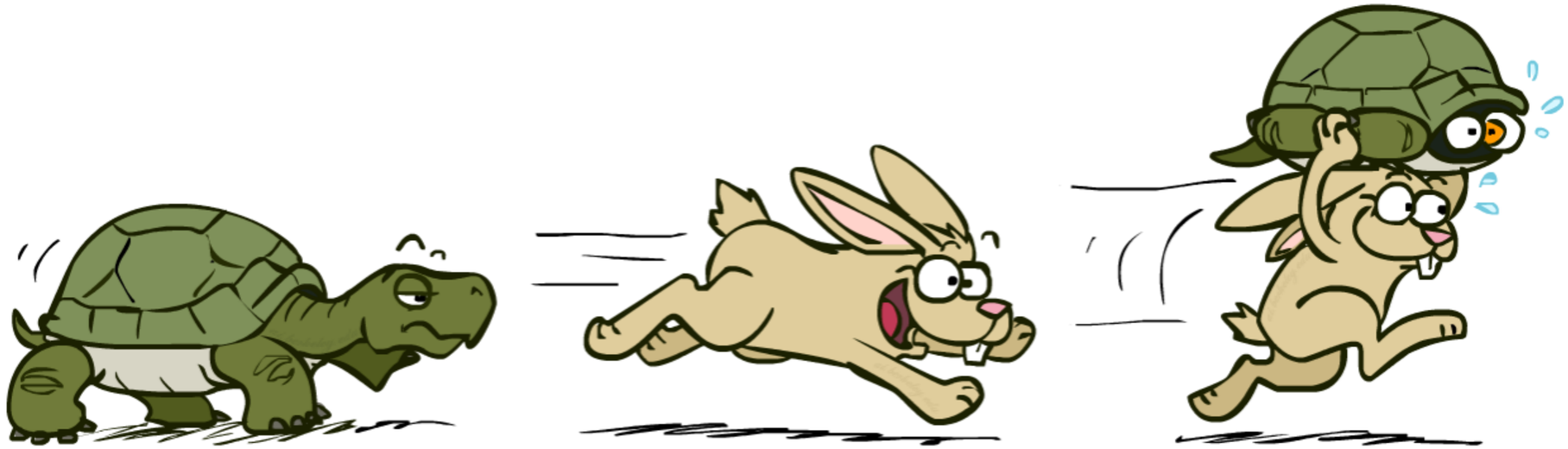
- Idea: select the path whose end is closest to a goal according to the heuristic function.
- Greedy Best-First Search - Best-First Search selects the next node for expansion using the heuristic function for its evaluation function, i.e. $f(n) = h(n)$
  - $h(n)=0 \Rightarrow n$ is a goal state
  - i.e. greedy search minimises the estimated cost to the goal; it expands whichever node $n$ is estimated to be closest to the goal.
- It treats the frontier as a priority queue ordered by h.
- Greedy: tries to “bite off” as big a chunk of the solution as possible, without worrying about long-term consequences.

# RECAP: PROPERTIES OF GREEDY BEST-FIRST SEARCH

| Algorithm | Completeness | Admissibility | Space | Time |
|---|---|---|---|---|
| **Greedy Best-First Search** | guaranteed, if b is finite and with repeated-state checking | not guaranteed | $O(b^m)$ | $O(b^m)$ |

- Complete: generally no! can get stuck in loops, e.g., but complete in finite space with repeated-state checking
- Time: $O(b^m)$, where $m$ is the maximum depth in search space.
- Space: $O(b^m)$ (retains all nodes in memory)
- Optimal: No!
- Therefore Greedy Search has the same time deficits as Depth-First Search. However, a good heuristic can reduce time and memory costs substantially.

# A*: A COMBINATION OF UCS AND GREEDY



Uniform-Cost Search | Greedy Search | A*

# COMBINING UCS AND GREEDY

Uniform-cost orders by path cost, or ***backward*** *cost* g(n)

Greedy orders by goal proximity, or ***forward*** *cost* h(n)

A* Search orders by the sum: f(n) = g(n) + h(n)

# A* SEARCH

- Idea: Use both cost of path generated and estimate to goal to order nodes on the frontier
- $g(n)$ = cost of path from start to $n$; $h(n)$ = estimate from $n$ to goal
- Order priority queue using function $f(n) = g(n) + h(n)$
- $f(n)$ is the estimated cost of the cheapest solution extending this path

# A* SEARCH

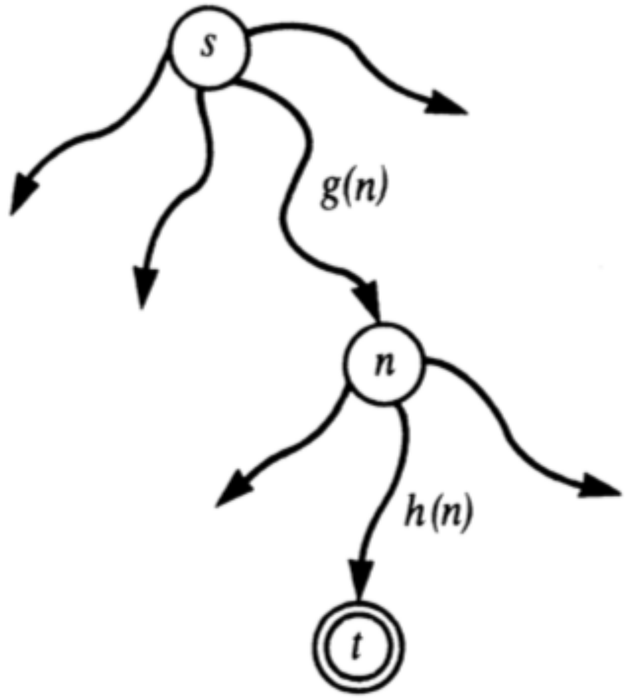
- A* Search uses evaluation function $f(n) = g(n) + h(n)$
  - $g(n)$ = cost from initial node to node $n$
  - $h(n)$ = estimated cost of cheapest path from $n$ to goal
  - $f(n)$=estimated total cost of cheapest solution through node $n$
- Combines uniform-cost search and greedy search
- Greedy Search minimizes $h(n)$
  - efficient but not optimal or complete
- Uniform Cost Search minimizes $g(n)$
  - optimal and complete but not efficient

# HEURISTIC FUNCTION

Heuristic estimate *f(n) of* the cost of the cheapest paths from **s** to **t** via **n**: f(n) = g(n)+ h(n)

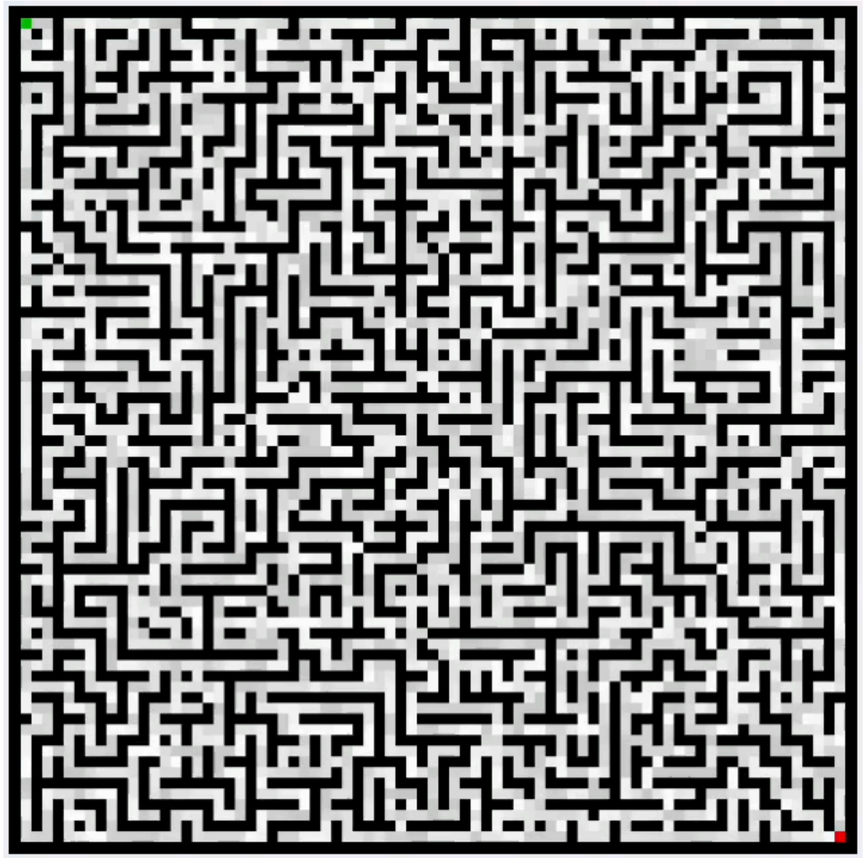
*g(n)* is an estimate of the cost of an optimal path from s to n

*h(n)* is an estimate of the cost of an optimal path from n to t.



# GUESS THE SEARCH PATH

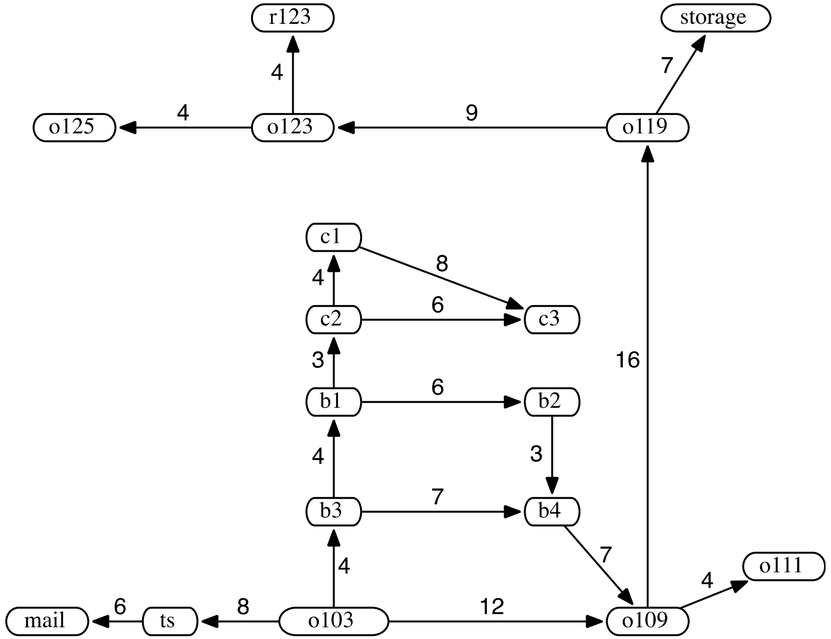
Start



Goal

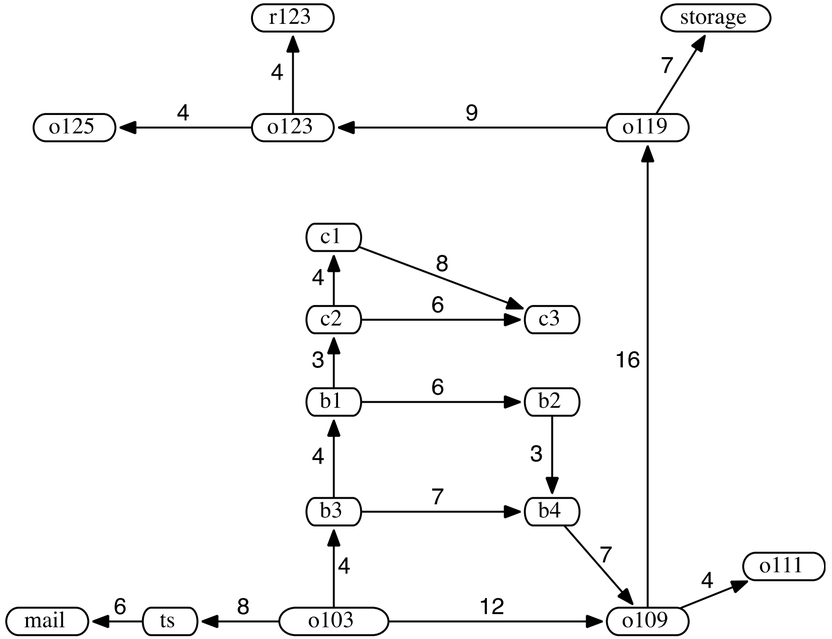
Weighted maze, Euclidean distance heuristic

# A* SEARCH - THE DELIVERY ROBOT

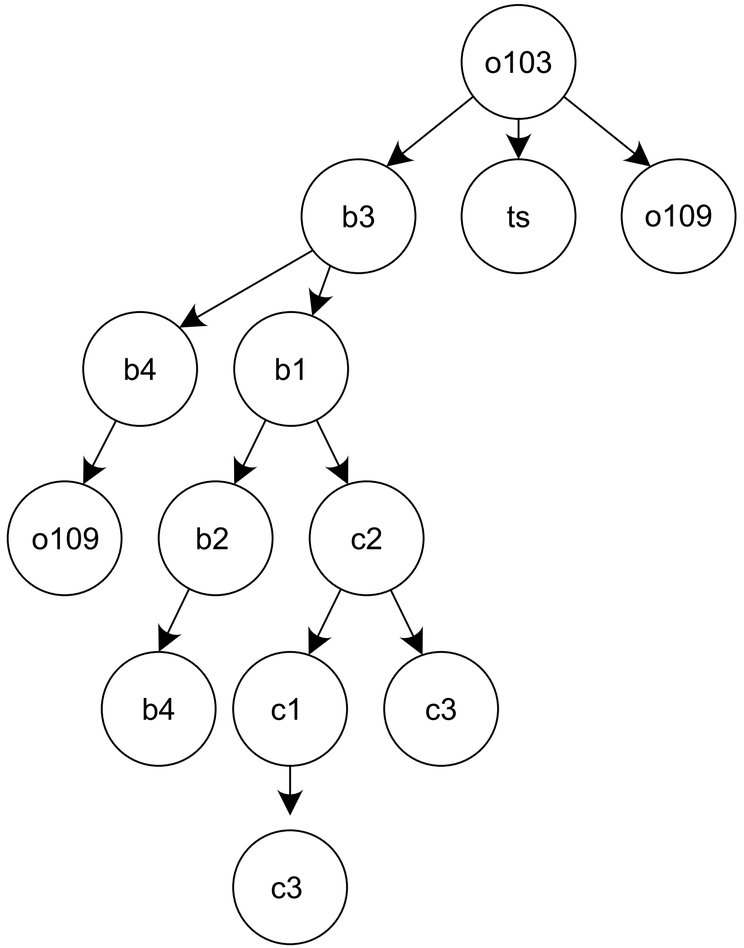



$$
\begin{array}{rcrcrc}
h(mail) & = & 26 & h(ts) & = & 23 \\
h(o109) & = & 24 & h(o111) & = & 27 \\
h(o123) & = & 4 & h(o125) & = & 6 \\
h(b1) & = & 13 & h(b2) & = & 15 \\
h(b4) & = & 18 & h(c1) & = & 6 \\
h(c3) & = & 12 & h(storage) & = & 12
\end{array}
\qquad
\begin{array}{rcrcrc}
h(o103) & = & 21 \\
h(o119) & = & 11 \\
h(r123) & = & 0 \\
h(b3) & = & 17 \\
h(c2) & = & 10
\end{array}
$$

# A* SEARCH - THE DELIVERY ROBOT




$$
\begin{array}{lll}
h(mail) = 26 & h(ts) = 23 & h(o103) = 21 \\
h(o109) = 24 & h(o111) = 27 & h(o119) = 11 \\
h(o123) = 4 & h(o125) = 6 & h(r123) = 0 \\
h(b1) = 13 & h(b2) = 15 & h(b3) = 17 \\
h(b4) = 18 & h(c1) = 6 & h(c2) = 10 \\
h(c3) = 12 & h(storage) = 12 &
\end{array}
$$

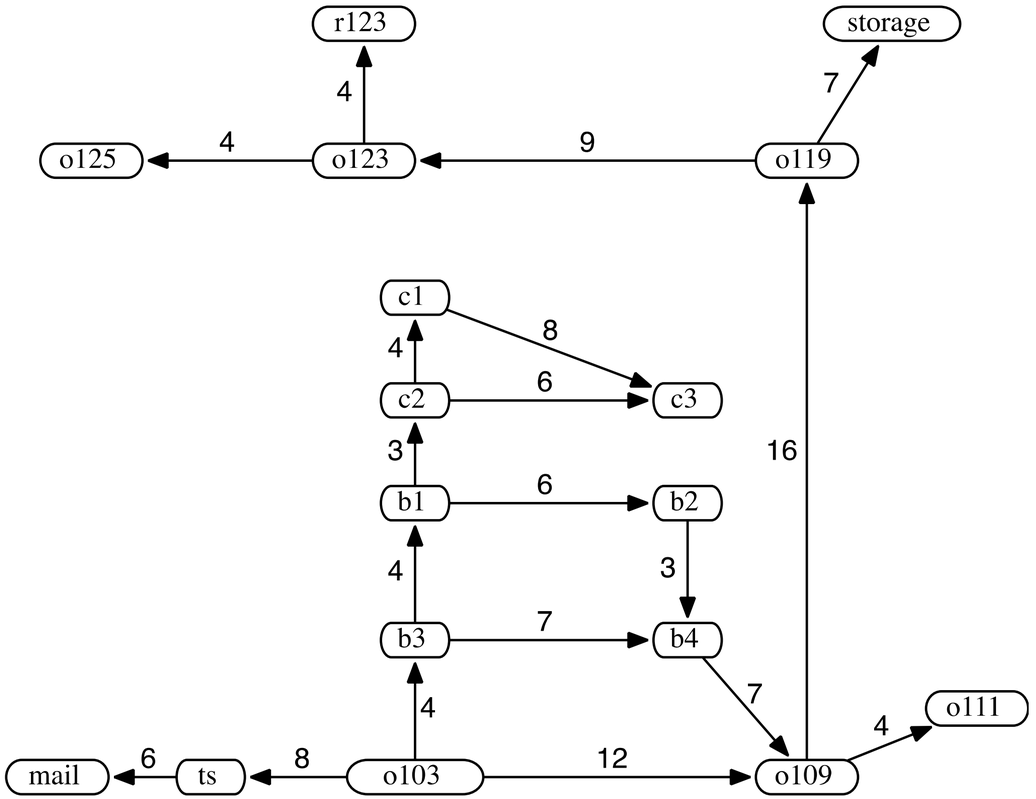


# A* SEARCH - THE DELIVERY ROBOT

$$
\begin{array}{rcrcrcr}
h(mail) & = & 26 & h(ts) & = & 23 & h(o103) & = & 21 \\
h(o109) & = & 24 & h(o111) & = & 27 & h(o119) & = & 11 \\
h(o123) & = & 4 & h(o125) & = & 6 & h(r123) & = & 0 \\
h(b1) & = & 13 & h(b2) & = & 15 & h(b3) & = & 17 \\
h(b4) & = & 18 & h(c1) & = & 6 & h(c2) & = & 10 \\
h(c3) & = & 12 & h(storage) & = & 12 & & &
\end{array}
$$

$$f(\langle o103, b3 \rangle) = cost(\langle o103, b3 \rangle) + h(b3) = 4 + 17 = 21.$$

A lowest-cost path to the goal is eventually found. The algorithm is forced to try many different paths, because several of them temporarily seemed to have the lowest cost. It still does better than either lowest-cost-first search or greedy best-first search.



# A* SEARCH

- A* Search minimises $f(n) = g(n) + h(n)$
  - idea: preserve efficiency of Greedy Search but avoid expanding paths that are already expensive

- Question: is A* Search optimal and complete?
- Answer: Yes! provided $h()$ is **admissible** in the sense that it never overestimates the cost to reach the goal or **consistent**.

# CONDITIONS FOR OPTIMALITY: ADMISSIBILITY AND CONSISTENCY

- The first condition we require for optimality is that h(n) be an admissible heuristic.

- An admissible heuristic is one that *never overestimates* the cost to reach the goal.
  - Because g(n) is the actual cost to reach n along the current path, and
  - f (n) = g(n) + h(n), we have as an immediate consequence that f(n) never overestimates the true cost of a solution along the current path through n.

- Admissible heuristics are by nature optimistic because they think the cost of solving the problem is less than it actually is.

# A* SEARCH – CONDITIONS FOR OPTIMALITY

- Heuristic ***h*** is called **admissible** if
- $\forall n\ h(n) \leq h^*(n)$ where $h^*(n)$ is true cost from $n$ to the goal

- If ***h*** is **admissible** then ***f*** **(*n*)** never overestimates the actual cost of the best solution through *n.*

- Example: $\boldsymbol{h_{SLD}}$ is admissible for delivery robot problem because the shortest path between any two points is a line.

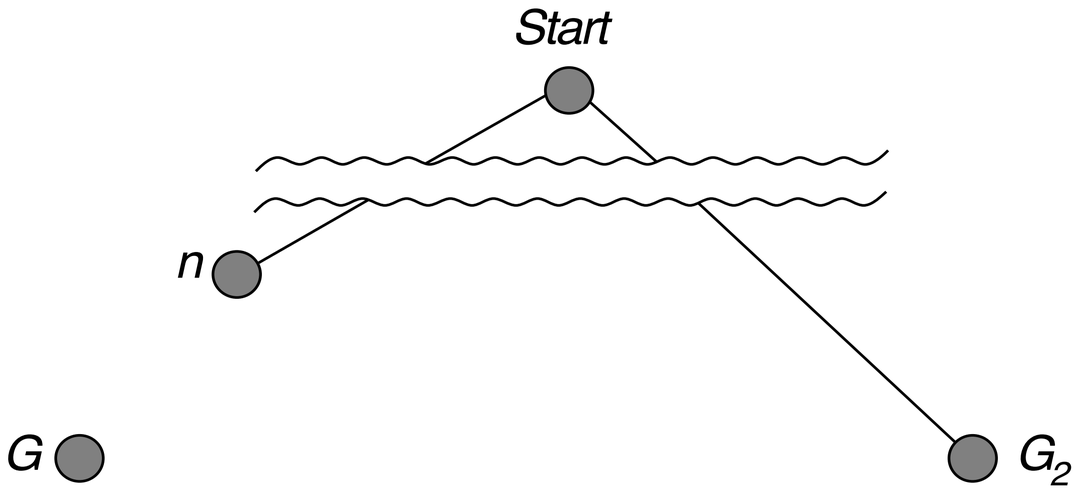
- Theorem: A* Search finds optimal solution if ***h*()** is admissible.

# OPTIMALITY OF A* SEARCH

Suppose a suboptimal goal node *G2* has been generated and is in the queue. Let *n* be the last unexpanded node on a shortest path to an optimal goal node *G*.

**$f(G_2) = g(G2)$ since $h(G_2) = 0$**

**$f(G_2) > g(G)$ since $G_2$ is suboptimal**
**$f(G_2) \geq f(n)$ since h is admissible**

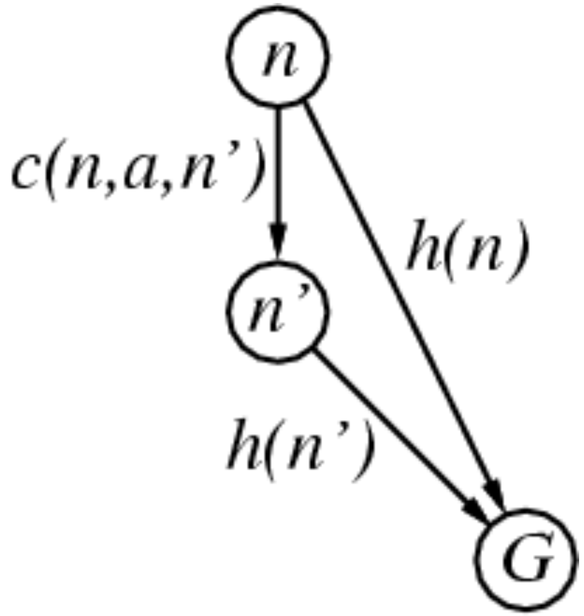


# OPTIMALITY OF A* SEARCH

- Since $f(G2) > f(n)$, A* will never select $G2$ for expansion.
- Note: suboptimal goal node $G2$ may be generated, but it will never be expanded.
- In other words, even after a goal node has been generated, A* will keep searching so long as there is a possibility of finding a shorter solution.
- Once a goal node is selected for expansion, we know it must be optimal, so we can terminate the search.

# CONSISTENT HEURISTICS

- A heuristic is consistent if for every node *n*, every successor *n'* of *n* generated by any action *a*,
  $h(n) \leq c(n,a,n') + h(n')$




If *h* is consistent, we have

$$
\begin{aligned}
f(n') &= g(n') + h(n') \\
&= g(n) + c(n,a,n') + h(n') \\
&\geq g(n) + h(n) \\
&= f(n)
\end{aligned}
$$

Theorem:
If *h(n)* is consistent, A* using `GRAPH-SEARCH` is optimal

# OPTIMALITY OF A* SEARCH

| Algorithm | Completeness | Admissibility | Space | Time |
|---|---|---|---|---|
| **A* Search** | guaranteed | guaranteed if heuristic is admissible/consistent | $O(b^d)$ | $O(b^d)$ |

- Complete: Yes, unless there are infinitely many nodes with f ≤ cost of solution
- Time: Exponential
- Space: Keeps all nodes is memory
- Optimal: Yes (assuming *h* is admissible).