COMPSCI 761: ADVANCED TOPICS IN ARTIFICIAL INTELLIGENCE

INFORMED SEARCH I

Anna Trofimova, July 2022

TODAY

- Last lecture recap
- Heuristic
- Informed Search Algorithms
- Greedy Best-First Search

RECAP: SOME SEARCH PROBLEMS TYPES

Types of Problems (What is returned?)

- A “goal state” (e.g., local search – 8-queens)
- A “move” (e.g., adversarial search – chess)
- A “plan” (e.g., uninformed search – 8-puzzle)

RECAP UNINFORMED SEARCHING FOR A PLAN

Search problem (State Space Graph):
- States (configurations of the world)
- Actions and costs (successor function)
- Start state and goal test

Search tree:
- Nodes: represent states
- Edges: represent actions
- Paths: represent plans of action

Search algorithm:
- Systematically builds a search tree
- Chooses an ordering of the fringe (unexplored nodes)
- Optimal: finds least-cost plans

RECAP: COMPLEXITY RESULTS FOR UNINFORMED SEARCH

Algorithm	Completeness	Admissibility	Space	Time
Breadth-First Search	guaranteed, if b is finite	guaranteed, if arcs have the same cost and >=0	$O(b^d)$	$O(b^d)$
Depth-First Search	not guaranteed	not guaranteed	$O(bm)$	$O(b^m)$
Uniform Cost Search	guaranteed, if b is finite and cost ≥ ε	guaranteed	$O(b^{[1 + C / \varepsilon]})$	$O(b^{[1 + C / \varepsilon]})$
Depth Limited Search	guaranteed, if b is finite	not guaranteed	$O(bk)$	$O(bk)$
Iterative Deepening Search	guaranteed, if b is finite	guaranteed, if arcs have the same cost and >=0	$O(bd)$	$O(b^d)$
Bidirectional Search	depends on search algorithms used	depends on search algorithms used	$O(b^{d/2})$	$O(b^{d/2})$

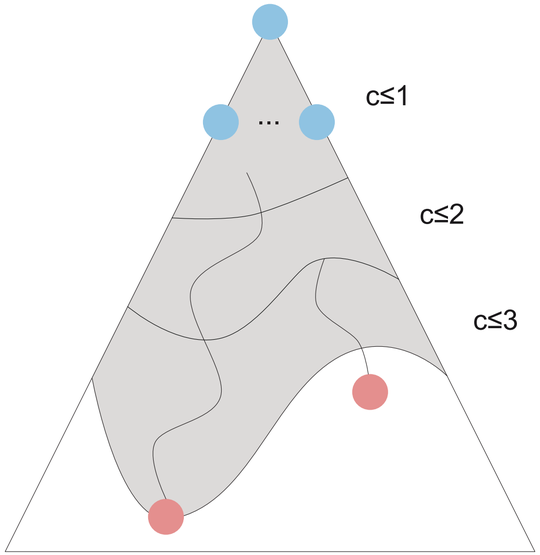
COST SENSITIVE SEARCH

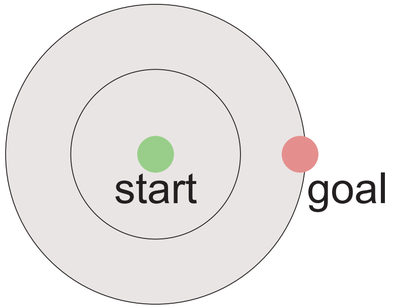
DFS is not optimal.

BFS (Breadth-first search) finds the shortest path in terms of number of actions.It does not find the least-cost path.

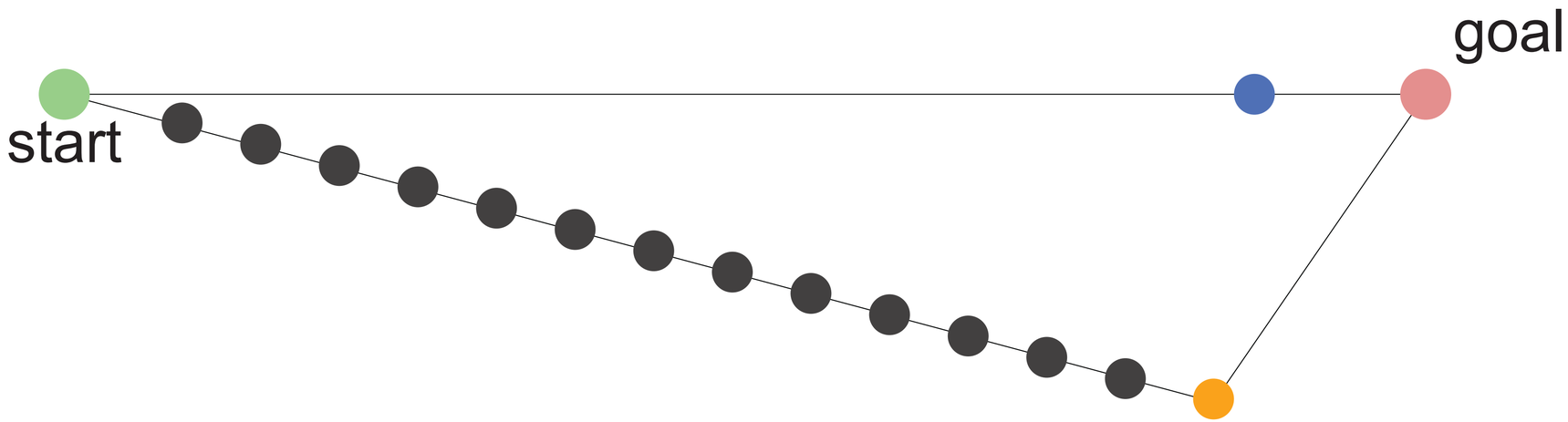
Uniform Cost Search is similar to BFS, but sorts the queue using the path cost.

- Expands the lowest path cost
- The good: UCS is complete and optimal*
- The bad: explores options in every “direction”
- No information about the goal location





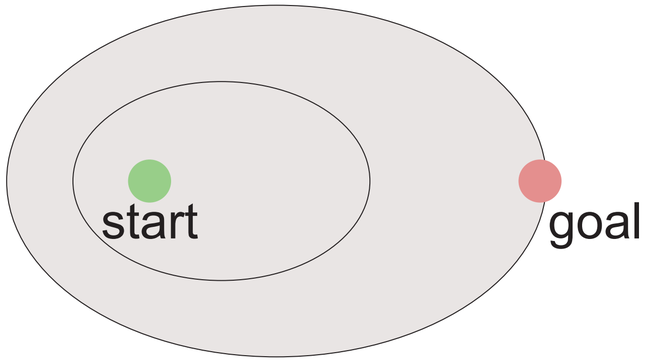
EXAMPLE: THE “BAD” OF UNIFORM COST SEARCH



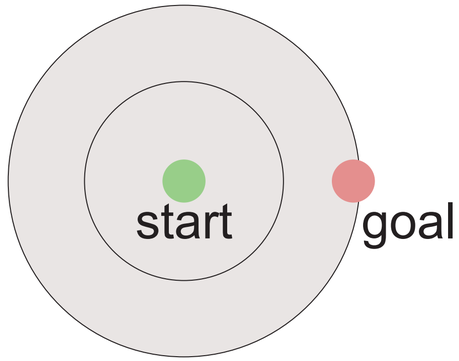
WHAT WE WOULD LIKE TO HAVE

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

Informed



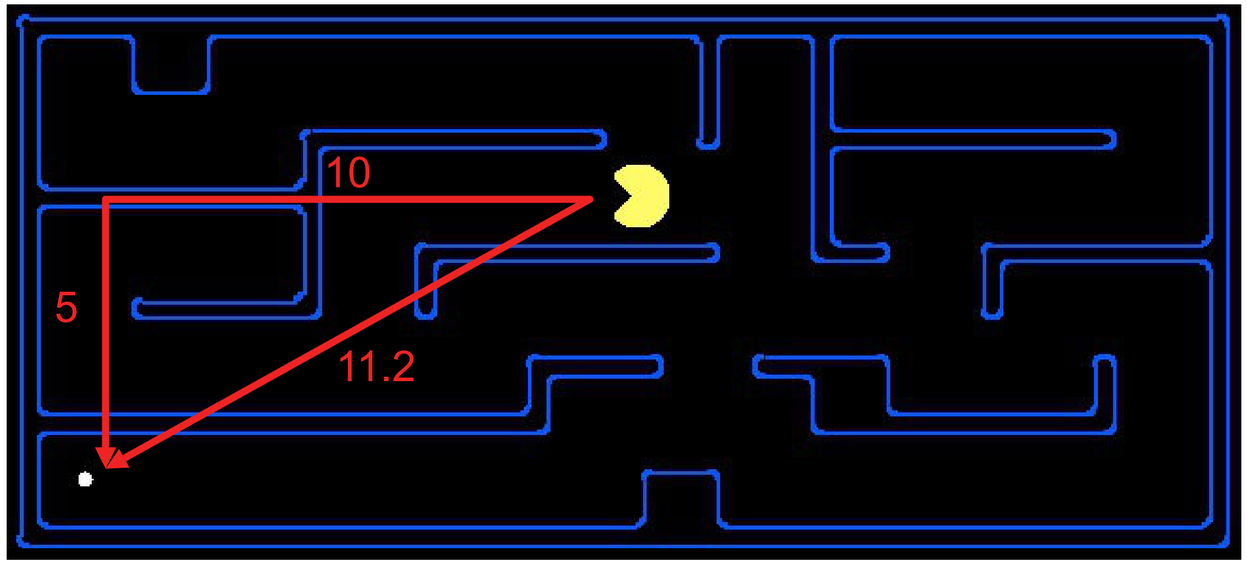
Uninformed



INFORMED SEARCH HEURISTIC

A heuristic is:

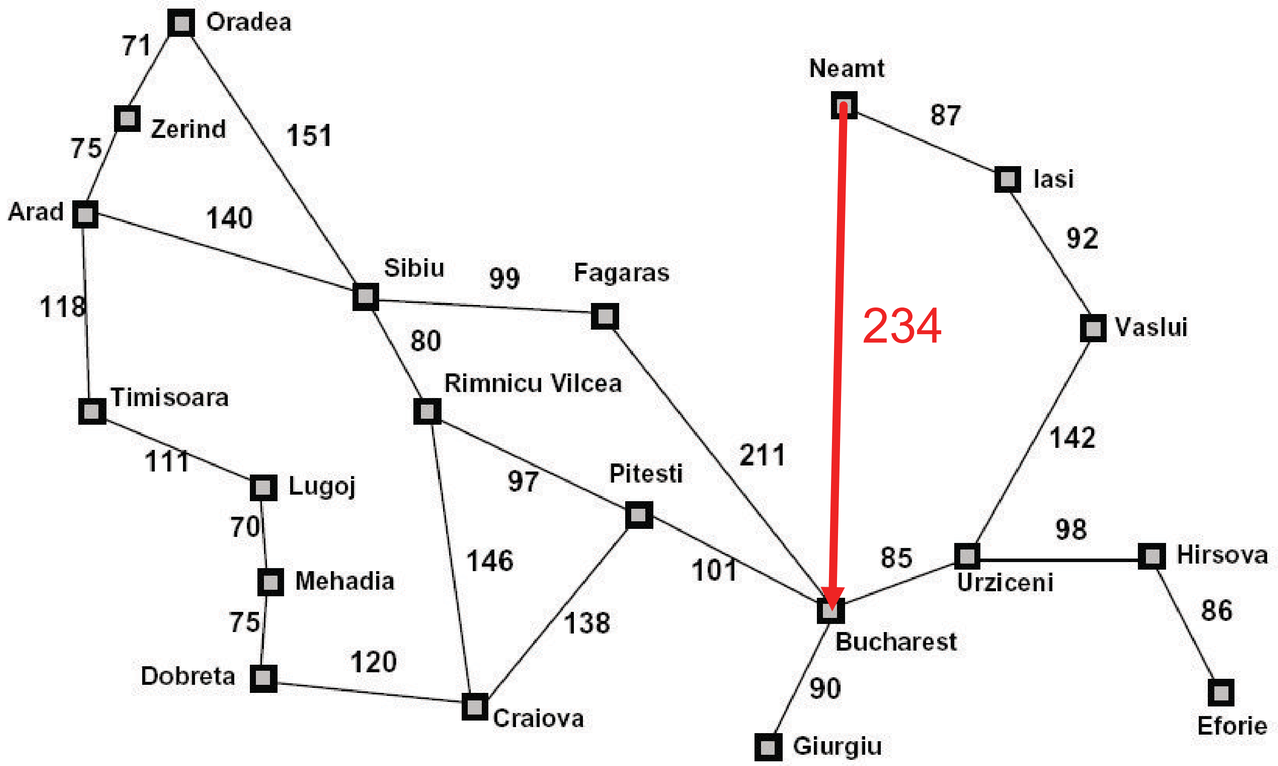
- A function that ***estimates*** how close a state is to a goal
- Designed for a particular search problem
- Examples: Manhattan distance, Euclidean distance for path finding



HEURISTICS

- Heuristics are “rules of thumb”
- Heuristics are criteria, methods or principles for deciding which among several alternative courses of action promises to be the most effective in order to achieve some goal. “Heuristics” (Pearl 1984)
- Can make use of heuristics in deciding which is the most “promising” path to take during search
- In search, heuristic should be an underestimate of actual cost to get from current node to any goal — an admissible heuristic
- Denoted $h(n)$; $h(n)=0$ when ever n is a goal node

EXAMPLE: HEURISTIC FUNCTION



Straight−line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

h(n)

INFORMED (HEURISTICS) SEARCHES

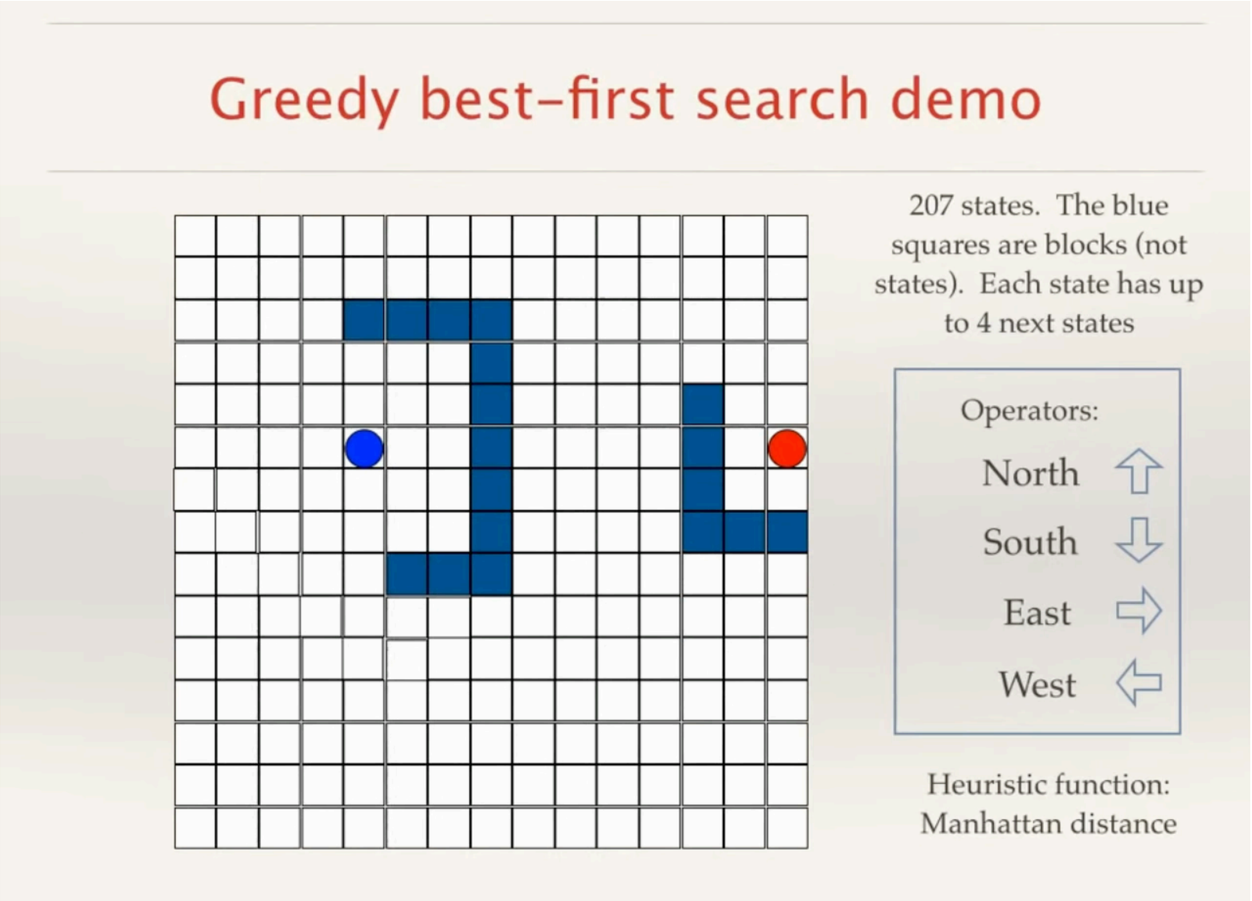
- Uninformed search algorithms—algorithms that are given no information about the problem other than its definition.
  - some of these algorithms can solve any solvable problem, but none of them can do it efficiently
- Informed search strategy
  - one that uses problem-specific knowledge beyond the definition of the problem itself
  - can find solutions more efficiently than uninformed strategy
- Informed search algorithms, can do quite well given some guidance on where to look for solutions.
- All implemented using a priority queue to store frontier nodes

GREEDY [BEST-FIRST] SEARCH

GREEDY BEST-FIRST SEARCH

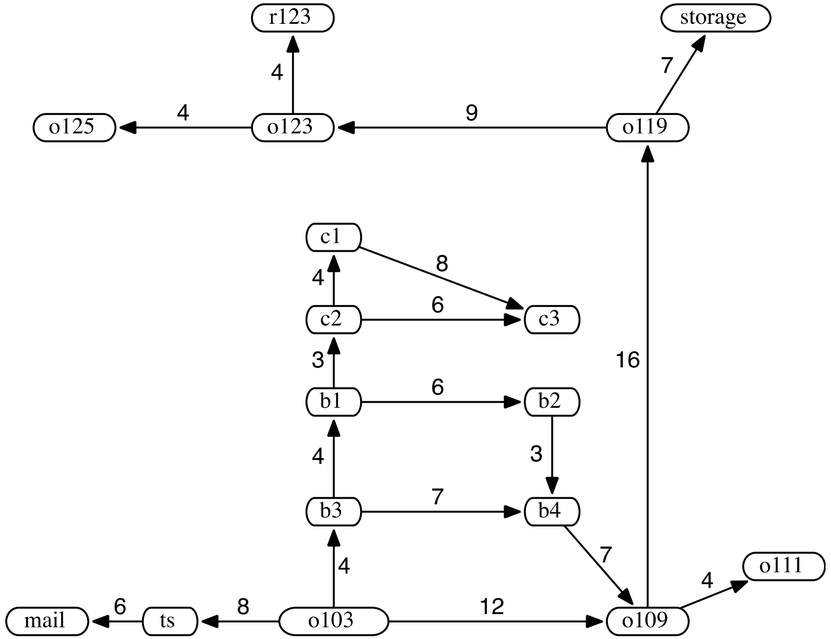
- Idea: select the path whose end is closest to a goal according to the heuristic function.
- Greedy Best-First Search - Best-First Search selects the next node for expansion using the heuristic function for its evaluation function, i.e. $f(n) = h(n)$
  - $h(n)=0 \Rightarrow n$ is a goal state
  - i.e. 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.

GREEDY BEST-FIRST SEARCH DEMO



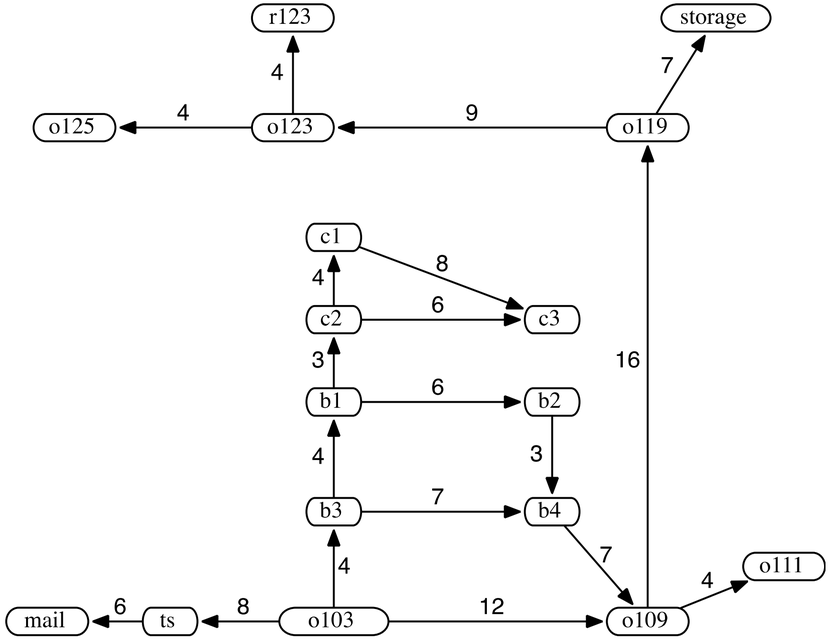
https://www.youtube.com/watch?v=TdHbO3w68fY

EXAMPLES OF GREEDY BEST-FIRST SEARCH

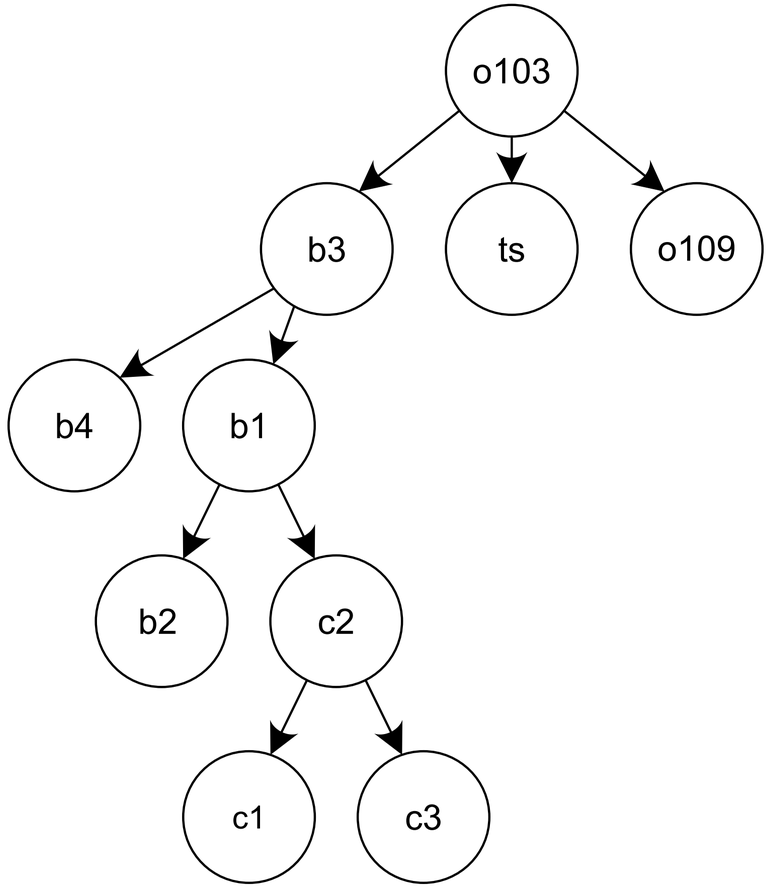


$$
\begin{array}{rcrcrc}
h(mail) & = & 26 & h(ts) & = & 23 \\
h(o109) & = & 24 & h(o111) & = & 27 \\
h(o123) & = & 4 & h(o125) & = & 6 \\
h(b1) & = & 13 & h(b2) & = & 15 \\
h(b4) & = & 18 & h(c1) & = & 6 \\
h(c3) & = & 12 & h(storage) & = & 12
\end{array}
\qquad
\begin{array}{rcrcrc}
h(o103) & = & 21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

EXAMPLES OF GREEDY BEST-FIRST SEARCH



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



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.

SUMMARY

- Heuristics can be applied to reduce search cost.
- Greedy Search tries to minimise cost from current node *n* to the goal.
- Informed search makes use of problem-specific knowledge to guide progress of search