Analyzing Search
Generic search algorithm

add start to frontier
while frontier not empty
    get state from frontier
    if state is goal
        return
    end if
    for neighbor of state
        add neighbor to frontier
    end for
end while
Uninformed Search

Given only the problem definition:
- start state
- goal function
- a way to generate successors

Depth First
- FIFO frontier

Breadth First
- LIFO frontier

Uniform Cost
- frontier ordered by $c(s)$

Informed Search

Given:
- problem definition
- heuristic to estimate cost-to-goal

A*
- frontier ordered by $h(s) + c(s)$

Greedy
- frontier ordered by $h(s)$

$h(s)$: heuristic value of state $s$
$c(s)$: cost to get to state $s$
Measuring Performance

1. **Completeness**: Is the search guaranteed to find a solution (if one exists)?

2. **Optimality**: Is the search guaranteed to find the lowest-cost solution?

3. **Time complexity**: How long does it take to find a solution?
   - How many nodes are expanded?

4. **Space complexity**: How much memory is needed to perform the search?
   - How many nodes get stored in frontier + visited?

\[ b: \text{branching factor} \]
\[ d: \text{depth of the goal} \]
Example Domain

Given a Romanian road map, navigate from Arad to Bucharest.
Frontier
Z=75, T=118, S=140
T=118, S=140, O=146
S=140, O=146, L=229
O=146, R=220, L=229, F=239
R=220, L=229, F=239
L=229, F=239, P=317, C=366
F=239, P=317, M=299, C=366
P=317, M=299, C=366, B=450
M=299, C=366, B=418
C=366, D=374, B=418
D=374, B=418
B=418
Frontier
S=393, T=447, Z=449
R=413, F=417, T=447,
Z=449
P=415, F=417, T=447,
Z=449, C=526
F=417, B=418, T=447,
Z=449, C=526
B=418, T=447,
Z=449, C=526

h(A) = 366
h(C) = 160
h(D) = 242
h(F) = 178
h(L) = 244
h(M) = 241
h(O) = 380
h(P) = 98
h(R) = 193
h(S) = 253
h(T) = 329
h(Z) = 374
Frontier
S=253, T=329, Z=374
F=178, R=193, T=447,
Z=449
B=0, R=193, T=447,
Z=449

Greedy

h(A) = 366
h(C) = 160
h(D) = 242
h(F) = 178
h(L) = 244
h(M) = 241
h(O) = 380
h(P) = 98
h(R) = 193
h(S) = 253
h(T) = 329
h(Z) = 374
<table>
<thead>
<tr>
<th>Method</th>
<th>BFS</th>
<th>DFS</th>
<th>UCS</th>
<th>A*</th>
<th>Greedy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Complete?</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>Optimal?</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>Time efficient?</td>
<td></td>
<td></td>
<td></td>
<td>!!!</td>
<td></td>
</tr>
<tr>
<td>Space efficient?</td>
<td>!!</td>
<td></td>
<td></td>
<td></td>
<td>!!</td>
</tr>
</tbody>
</table>
Devising Heuristics

- Must be **admissible**: never overestimate the cost to reach the goal.
- Should strive for **consistency**: $h(s) + c(s)$ non-decreasing along paths.
- The higher the estimate (subject to admissibility), the better.

Key idea: simplify the problem.

- Traffic Jam: ignore some of the cars.
- Path Finding: assume straight roads.
Clicker Question

Why does A* need an admissible heuristic?

a) required for completeness
b) required for optimality
c) improves time complexity
d) improves space complexity
e) some other reason
A* with an uninformative heuristic (example: \( h(s) = 5, \quad \forall s \)) is equivalent to:

a) breadth first search
b) depth first search
c) uniform cost search
d) greedy search
e) none of these