MAI 5100 — Foundations of Artificial Intelligence
Homework 0 (Part 2)

Solution

Dr Christopher Clarke

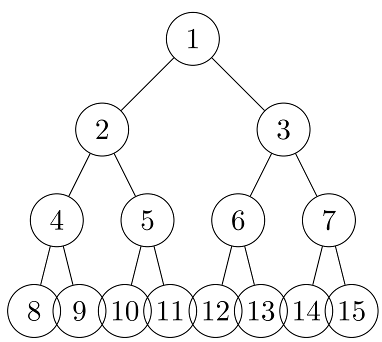
Semester II, 2025

Problem 3.15

Consider a state space where the start state is 1 *and each state* k *has two successors:* $2k$ *(Left) and* $2k+1$ *(Right). Answer parts (i)–(v) for states* 1–15 *with goal state* 11.

Solution

(i) State–space diagram (states 1–15)



(ii) Node–visit orders (goal = 11)

- *Breadth-First Search (BFS):*
  $1, 2, 3, 4, 5, 6, 7, 8, 9, 10, \boxed{11}$
- *Depth-Limited DFS, limit* 3 *(Left before Right):*
  $1, 2, 4, 8, 9, 5, 10, \boxed{11}$
- *Iterative Deepening DFS (limits* $0 \to 1 \to 2 \to 3$*):*

  limit 0: 1

  limit 1: $1, 2, 3$

  limit 2: $1, 2, 4, 5, 3, 6, 7$

  limit 3: $1, 2, 4, 8, 9, 5, 10, \boxed{11}$

  (The final limit discovers the goal at optimal depth.)

(iii) Bidirectional Search

Moving *forward* from the root, each node has two children $\Rightarrow b_f = 2$. Moving *backward* from any node (except the root), there is exactly one parent $\lfloor k/2 \rfloor \Rightarrow b_b = 1$. Because the backward frontier grows linearly, the two frontiers meet after at most $\lceil \log_2 11 \rceil = 4$ levels, so bidirectional search is highly efficient here.

(iv) Reformulation

Each state label encodes its unique path in binary. Writing the goal in binary, dropping the most-significant 1, and reading the remaining bits left-to-right gives the complete action sequence (bit 0=Left, bit 1=Right). Thus the path from 1 to any goal can be obtained with *no search.*

(v) Zero-search algorithm

1. Input a goal state $n \geq 1$.
2. Convert n to binary and discard the leading 1.
3. Scan the remaining bits:
   - bit 0 $\rightarrow$ output Left;
   - bit 1 $\rightarrow$ output Right.

Example for $n = 11$: $11_{10} = 1011_2 \Longrightarrow$ bits "011" $\Rightarrow$ actions L R R. Time complexity $\mathcal{O}(\log n)$, memory $\mathcal{O}(1)$.

Problem 3.27

n labelled vehicles initially occupy the bottom row of an $n \times n$ grid: vehicle i starts in $(i, 1)$ and must be moved to $(n - i + 1, n)$ (top row, reversed order). At every discrete time-step *each* vehicle may

- move one square *Up, Down, Left,* or *Right*, or
- *Stay.* If it stays, at most one adjacent vehicle may *hop* over it.

No two vehicles may occupy the same square.

(a) Size of the state space. Each legal state is a placement of the n *distinct* vehicles on n distinct squares out of the n^2 available cells. Hence

$$\boxed{|S(n)| = \frac{(n^2)!}{(n^2 - n)!}}.$$

(b) Branching factor. For an individual vehicle there are at most 5 distinct actions (the four cardinal moves plus *Stay/Hop*). Assuming no action conflicts, the joint action at a time-step is the Cartesian product of the individual choices, yielding an upper bound

$$\boxed{b(n) = 5^{\,n}}.$$

(The actual factor is lower because some joint moves are illegal when two vehicles target the same square, etc.)

(c) An admissible single–vehicle heuristic. Let vehicle i be at (x, y); its goal is $g_i = (n-i+1, n)$. Define the Manhattan distance

$$d_M(i) = |\, x - (n - i + 1)\, | + |\, y - n\, |.$$

A hop can reduce d_M by *at most two* squares in one time-step, so

$$\boxed{h_i = \left\lceil \frac{d_M(i)}{2} \right\rceil}$$

never overestimates the remaining time and is therefore admissible (and dominates the plain Manhattan distance).

(d) Combined heuristics for the multi-vehicle problem. Let C^* be the true cost (number of time-steps until *all* vehicles reach their goals).

(i) $H_\Sigma = \sum_{i=1}^{n} h_i$

Not admissible. Because the n vehicles move *simultaneously*, their individual costs do *not* add: if two cars each need 3 steps, $C^* = 3$ but $H_\Sigma = 6 > C^*$.

(ii) $H_{\max} = \max_i h_i$

Admissible. The fleet cannot finish before its slowest member, so $H_{\max} \le C^*$ always holds.

(iii) $H_{\min} = \min_i h_i$

Admissible (but weak). It is clearly a lower bound ($H_{\min} \le C^*$) but usually provides very little guidance because it ignores the slower vehicles.

"Manual" Search Tree Generation

Given the highway graph in Figure 1 of the assignment (major cities in the U.S. North-East / Mid-West with edge costs = road mileage), solve

Start = Chicago $\longrightarrow$ **Goal = Boston**

with four search strategies:

(i) Breadth-First Search (BFS)

(ii) Depth-First Search (DFS)

(iii) Uniform-Cost Search (UCS)

(iv) A* Search with an admissible straight-line heuristic $h(n)$

For each method give the *expansion order*, *solution path*, *total mileage* g^*, and comment on *optimality*.

In this solution, children are generated in ALPHABETICAL order.

Figure 1: Highway network supplied in the assignment. Purple edges are labelled with driving mileage and are treated as step-costs.

1 Pre-computed edge lengths

Edge	Miles	Edge	Miles
Chicago → Cleveland	345	Cleveland → Pittsburgh	134
Chicago → Detroit	283	Cleveland → Buffalo	189
Chicago → Indianapolis	182	Columbus → Pittsburgh	185
Detroit → Buffalo	256	Buffalo → Syracuse	150
Cleveland → Columbus	144	Syracuse → Boston	312
Pittsburgh → Buffalo	215	Syracuse → New York	254
Philadelphia → New York	97	Boston → Providence	50
New York → Providence	181	Boston → Portland	107
Baltimore → Philadelphia	101	Pittsburgh → Baltimore	247

2 Breadth-First Search (BFS)

Expansion order

Level 0: Chicago

Level 1: Cleveland, Detroit, Indianapolis

Level 2: Buffalo, Columbus, Pittsburgh

Level 3: Syracuse, Baltimore, Philadelphia

Level 4: $\boxed{\text{Boston}}$

Solution path & cost

Chicago → Cleveland → Buffalo → Syracuse → Boston

$$g^*_{\text{BFS}} = 345 + 189 + 150 + 312 \;=\; \boxed{996\text{ mi}}$$

Optimality BFS is *optimal in number of edges*. Because all solutions have the same depth (4), the first found happens also to be the least mileage, but BFS does *not* guarantee minimal cost in weighted graphs.

---

3 Depth-First Search (DFS)

With the same alphabetical ordering the recursion explores the leftmost branch first.

Expansion order Chicago, Cleveland, Buffalo, Syracuse, Boston.

Solution path & cost (identical to BFS under this ordering)

$$g^*_{\text{DFS}} = 996 \text{ mi}$$

Optimality DFS is neither optimal nor complete in graphs. It *happened* to return an optimal path, but this is coincidence, not a guarantee.

---

4 Uniform-Cost Search (UCS)

UCS always removes the node with smallest path-cost g.

Expansion #	City (n)	$g(n)$ [mi]
1	Chicago	0
2	Indianapolis	182
3	Detroit	283
4	Cleveland	345
5	Columbus	358
6	Pittsburgh	479
7	Buffalo	534
8	Syracuse	684
9	$\boxed{Boston(goal)}$	996

Solution path & cost

$$\text{Chicago} \rightarrow \text{Cleveland} \rightarrow \text{Buffalo} \rightarrow \text{Syracuse} \rightarrow \text{Boston} \qquad g^*_{\text{UCS}} = 996 \text{ mi}$$

Optimality Because every edge cost is positive, UCS is *optimal*; the first time the goal leaves the priority queue its cost is the global minimum.

---

5 A* Search

Heuristic $h(n)$ (straight-line to Boston)

City	$h(n)$ [mi]	City	$h(n)$ [mi]
Chicago	850	Philadelphia	280
Indianapolis	900	New York	200
Detroit	600	Providence	40
Cleveland	550	Portland	120
Columbus	650	Boston	0
Pittsburgh	500	–	–
Buffalo	400	Syracuse	300

The table values are all *lower* than the true driving distances, so h is **admissible** and, because straight-line obeys the triangle inequality, also **consistent**.

Key steps

#	Node n	$g(n)$	$h(n)$	$f(n) = g + h$
1	Chicago	0	850	850
2	Cleveland	345	550	895
3	Buffalo	534	400	934
4	Syracuse	684	300	984
5	$\boxed{\textit{Boston}}$	996	0	996

Solution path & cost Same optimal route:

$$g^*_{A^\star} = 996 \text{ mi}$$

Optimality Because h is admissible and consistent, A* is *both complete and optimal.* Expanding only five nodes shows the benefit of informed search.