assignment 1

these are our solutions to the first assignment of TDT4136.

this document was created using typst.

this is the collaborative effort of group 192, Erlend Ulvund Skaarberg and Fredrik Robertsen

part 1

1)

these are the costs

	A	B	C	D	E	F	G	H	I
A							4		1
B			4					3	2
C		4			1				
D					2	1			
E			1	2				1	
F				1					
G	4								1
H		3			1				
I	1	2					1		

2)

a) BFS

- order of expansion: A, G, I, B, C, H, E, D
- found path: A, I, B, C, E, D, F
- path cost: $1 + 2 + 4 + 1 + 2 + 1 = 11$

b) DFS

- order of expansion: A, G, I, B, C, E, D, F
- found path: A, G, I, B, C, E, D, F

- path cost: $4 + 1 + 2 + 4 + 1 + 2 + 1 = 15$

c) UCS

- order of expansion: A, I, B, H, C, E, D
- found path: A, I, B, H, C, E, D, F
- path cost: $1 + 2 + 3 + 1 + 1 + 2 + 1 = 11$

part 2

3)

a) greedy best first search

- order of expansion: A, I, B, C, E, D
- found path: A, I, B, C, E, D, F
- path cost: $1 + 2 + 4 + 1 + 2 + 1 = 11$

b) A*

- order of expansion: A, I, B, C, E, D
- found path: A, I, B, C, E, D, F
- path cost: $1 + 2 + 4 + 1 + 2 + 1 = 11$

part 3

4)

admissibility is needed for A^* to be optimal.

our heuristics are inadmissible, as they overestimate the cost of the optimal path from each node to the goal. if we look at the graph, the heuristics may encode a sense of distance between each node accurately, however the problem states that the cost of an action is the same between any two neighboring nodes (unless there is a lift or a staircase present). thus, the heuristic estimates are not very optimistic.

5)

the heuristic $h(E) = 5$ is an overestimation, as the shortest path to the goal F is trivially seen to be 3. this is a counter-example to the statement that the heuristic function $h(n)$ is admissible, as admissibility is defined as

$$h(n) \leqslant C^*(n) \quad \forall \quad n \in V$$

where $C^*(n)$ is the cost of the optimal path from the node n.

6)

- order of expansion: A, I, B, H, E, D
- found path: A, I, B, H, E, D, F
- path cost: $1 + 2 + 3 + 1 + 2 + 1 = 10$

7)

the new heuristic function $h'(n)$ is admissible, because it does not overestimate the cost of getting from any node to the goal. thus we can see that when performing A^* in 6) we find the cost-optimal path from A to F.

suppose that there is a node $n \in V$ such that

$$h'(n) > C^*(n).$$

if such a node exists, then $h'(n)$ is inadmissible. since no such node exists, $h'(n)$ is admissible.

admissibility guarantees that A^* finds the cost-optimal path, however it says nothing about optimal efficiency, like consistency (see 8)).

8)

consistency of a heuristic $h(n)$ is defined as a triangle inequality

$$h(n) \leqslant c(n, a, n') + h(n')$$

where $c(n, a, n')$ denotes the cost of the path-action a from n to n'.

to see whether the heuristic $h'(n)$ is consistent, we must investigate each child node of each node on the cost-optimal path to see if this identity holds.

- from A, it is not cheaper to get to I through G, thus the identity holds.
- from B, it is not cheaper to get to H through C, thus the identity holds.
- from H, it is not cheaper to get to E through C, thus the identity holds.

these are all the nodes on the cost-optimal path with multiple children where inconsistency could occur, but these cases are consistent, thus our heuristic $h'(n)$ is consistent.

bonus task

9)

if we let $f(n) = g(n) + h(n)$ in `Best-First-Search(problem, f)` with a heuristic $h(n)$ that is inadmissible for at least one node on the cost-optimal path, unless it is consistent.

however, we can also change the code implementation of A^* to require consistency of $f(n)$:

```
function BEST-FIRST-SEARCH(problem,f) returns a solution node or failure
  node←NODE(STATE=problem.INITIAL)
  frontier←a priority queue ordered by f, with node as an element
  reached←a lookup table, with one entry with key problem.INITIAL and value node
  while not IS-EMPTY(frontier) do
    node←POP(frontier)
    if problem.IS-GOAL(node.STATE) then return node
    for each child in EXPAND(problem, node) do
      s←child.STATE
      if s is not in reached or child.PATH-COST < reached[s].PATH-COST then
        reached[s]←child
        add child to frontier
        // ------------
        // calculate all parts of triangle inequality
        h←f(n) - node.PATH-COST
        h'←f(n) - child.PATH-COST
        c←child.PATH-COST - node.PATH-COST
        if h > c + h' then return failure  // fail on inconsistency
        // ------------
  return failure
```