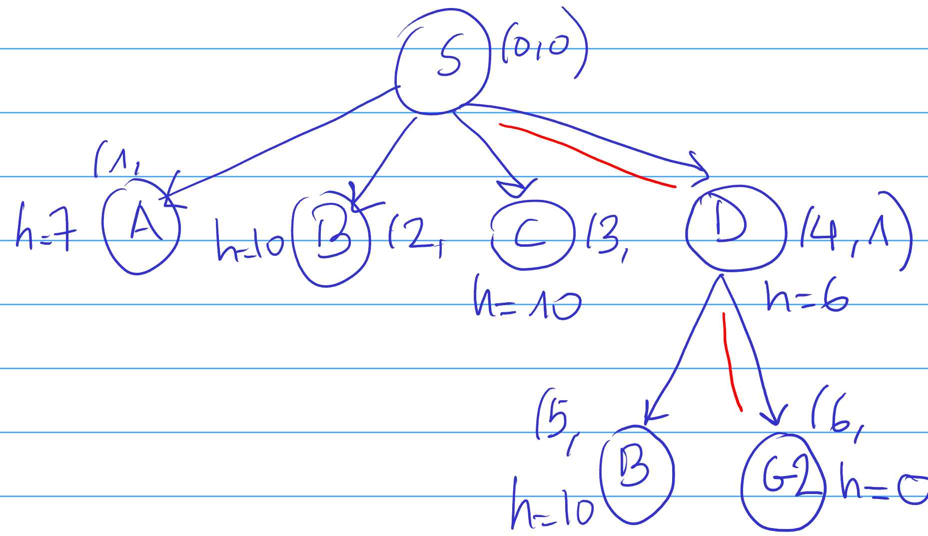
SOLUTION PARTIAL 1 2015 IA BIL

PROB 1

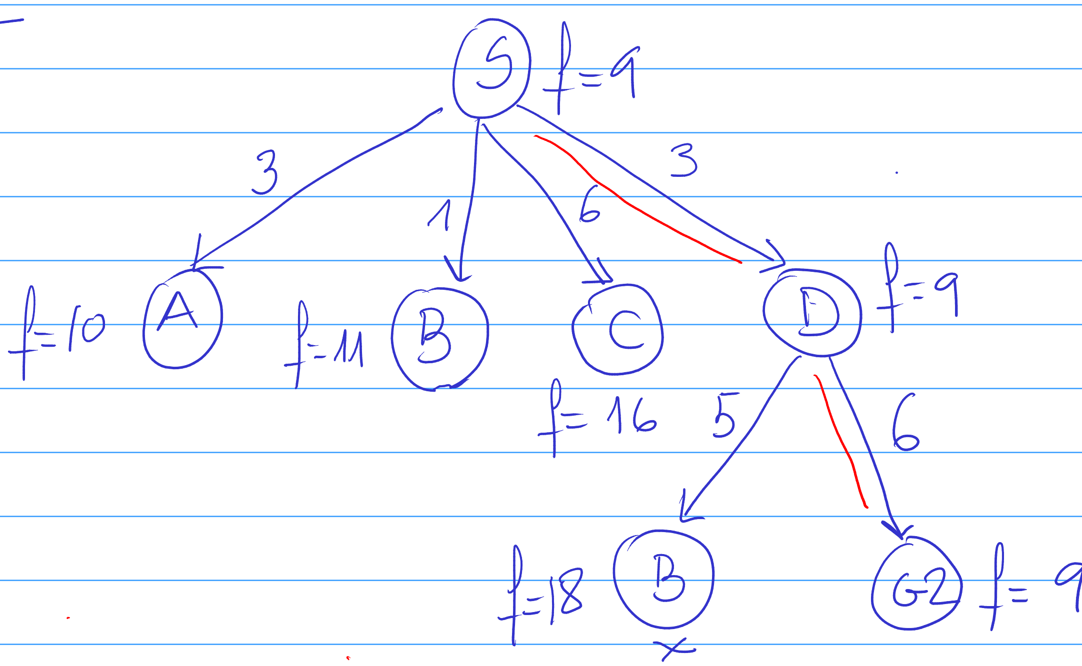
HC




SOLUTION: S-D-G2

COST: 9

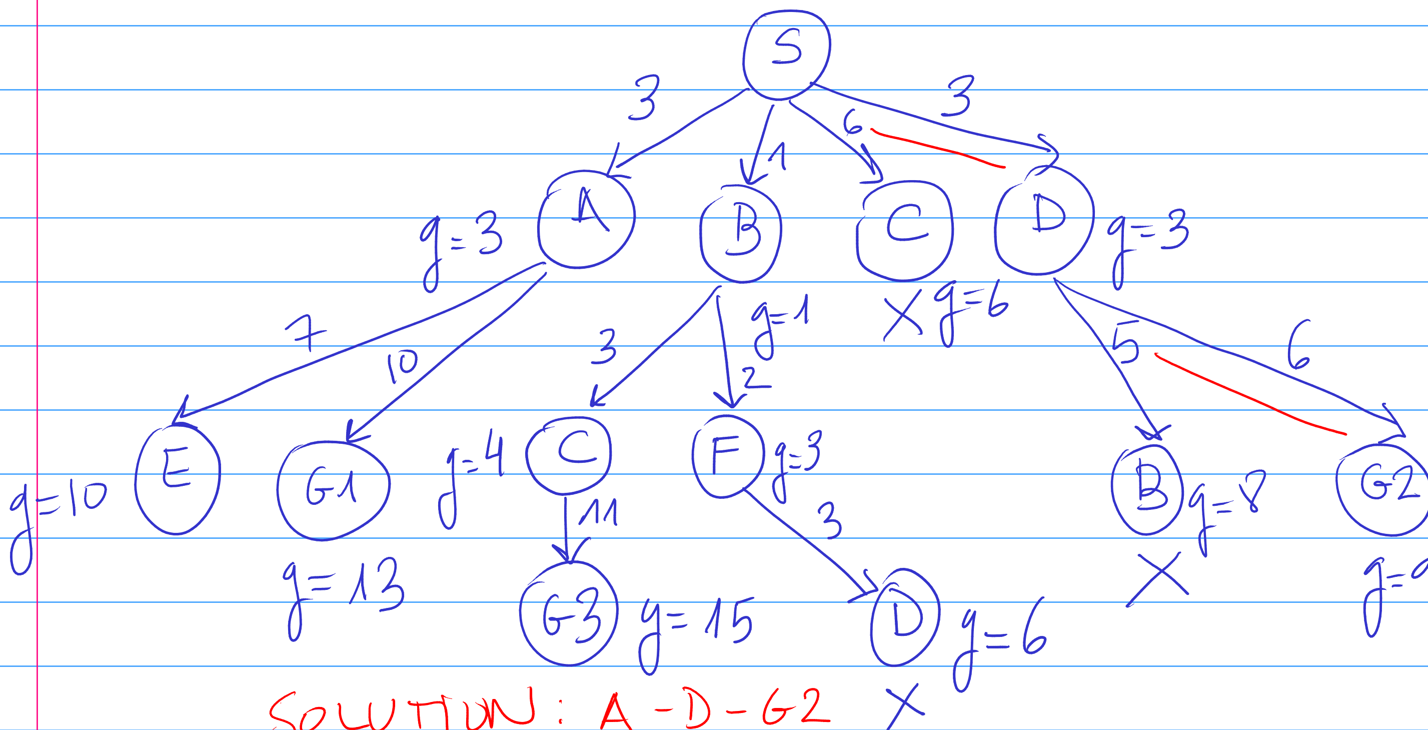
A*




SOLUTION: S-D-G2

COST: 9

## DIJKSTRA




SOLUTION : A - D - G2
COST : 9

## QUESTION 1

The 3 algorithms find the optimal sol. Dijkstra always guarantees it. It is a particular case of A* with $h=\phi$. A* guarantees the optimal sol. with an admissible heuristic. In this exercise $h(n)$ is admissible since it never overestimate the real optimal cost. HC is neither complete nor admissible, but for this specific problem the heuristic guides well to the optimal sol.

## QUESTION 2

A* because it is admissible; and it is expected to expand less nodes than Dikjstra

PROB2

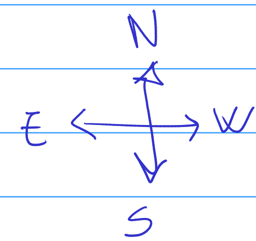
1. States $\quad S = (\overset{S_0}{X}, \overset{S_1}{Y}, \overset{S_2}{D}, \overset{S_3}{V})$ VECTOR

$X \in \{1,2,3,4,5\}$ XPOS
$Y \in \{1,2,3,4,5\}$ YPOS
$D \in \{N,S,E,W\}$ DIRECTION
$V \in [0, V_{Max}\}$ VELOCITY

2. $EI = (5, 1, N, 0)$

$GOAL = (1, 1, D, V)$ $\quad D \in \{N,S,E,W\}$
$V \in [0, V_{MAX}]$

3. OPERATORS $\quad O(S) = S'$




- CHANGE_DIRECTION+

Applicability: $S_3 == \emptyset$
Result: $S'_i = S_i$ for $i \in \{0,1,3\}$
$S'_2 = Rotate\_90(S_2)$
↳ this is a fuction that rotate the dir by +90 degrees. For instance, for N returns W.

Cost = 1

- CHANGE_DIRECTION− : as the previous one but in the result $S_2' = Rotate\_{-90}(S_2)$
↳ Rotate the dir by −90 degrees

• ADVANCE ($\Delta$) $\Delta \in \{1, -1, 0\}$

Applicable when ($S_3 \neq 0$ OR $\Delta \neq 0$)
AND $S_3 + \Delta \leq V_{MAX}$ AND

* IF $S_2 = N$ /* North */
Applicable when
$\forall y \in \{0, \ldots S_3 - 1\}$ $wall(S_0, S_1 - y, N)$
is FALSE

We can assume a boolean function $wall(x, y, D)$: true when a cell has a wall in direction D
It will be also true if there is no cell to check limits

Result:
$$\begin{cases} S'_0 = S_0 \\ S'_1 = S_1 - S_3 \\ S'_2 = S_2 \\ S'_3 = S_3 + \Delta \end{cases}$$

Cost: 1

* IF $S_2 = W$ : Applicable: $\forall x \in \{0, \ldots, S_3 - 1\}$
$wall(S_0 + x, S_1, W) == FALSE$

Result: $S'_0 = S_0 + S_3$ ; $S'_1 = S_1$ , $S'_2 = S_2$
$S'_3 = S_3 + \Delta$

Cost: 1

* IF $S_2 = S$ : Applicable. $\forall y \in \{0, \ldots, S_3 - 1\}$
$wall(S_0, S_1 + y, S) == FALSE$

Result: $S'_0 = S_0$ , $S'_1 = S_1 + S_3$ ; $S'_2 = S_2$ ; $S'_3 = S_3 + \Delta$

Cost: 1

* IF $S_2 = E$ : Applicable: $\forall x \in \{0, \ldots, S_3 - 1\}$
$wall(S_0 - x, S_1, E) == FALSE$

Result: $S'_0 = S_0 - S_3$ ; $S'_1 = S_1$ ; $S'_2 = S_2$
$S'_3 = S_3 + \Delta$

Cost: 1

4. Breadth-First-Search (or Dikjstra since costs are unitary) and A* with admissible h

5. An easy admissible heuristic is $h(n) = \frac{\text{Manhattan}(n,g)}{V_{MAX}}$

PROB 3 (there are different valid solutions to this problem)

1)

- car(X, Y, D, V) → position, dir, and car velocity
  $X \in \{1..5\}$ ; $Y \in \{1..5\}$
  $D \in \{N, S, E, W\}$
  $V \in [0, V_{MAX}]$

- max_advance(X, Y, D, N) : $x, y \in \{1..5\}$ $D \in \{N, S, E, W\}$
  $N \in [0, V_{MAX}]$
  ↳ maximum number of cells the car can advance from x, y in direction D considering walls and limits

- direction_change(X, Y) : $X, Y \in \{N, S, W, E\}$
  allowed direction changes

- movement(D, ΔX, ΔY) → $\Delta x, \Delta y \in \{0, 1, -1\}$
  $D \in \{N, S, E, W\}$
  allowed movements

- vel_increment(Δ) → $\Delta \in \{0, 1, -1\}$
  allowed increments of velocity

$MT_0 = \{ car(5,1,N,\phi)$, max_advance$(5,1,N,0)$, max-advance$(5,2,N,2)$,
..., direction_change $(N,E)$, direction_change $(N,W)$ ....,
movement$(N,-1,0)$, movement$(W,0,+1)$ ....,
vel-increment$(0)$, vel-increment$(1)$, vel-increment$(-1)\}$

2) With rules. With the proposed representation we need just one rule to advance

R ADVANCE:
IF car$(X,Y,D,V)$ AND MOVEMENT$(D,\Delta X,\Delta Y)$
AND VEL_INCREMENT$(\Delta V)$ AND
max_advance$(X,Y,D,N)$ AND $V \leq N$ AND
$V + \Delta V > 0$
THEN not car$(X,Y,D,V)$
car$(X+(\Delta x \times v), Y+(\Delta Y \times v), D, v+\Delta v)$

PROB 4

$WM_0 = \{a(1,1), a(2,1), a(3,2), a(4,2), b(1,2), b(2,1)\}$
$CC_0 = \{R1(x=1, y=1, z=1, y1=2, z1=2)$,
$R1(x=2, y=1, z=2, y1=2, z1=1)$,
$\underline{R1(x=3, y=2, z=2, y1=1, z1=1)}\}$

$WM_1 = \{a(1,1), a(2,1), a(3,1), a(4,1), b(1,2), b(2,1)\}$

$CC_1 = \{R1(X=1, Y=1, Z=1, Y1=2, Z1=2)$
$R1(X=2, Y=1, Z=1, Y1=2, Z1=2)$
$R1(X=3, Y=1, Z=1, Y1=2, Z1=2)$
$\underline{R2(X=1)}\}$

stop_exec()