Course "**Fundamentals of Artificial Intelligence**"
EXAM TEXT

Prof. Roberto Sebastiani
DISI, Università di Trento, Italy

2026.02.19

822917

[COPY WITH SOLUTIONS]

1

Given the following symbols, representing concept, relation and individual names in the alien language of the remote planet Sgotz:

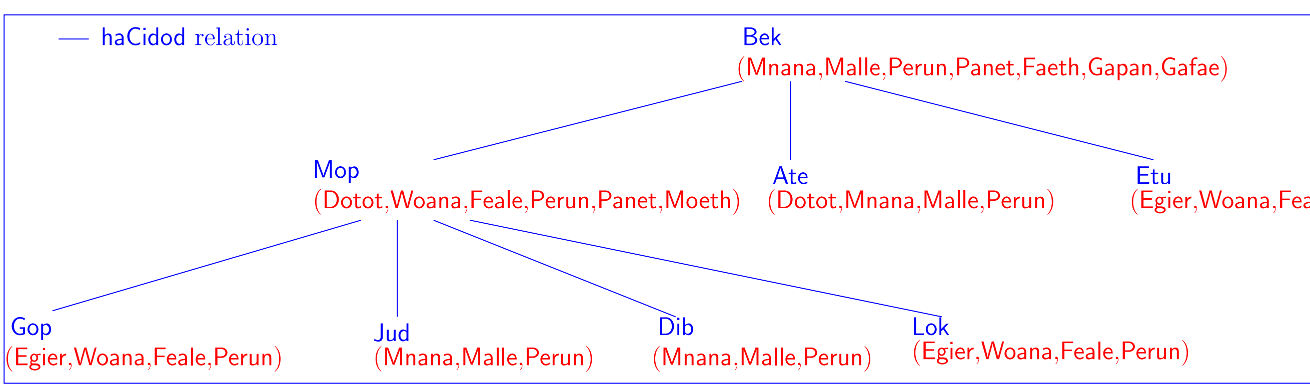
- a set of primitive $\mathcal{ALCQ}$ concept names: {Perun, Malle, Feale, Dotot, Egier}
- a set of $\mathcal{ALCQ}$ relation names: {haCidod}
- a set of $\mathcal{ALCQ}$ individual names: {Lok, Bek, Mop, Dib, Ate, Jud, Etu, Gop}

and the following $\mathcal{ALCQ}$ $\mathcal{T}$-box $\mathcal{T}$ and $\mathcal{A}$-box $\mathcal{A}$:

$\mathcal{T}$			$\mathcal{A}$
Perun		⟨*primitive concept*⟩	Mop : Woana; Etu : Woana; Gop : Woana; Lok : Woana;
Feale		⟨*primitive concept*⟩	
Malle		⟨*primitive concept*⟩	Bek : Mnana; Ate : Mnana; Jud : Mnana; Dib : Mnana;
Dotot		⟨*primitive concept*⟩	
Egier		⟨*primitive concept*⟩	Ate : Dotot; Mop : Dotot
Woana	≡	Perun ⊓ Feale	
Mnana	≡	Perun ⊓ Malle	Etu : Egier; Gop : Egier; Lok : Egier
Moeth	≡	Woana ⊓ ∃haCidod.Perun	
Faeth	≡	Mnana ⊓ ∃haCidod.Perun	⟨Bek, Mop⟩ : haCidod; ⟨Bek, Ate⟩ : haCidod;
Panet	≡	Perun ⊓ haCidod.Perun	⟨Bek, Etu⟩ : haCidod;
Gamae	≡	Moeth ⊓ haCidod.Panet	
Gafae	≡	Faeth ⊓ haCidod.Panet	⟨Mop, Gop⟩ : haCidod; ⟨Mop, Jud⟩ : haCidod;
Gapan	≡	Panet ⊓ haCidod.Panet	⟨Mop, Dib⟩ : haCidod; ⟨Mop, Lok⟩ : haCidod;

For each of the following $\mathcal{ALCQ}$ queries to $\mathcal{T} \cup \mathcal{A}$, say if it is true or false.

(a) Mop : Moeth ⊓ (≥ 3)haCidod.Egier [Solution: false]
(b) Bek : Faeth ⊓ (≥ 2)haCidod.Dotot [Solution: true]
(c) Bek : Faeth ⊓ ∃haCidod.Panet [Solution: true]
(d) Bek : Gapan ⊓ ∀haCidod.Moeth [Solution: false]

[Solution:


]

2

For each of the following facts about conditional independence, say if it is true or false.

(a) If D and E are conditionally independent given F, then $\mathbf{P}(D,F|E) = \mathbf{P}(D,F)$
[Solution: false: $\mathbf{P}(D,F|E) = \mathbf{P}(D|F,E)\mathbf{P}(F|E) \neq \mathbf{P}(D|F,E)\mathbf{P}(F) = \mathbf{P}(D|F)\mathbf{P}(F) = \mathbf{P}(D,F)$]

(b) If D and E are conditionally independent given F, then $\mathbf{P}(D,E|F) = \mathbf{P}(D,E)$
[Solution: false: $\mathbf{P}(D,E|F) = \mathbf{P}(D|F)\mathbf{P}(E|F) \neq \mathbf{P}(D,E)$]

(c) If D and E are conditionally independent given F, then $\mathbf{P}(D,E,F) = \mathbf{P}(D)\mathbf{P}(E)\mathbf{P}(F)$
[Solution: false: $\mathbf{P}(D,E,F) = \mathbf{P}(D,E|F)\mathbf{P}(F) = \mathbf{P}(D|F)\mathbf{P}(E|F)\mathbf{P}(F) \neq \mathbf{P}(D)\mathbf{P}(E)\mathbf{P}(F)$]

(d) If D and E are conditionally independent given F, then $\mathbf{P}(D,E,F) = \mathbf{P}(D|F)\mathbf{P}(E|F)\mathbf{P}(F)$
[Solution: true, $\mathbf{P}(D,E,F) = \mathbf{P}(D,E|F)\mathbf{P}(F) = \mathbf{P}(D|F)\mathbf{P}(E|F)\mathbf{P}(F)$]

3

Given a generical search problem, assume time and space complexity are measured in terms of

b : maximum branching factor of the search tree

m : maximum depth of the state space (assume m is finite)

d : depth of the shallowest solution

Assume also that all steps cost are 1.

For each of the following facts, say if it is true or false

(a) Breadth-First Search requires $O(b^m)$ memory to find a solution.
[Solution: false, requires $O(b^d)$ memory]

(b) Breadth-First Search is optimal
[Solution: true]

(c) Depth-First Search with loop-prevention requires $O(b^d)$ steps to find a solution.
[Solution: false, requires $O(b^m)$ steps]

(d) Depth-First Search with loop-prevention requires $O(b^d)$ memory to find a solution.
[Solution: false, requires $O(bm)$ memory]

4

Consider propositional logic (PL); let F, C, D, E, B, A, G be atomic propositions We adopt the set notation for resolution rules, s.t. Γ denotes a set of clauses.

For each of the following statements, say if it is true or false.

(a) The following is a correct application of the PL unit-resolution rule:

$$\frac{\Gamma, \quad (\ D), \quad (\ A \vee \neg D \vee \ F)}{\Gamma, \quad (\ D), \quad (\ A \vee \ F)}$$

[Solution: true]

(b) The following is a correct application of the PL clause-subsumption rule:

$$\frac{\Gamma, \quad (\ A \vee \ F), \quad (\ A \vee \neg D \vee \ F)}{\Gamma, \quad (\ A \vee \neg D \vee \ F)}$$

[Solution: false]

(c) The following is a correct application of the PL general resolution rule:

$$\frac{\Gamma, \quad (\ D \vee \neg E \vee \neg A), \quad (\neg F \vee \neg D \vee \neg E)}{\Gamma, \quad (\neg E \vee \neg F \vee \neg A)}$$

[Solution: true]

(d) The following is a correct application of the PL general resolution rule:

$$\frac{\Gamma, \quad (\ A \vee \neg D \vee \ F), \quad (\neg A \vee \neg F \vee \ E)}{\Gamma, \quad (\neg D \vee \ E)}$$

[Solution: False]

5

In the following FOL formulas, let R, P, Q, and $>$, $\leq$, $<$, $\geq$ denote predicates,
h, f, g, F_1, F_2, F_3 and $+, -, \cdot, /$ denote functions,
x, y, z, x_1, x_2, x_3 denote variables,
A, B, C, C_1, C_2, C_3 and $0, 1, 2, 3, 4$ denote constants.

For each of the following facts, say if it is true or false.

(*a*) The FOL formula $(\forall x_2 \exists x_1 . R(x_1, x_2)) \rightarrow (\exists x_1 \forall x_2 . R(x_1, x_2))$ is valid.
[Solution: false]

(*b*) The FOL formula $(\exists x_1 . \neg R(x_1)) \leftrightarrow (\neg \forall x_1 . R(x_1))$ is valid
[Solution: true]

(*c*) The FOL formula $\forall x.((x > 2) \rightarrow (x > 1))$ is valid.
[Solution: false (in FOL ">", "2", "1" have no fixed interpretation)]

(*d*) The FOL formula $(2 > 4)$ is unsatisfiable.
[Solution: false (in FOL ">", "2", "4" have no fixed interpretation)]

6

Consider the following CNF formula in PL:

$$\begin{array}{llll}(\neg E & \vee\ B & \vee\ N) & \wedge \\ (\ A & \vee\ H & \vee\ C) & \wedge \\ (\neg H & \vee\ I & \vee\ A) & \wedge \\ (\neg L & \vee\ C & \vee \neg M) & \wedge \\ (\neg G & \vee \neg A & \vee\ E) & \wedge \\ (\neg E & \vee \neg G & \vee\ A) & \wedge \\ (\neg E & \vee \neg F & \vee \neg A) & \wedge \\ (\ I & \vee\ L & \vee\ M) & \wedge \\ (\neg N & \vee\ L & \vee\ M) & \end{array}$$

Consider the WalkSAT algorithm, with probability parameter $p = 0.2$. Suppose at a given step the current assignment is

$$\{\ A, \neg B,\ \ C,\ \ D,\ \ E, \neg F,\ \ G, \neg H,\ \ I, \neg L, \neg M, \neg N\}.$$

Assuming the most-likely event happens, describe what the assignment is after the next step.

[Solution:

The current assignments makes only the 1^{st} clause unsatisfied. Since $p = 0.2$, the most likely event is that the algorithm flips the symbol in the first clause which maximizes the number of satisfied clause at next step.

We notice that flipping $\neg B$ would cause no other clause to become unsatisfied, whereas flipping either of $E, \neg N$ would cause other clauses to become unsatisfied. Thus WalkSAT flips $\neg B$, obtaining the assignment

$$\{\ A,\ \ B,\ \ C,\ \ D,\ \ E, \neg F,\ \ G, \neg H,\ \ I, \neg L, \neg M, \neg N\}.$$

which satisfies all clauses.]

7

An experienced doctor has to cope with an epidemic of covid19, where 60% of people of the area have been infected. She considers the following possible symptoms:

Symptom #1: headache;
Symptom #2: nausea;
Symptom #3: fever.

She models the cause-effect relation as a **Naive Bayes Model scenario**, s.t the effects are considered conditionally independent given the cause, and she knows from statistics the following data: [1]

$P(headache \mid covid19)$	$= 0.8$
$P(headache \mid \neg covid19)$	$= 0.1$
$P(nausea \mid covid19)$	$= 0.4$
$P(nausea \mid \neg covid19)$	$= 0.2$
$P(fever \mid covid19)$	$= 0.6$
$P(fever \mid \neg covid19)$	$= 0.3$

She is informed that one patient has nausea and fever but not headache. Compute the probability that such patient has contracted covid19.

Notice: *the problem must be solved my using the Naive Bayes Model scenario. Any attempt to use any other technique will be considered incorrect.*

[Solution: With a **Naive Bayes Model** scenario, we have that:

$$\mathbf{P}(Cause|Effect_1, ..., Effect_n) = \alpha\mathbf{P}(Cause) * \prod_{i=1}^{n} \mathbf{P}(Effect_i|Cause)).$$

for some normalization constant α. Thus, we have:

$$\begin{array}{rl} & P(\ covid19|\neg headache \wedge nausea \wedge fever) \\ = & \alpha * P(\ covid19) * P(\neg headache|\ covid19) * P(\ nausea|\ covid19) * P(\ fever|\ covid19) \\ = & \alpha * 0.6 * (1 - 0.8) * 0.4 * 0.6 \\ = & \alpha * 0.0288 \\ & P(\neg covid19|\neg headache \wedge nausea \wedge fever) \\ = & \alpha * P(\neg covid19) * P(\neg headache|\neg covid19) * P(\ nausea|\neg covid19) * P(\ fever|\neg covid19) \\ = & \alpha * (1 - 0.6) * (1 - 0.1) * 0.2 * 0.3 \\ = & \alpha * 0.0216 \end{array}$$

Thus, after normalization:

$$\begin{array}{rl} P(\ covid19|\neg headache \wedge nausea \wedge fever) = & 0.5714285714285714 \\ P(\neg covid19|\neg headache \wedge nausea \wedge fever) = & 0.4285714285714286 \end{array}$$

]

[1] The data here are pure fantasy and are not supposed to correspond to actual medical data.

8

(*a*) Describe as Pseudo-Code the Depth-Limited Search and Iterative-Deepening procedures.

[Solution:

```
function DEPTH-LIMITED-SEARCH(problem, limit) returns a solution, or failure/cutoff
  return RECURSIVE-DLS(MAKE-NODE(problem.INITIAL-STATE), problem, limit)

function RECURSIVE-DLS(node, problem, limit) returns a solution, or failure/cutoff
  if problem.GOAL-TEST(node.STATE) then return SOLUTION(node)
  else if limit = 0 then return cutoff
  else
      cutoff_occurred? ← false
      for each action in problem.ACTIONS(node.STATE) do
          child ← CHILD-NODE(problem, node, action)
          result ← RECURSIVE-DLS(child, problem, limit − 1)
          if result = cutoff then cutoff_occurred? ← true
          else if result ≠ failure then return result
      if cutoff_occurred? then return cutoff else return failure

function ITERATIVE-DEEPENING-SEARCH(problem) returns a solution, or failure
   for depth = 0 to ∞ do
      result ← DEPTH-LIMITED-SEARCH(problem, depth)
      if result ≠ cutoff then return result
```

or any schema equivalent to the above one (from AIMA book).]

(*b*) calling B the branching factor and D the depth of the shallowest solution,

- what is the time complexity of the Iterative-Deepening procedure? [Solution: $O(B^D)$]
- what is the memory complexity of the Iterative-Deepening procedure? [Solution: $O(BD)$]

9

(a) Describe as Pseudo-Code the specialized solving procedure for tree-structured CSPs.
[Solution:

function TREE-CSP-SOLVER(*csp*) **returns** a solution, or failure
  inputs: *csp*, a CSP with components X, D, C

  $n \leftarrow$ number of variables in X
  $assignment \leftarrow$ an empty assignment
  $root \leftarrow$ any variable in X
  $X \leftarrow$ TOPOLOGICALSORT($X, root$)
  for $j = n$ **down to** 2 **do**
    MAKE-ARC-CONSISTENT(PARENT(X_j), X_j)
    if it cannot be made consistent **then return** *failure*
  for $i = 1$ **to** n **do**
    $assignment[X_i] \leftarrow$ any consistent value from D_i
    if there is no consistent value **then return** *failure*
  return *assignment*

or any schema equivalent to the above one (from AIMA book).]

(b) Say if the following sentence is true or false, and briefly explain why.

- It requires polynomial time in worst-case if the input constraint graph has no loops.

[Solution: True. It requires $O(nd^2)$ steps in worst case.]

10

Given the following Sudoku scenario:

1	2	3	4	5	6	7	8	9	
1		8							A
2				6					B
3						4	5		C
8								5	D
7								4	E
6								9	F
5	8	1							G
	7	2		9					H
		3							I

(a) Apply the AC-3 algorithm. Describe in the right sequence the domains of unassigned nodes whose domains become unary after one run of AC-3. (E.g.:
$D_{A1} := \{3\}$,
$D_{B1} := \{7\}$,
...)

(b) Can AC-3 reduce to unary the domains of nodes $B3$ and $C3$?

(c) After one run of AC-3, is the resulting graph arc-consistent?

[Solution:

(a) We have, in sequence
$D_{H1} := \{4\}$,
$D_{I1} := \{9\}$,
$D_{I2} := \{6\}$,
$D_{C2} := \{9\}$

(b) No. Decent Sudoku players can infer also $A2 = B2 = \{5, 4\}$, so that, by path-consistency (not arc-consistency!), we have $B3 := \{7\}$, $C3 := \{6\}$, but this cannot be performed by AC-3.

(c) Yes. AC-3 always makes a graph arc-consistent.

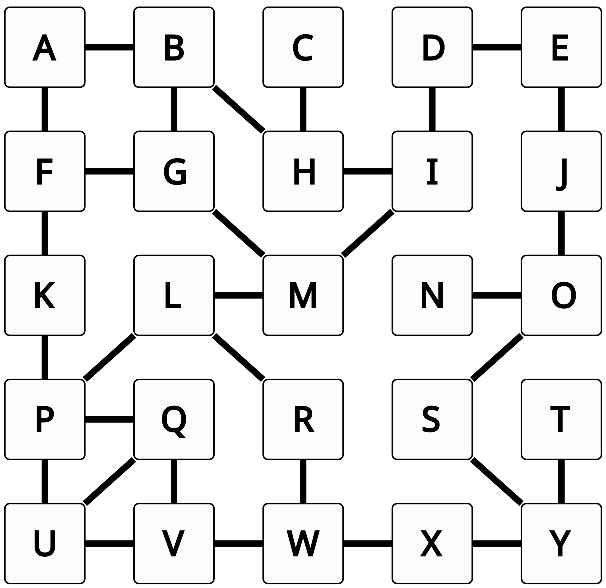
]

11

In the following state graph, apply *breadth-first search* and report for each step:

- the node extracted from the frontier;
- the nodes added to the frontier in the current step.

Actions are sorted according to the (ascending) alphabetical order of the destination.



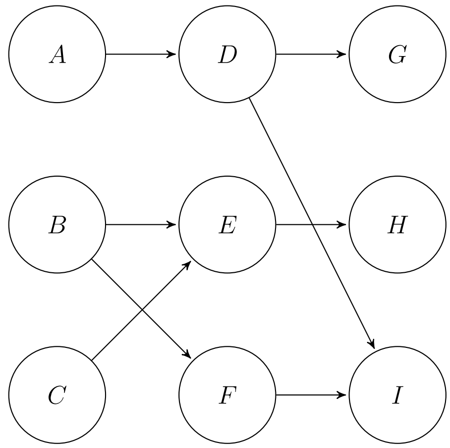
Start: **M** - Goal: **X**

[Solution:
M - [G, I, L]
G - [B, F]
I - [D, H]
L - [P, R]
B - [A]
F - [K]
D - [E]
H - [C]
P - [Q, U]
R - [W]
A - []
K - []
E - [J]
C - []
Q - [V]
U - []
W - stop
]

12

Consider the following actions with durations / dependencies:

a	duration(a)
A	1
B	2
C	1
D	2
E	2
F	3
G	3
H	1
I	2



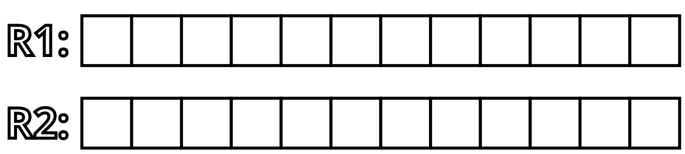
1) Compute the earliest / latest possible start time (ES/LS) for each action using the Critical Path method. [40pts.]

Assume that completing each action requires one unit of a reusable resource:

2) Report an optimal schedule given 2 reusable resources in the grid below. [40pts.]

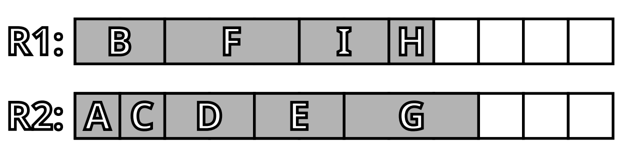
3) What would be the optimal execution time given 1 reusable resource? [10pts.]

4) What would be the optimal execution time given infinite resources? [10pts.]



[Solution:

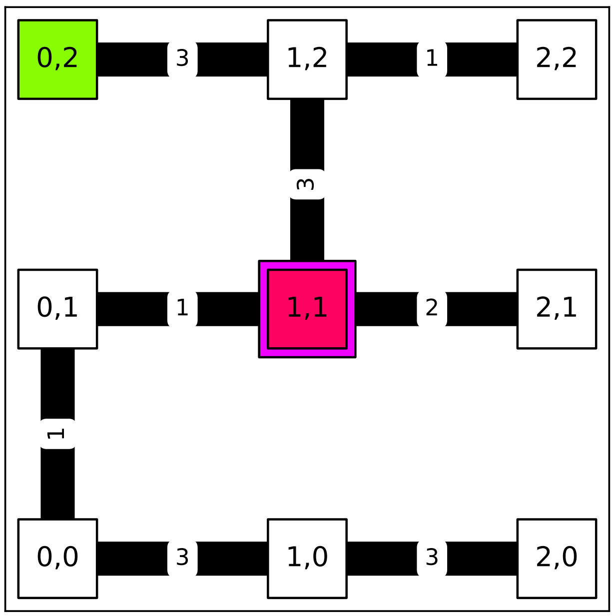
a	ES	LS
A	0	1
B	0	0
C	0	3
D	1	2
E	2	4
F	2	2
G	3	4
H	4	6
I	5	5



3) 17.
4) 7.]

13

In the following state graph, apply *LRTA** and report the list of visited states, including repetitions (e.g. $(0,1) \rightarrow (0,0) \rightarrow (0,1) \rightarrow ...$). The order of (untried) actions is $[up, right, down, left]$.



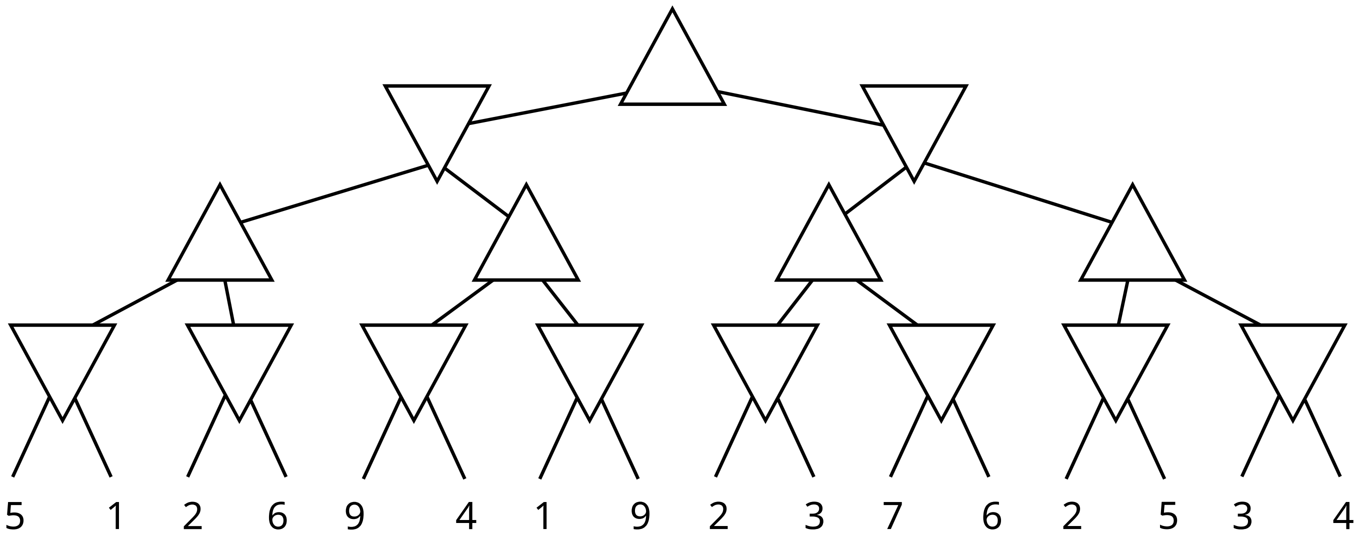
Start: $(1,1)$ - Goal: $(0,2)$

[Solution:

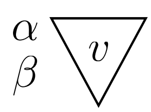
$(1,1)$, $(1,2)$, $(2,2)$, $(1,2)$, $(1,1)$, $(2,1)$, $(1,1)$, $(0,1)$, $(1,1)$, $(0,1)$, $(0,0)$, $(0,1)$, $(1,1)$, $(1,2)$, $(0,2)$
]

14

Use *Minimax with α, β-pruning* to propagate the utilities from the leaves to the root of the tree.

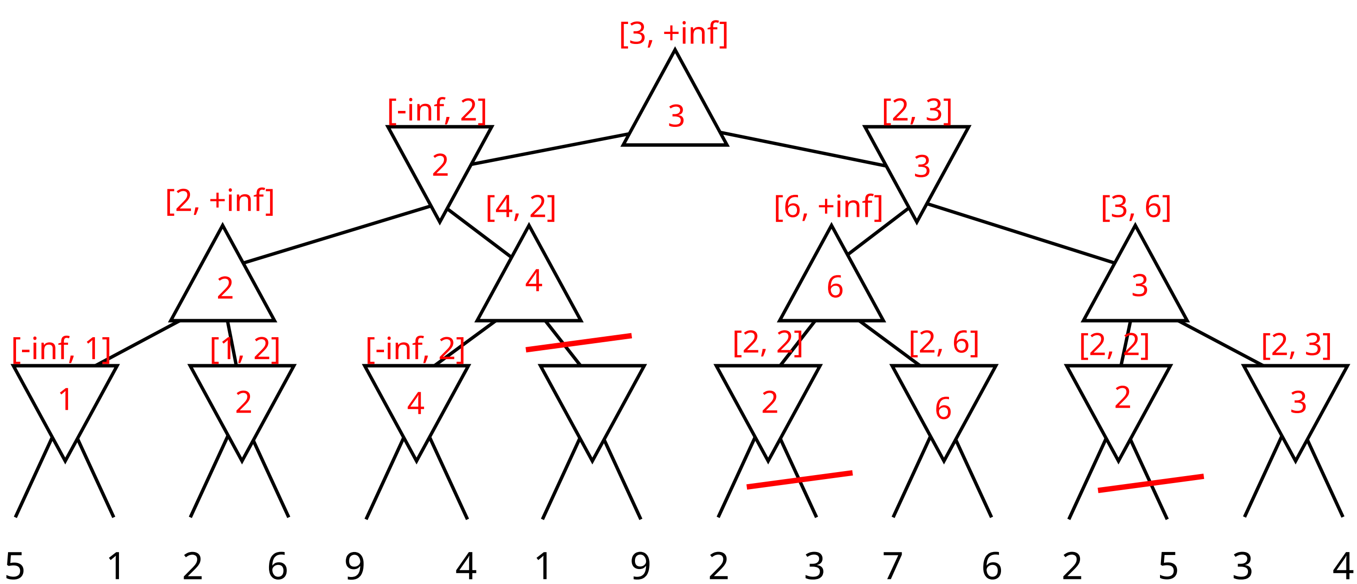


For each node report the values of α, β, as well as its returned value v **when the recursive calls ends**. Use the following format:



Additionally, clearly mark in the tree the pruned branches.

[Solution:



]

15

Use *hill climbing* for solving the maximization problem over (x, y) with the following objective function:

y								
7	0	2	3	4	4	2	1	0
6	1	1	4	5	5	3	2	1
5	2	3	5	8	6	4	2	3
4	3	5	6	9	10	6	5	1
3	3	4	7	10	9	6	4	1
2	2	2	4	6	7	5	4	3
1	1	1	3	5	4	3	1	1
0	0	1	2	3	4	2	2	0
	0	1	2	3	4	5	6	7	x

The next state is selected among the neighbors with strictly higher objective function value with *probability proportional to their objective function value*. Neighbors of (x, y) are sorted as follows:

$$(x-1, y-1), (x, y-1), (x+1, y-1), (x-1, y), (x+1, y), (x-1, y+1), (x, y+1), (x+1, y+1)$$

The choice vector is: $[1/4, 3/4]$.

The *initial state* is: $(0, 7)$.
For each step in the resolution process report (1) the current state; (2) the list of candidate next states.

[Solution:

it: 1, curr: (0, 7), candidates = [(0, 6) : 1/4, (1, 6) : 1/4, (1, 7) : 1/2], choice: 1/4 → (0, 6)
it: 2, curr: (0, 6), candidates = [(0, 5) : 2/7, (1, 5) : 3/7, (1, 7) : 2/7], choice: 3/4 → (1, 7)
it: 3, curr: (1, 7), candidates = [(2, 6) : 4/7, (2, 7) : 3/7], choice: 1/4 → (2, 6)
it: 4, curr: (2, 6), candidates = [(2, 5) : 5/18, (3, 5) : 8/18, (3, 6) : 5/18], choice: 3/4 → (3, 6)
it: 5, curr: (3, 6), candidates = [(3, 5) : 8/14, (4, 5) : 6/14], choice: 1/4 → (3, 5)
it: 6, curr: (3, 5), candidates = [(3, 4) : 9/19, (4, 4) : 10/19], choice: 3/4 → (4, 4)
it: 7, curr: (4, 4), candidates = [], STOP

]