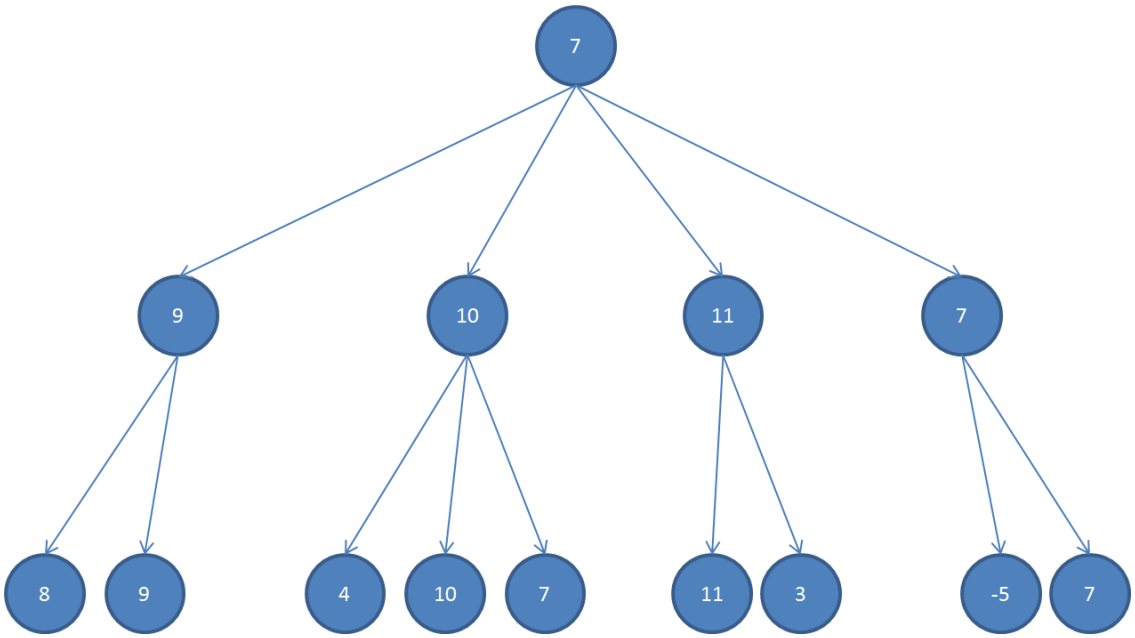
Timothee Guerin
260447866

Question 1

Samuel introduces with heuristics and the idea of a look-ahead tree to model board positions then explain techniques for reducing the amount of computations. (alpha-beta pruning -> plausibility analysis, forward pruning).After he discuss on how to handle "pitch moves" or sacrificial moves, which in the short term are negative, but perhaps in the long term, are positive. In the evaluation sections he looks at various ways to interpret results: Linear polynomial eval, the signature table method. He also compares machines to humans in terms of pre-programming; using a pre-built database or book to increase the power/speed up the learning of the machine/human (and how this can be done using sig-tables). Samuel describes various techniques for strategizing and discusses their strengths and weaknesses.

Question 2

Question 2.a



Question 2.b

Send $[-\infty, +\infty]$ to the first node

- $[8, +\infty]$ on first node
- $[9, 9]$ on second node

Root node now has $[-\infty, 9]$

- $[4, 9]$ on the first node
- $[10, 9]$ on second node
- Then we skip the third one

Root node still has $[-\infty, 9]$

- $[11, 9]$ on first node
- Then we skip the second one

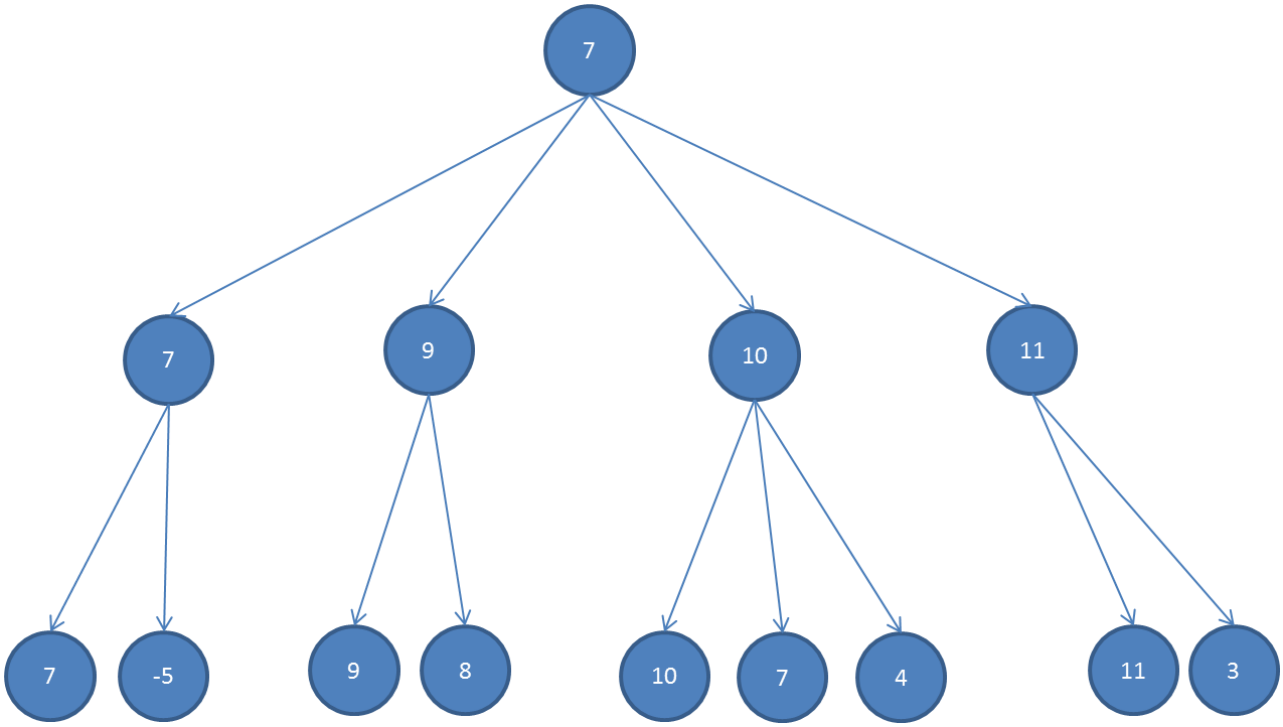
Root node still has $[-\infty, 9]$

- $[-5, 9]$ on first node
- $[7, 7]$ on second node

Then the root has the final value of [7,7]

Question 2.c

To skip the largest number of node we need to place the level 1 in ascending order and the level 2 in descending order so it will find the solution on try.



This is a possibility for eliminating the largest number of node with the $\alpha - \beta$ pruning.

Question 3

Question 3.a

We have two conditions:

- The number of vertices in V1 and V2 is as close as possible
- The number of edges where one end is a node in V1 and the other end is a node in V2 is as small as possible.

Then the conditions can be express as:

$$f = \begin{cases} \min(abs((vertices\ in\ V_1) - (vertices\ in\ V_2))) \\ \min((edge\ from\ V_1 to\ V_2) + (edges\ from\ V_2\ to\ V_1)) \end{cases}$$

Question 3.b

A state is a combination of two set V_1 and V_2 congaing all the node in the graph.

A neighbourhood state of a state can be a change of n node from one set to another. So the set of neighbourhoods is all the state such that n node changed set.

The function can pick a state if both values are smaller.

Question 3.c

- Individuals are the two set V_1 and V_2
- The fitness function is a combination of the two $(a * f_1 + b * f_2)$
- The mutation is inverting value in a state from state V_1 to V_2
- The crossover is to take a set of vertices from a state and put them in their respective set in another set.

Question 3.d

As the gradient ascent only going for better result it's probably going to get stuck without getting the best result. The simulated annealing however will be able to leave those gaps

Question 4

Question 4.a

Set of or constrains

Question 4.b

{(X, Y → green), or (X, Z -> green),or (Z, Y) ->green)

Question 4.c

For w_i^j with $i\ from\ 1\ to\ n^j$ we must have $(x + i, y) = w_i^j$ or $(x, y + i) = w_i^j$

Question 5

Question 5.a

i) unsatisfiable
ii) satisfiable all cases except when both P and Q are false
iii) satisfiable in cases where p = q

Question 5.b

i) The sentence is the conjunction of all $X_i\ from\ 1\ to\ n$ then for the sentence to be true all X_i must be true. Then we have only one solution

ii) We notice that for this sentence to be true we need a $i\ from\ 1\ to\ n-1$ such that $X_i and\ X_{i+1}$ are both true. The first term of the series are $0, 0, 1, 3, 8, 19, 43, 94, 201,...$ Then we get

$$2^n - Fibonacci(n+2)$$

Question 6

Question 6.a

Mary is in class but she is not bored:	$A \wedge \neg C$
Mary has to see a doctor if she has the flu.	$B \rightarrow D$
Mary does not go to class if she has the flu	$B \rightarrow \neg A$

Question 6.b

If the above is the only fact database then we need to have B in order to get to D. Then we need to have B to the knowledge database.

Question 7

Question 7.a

$$\forall x \; and \; \forall \, t : \; (cadet(x) \rightarrow obey(x, t))$$

$$\forall x \; and \; \exists \, t : \; (officer(x) \rightarrow \; obey(x, t))$$

$$\forall \, x : (member(x) -> \Big(\big(cadet(x) \vee \; officer(x)\big) \wedge \; \neg\big(cadet(x) \wedge \; officer(x)\big)\Big))$$

$$\forall \, x : \; cadet(x) \rightarrow \; blue(x)$$

$$\forall \, x : \; officer(x) \rightarrow \; red(x)$$

$$\neg blue(Data)$$

Question 7.b

It's not sufficient as there can be another uniform that blue or red then not having a blue doesn't specify we should have a red uniform. We could add an additional statement:

$$\neg blue(x) \rightarrow \; red(x)$$

Then we can deduce that

$$\neg blue(Data)$$

$$\neg blue(Data) \rightarrow \; red(Data)$$

However this is not yet sufficient, because all officers wear red uniforms but others could also wear red uniforms

So we need to add a new statement

$$\forall \, x : \; red(x) \rightarrow \; officer(x)$$

Then we have

$$red(Data) \rightarrow officer(Data)$$

$$officer(Data)$$

Question 7.c

You can't. Even assuming Data is an officer, we still have the sentence:

$$\forall x \; and \; \exists \; t : (officer(x) \rightarrow obey(x, t))$$

$$\exists \; t : (officer(Data) -> obey(Data, t))$$

Data will obey orders some time.

You can infer an existence E from a universal A but not the reverse.

Thus we cannot prove that data will obey orders all the time.