2018 cmput396 asn4 submit to eclass or in class by Nov 1 0800 15 marks

1. [1 marks] **Complete this page properly.**

   Each group submits only one assignment. Print the name and ID number of each group member (at most 4) for this assignment:

   _____

   _____

   _____

   _____

   Acknowledge **all** sources, including all references and all people not in your group with whom you discussed any part of any question (for each discussion, list the relevant questions) (continue on the back of this page if there is insufficient space). If none, write *webnotes and lectures only.*

   _____

   _____

   _____

   _____

   Each group member must read, agree to, and sign this statement:

   I am familiar with the Code of Student Behaviour. I understand that there are significant penalties for any infraction of this Code, including failure to acknowledge sources. I have not shared any written or printed version of any of my answers with any other student.

   _____

   _____

   _____

   _____

2. [7 marks] The function `foo()` from the quiz uses negamax to solve 3-pile nim.

(a) Starting with `foo()`, and making as few changes as necessary, write a python function `calls(a,b,c)` that returns the number of recursive calls made when computing `foo(a,b,c)`. (This is the same as the number of nodes in the search space.) E.g. calls(0,0,0) returns 1, calls(1,0,0) returns 2, calls (1,1,1) returns 16. Using your function, complete the table below.

j	0	1	2	3	4
calls(j,j,j)	1	16			

(b) Modify `foo()` by uncommenting the 3 comments. Also, insert `print(a,b,c)` as the first line of the function body. Now `(a,b,c)` is printed once every time a call is made. (If you prefer, you can modify `foo()` so that it returns both the value and the number of calls made, as in function `ab_neg()` in program `ttt_classic.py`.) Complete the following table.

j	0	1	2	3	4	5	6	7	8	9
calls made by modified foo(j,j,j)	1	6								

(c) Assume that you modified the negamax version of foo from (a) to use alpha-beta (this would give you the alphabeta-negamax version of foo). Would the number of calls made be **(choose one)** (always fewer, usually fewer, always equal, usually more, always more) than the number of calls made by the modified foo from (b)? Explain briefly.

(d) Modify function `solveall()` in program `nim.py` so that it prints the number of loss updates (each time execution reaches the comment "...loses, find all wins ...") and win updates (each time execution reaches the comment "print(self.crd(pjc),'wins')"), and complete the following table.

j	0	1	2	3	4	5	6	7	8	9
total loss/win updates made by solveall nim(j,j,j)	0	10								

Here is a simplified description of $O(n)$, $\Omega(n)$ and $\Theta(n)$. For a precise description, see wikipedia or any introduction to algorithms.

"The runtime is in O(f(n))" means that as a function of input size n, the runtime, say $r(n)$, is in a set of functions of n that grow no faster than some constant times $f(n)$. In particular, there is a starting integer k and a constant c, such that for all $n \geq k$, $r(n) \leq cf(n)$.

Similarly, "the runtime is in $\Omega(n)$" means that $r(n)$ is in a set of functions of n that grow at least as fast as a constant times $f(n) = n$. And "the runtime is in $\Theta(n)$" means that $r(n)$ is $O(f(n))$ and $r(n)$ is in $\Omega(f(n))$. Finally, "the runtime is in $O(n^2)$" is a short way of saying "the runtime is in $O(f(n))$, where $f(n) = n^2$".

Example. Let $r(n) = 3n + 1/n$ and let $f(n) = n$. Then $r(n)$ is in $\Omega(f(n))$.

Proof: for all integers $n \geq 1$, $1/n > 0$, so $r(n) = 3n + 1/n > 3n$. So, for all integers $n \geq 1$, $r(n) > 3 * n$. So we have found an integer k (1) and a constant c (3) so that $r(n) \geq cf(n)$, so $r(n)$ is in $\Omega(f(n))$.

3. [3 marks] Prove that $r(n) = 3n + 1/n$ is in $O(n)$. Follow the form of the previous example.

Plotting runtime for different values of n can sometimes indicate how the runtime grows. E.g. assume that $r(10) = 10$ seconds and $r(20) = 40$ seconds. Is it possible that $r(n) = cn$ for some constant c? Is is possible that $r(n) = cn^2$ for some constant c?

Answer. Assume $r(n) = cn$. Since $r(10) = 10$, then we must have $c = 1$, so $r(20)$ should be $c * 20 = 1 * 20 = 20$. But it is not. So there is no constant c such that $r(n) = cn$.

Now assume $r(n) = cn^2$. Since $r(10) = 10$, then we have $10 = c * 10 * 10$, so $c = 1/10$, and we expect $r(20) = c * 20 * 20 = 1/10 * 20 * 20 = 40$. This is correct. So there is a constant c such that $r(n) = cn^2$.

4. [2 marks] Assume $r(10) = 10$ and that $r(n) = cn^3$ for some constant c. Find c, and find $r(20)$.

5. [2 marks] Kim claims that the runtime of class program `nim.py` is in $O(s)$, where s is the number of different nim positions in the state space. For example, for nim(3, 3, 3), there are $4 * 4 * 4$ positions in the state space, since each pile can have 0,1,2 or 3 stones.

   Prove or disprove this claim. Hint: consider nim(2), nim(2,2), nim(2,2,2), and so on.