Discrete Structures

Worksheet 5

Spring 2021
February 3/5

Recall the definition of **Big-O** notation: Let $f, g\colon X \to \mathbf{R}$ be functions, for $X \subseteq \mathbf{R}$ and $a \in \mathbf{R}$. Then we say "$f(x)$ *is Big-O of* $g(x)$ *as* x *goes to* a", and write:

$$\text{“}f(x) = O(g(x)) \text{ as } x \to a\text{”}, \text{ or } \text{“}f(x) \text{ is } O(g(x)) \text{ as } x \to a\text{”}$$

if there exists $\epsilon > 0$ and $M > 0$ such that $|f(x)| \leqslant M|g(x)|$ for all $x \in (a - \epsilon, a + \epsilon)$. If a is clear from context, and most often $a = \infty$, we write

$$\text{“}f(x) = O(g(x)\text{”}, \text{ or } \text{“}f(x) \text{ is } O(g(x))\text{”},$$

and in the $a = \infty$ case, the condition on x is changed to "for all $x > \epsilon$". This condition is specialized to functions whose domain is $X = \mathbf{Z} \subseteq \mathbf{R}$ in Question 5.

---

1. **Warm up:** Write the following English sentences using logical symbols. Avoid using the negation symbol $\neg$ by changing the quantifiers and the inequality signs.

   (a) For functions $f : \mathbf{R} \to \mathbf{R}$ and $g : \mathbf{R} \to \mathbf{R}$ it is not the case that f is $O(g(n))$.

   (b) The inequality $f(n) > g(n)$ holds for any real argument n, but f is not $\Theta(g(n))$.

   (c) If f is not $O(g(n))$, then g is not $\Omega(f(n))$.

2. Recall the remainder function $\text{rem}\colon \mathbf{Z} \times \mathbf{Z} \to \mathbf{Z}$ that gives the remainder when the first input is divided by the second input. Consider the following algorithm, which takes as input an integer n.

```
1  procedure func(n : integer)
2  m := ⌊log_2(|n|)⌋
3  if rem(m, 2) = 0:
4     k := ⌊current temperature⌋
5  if rem(m, 2) = 1:
6     k := func(3n + 2)
7  return k
```

   (a) What is the output type of this algorithm?

   (b) Is this algorithm finite?

   (c) Is this algorithm effective?

   (d) What happens when a real number is input instead of an integer?

3. This question compares the linear search and the binary search algorithms.

   (a) What is a difference between the inputs of these two algorithms?

   (b) What is a difference between the procedures of these two algorithms?

   (c) What is a difference between the worst cases of these two algorithms?

   (d) Given an example of an input on which the binary search algorithm takes less steps than the linear seacrh algorithm. A "step" is considered to be every event in which the element to find x is compared with a list element.

   (e) Suppose that the input list is infinitely long.

      i. What problems are encoutered by each of these two algorithms?

      ii. Suggest a solution to fix these problems.

4. (a) In your own words, what does it mean for a problem to be an *optimization* problem?

   (b) In your own words, what does it mean for an algorithm to be a *greedy* algorithm?

   (c) Trace out S, j in the greedy algorithm for scheduling talks (Algorithm 8 on page 212) on the start times for the talks as below.

	$i=1$	$i=2$	$i=3$	$i=4$	$i=5$	$i=6$	$i=7$	$i=8$
s_i	9 : 00	13 : 30	17 : 00	13 : 00	11 : 30	15 : 00	11 : 00	9 : 30
e_i	10 : 00	14 : 45	17 : 30	14 : 00	12 : 00	15 : 50	12 : 30	10 : 30

5. For each function below and its given growth rate, find $C \in \mathbf{N}$ and the smallest possible $n_0 \in \mathbf{N}$ such that $\exists\, C \in \mathbb{Z}^+ \;\exists\, n_0 \in \mathbb{Z}^+ \;\forall\, n \in \mathbb{Z}^+ \; (n > n_0 \to |f(n)| \leq C \cdot |g(n)|)$.

   (a) Let $f(n) = n^3 + 88n^2 + 3$, and you may assume that $f(n)$ is $O(n^3)$.

   (b) Let $g(n) = \ln(n^4) + n \cdot \arctan(n)$, and you may assume that $g(n)$ is $O(n)$.

6. Let $n, m \in \mathbf{Z}_{>0}$.

   (a) Suppose that $f(x) = O(x^n)$ as $x \to 0$, and $g(x) = O(x^m)$ as $x \to 0$. Show that $f(x) + g(x) = O(x^k)$ as $x \to 0$, where $k = \min\{m, n\}$.

   (b) Suppose that $f(x) = O(x^n)$ as $x \to \infty$, and $g(x) = O(x^m)$ as $x \to \infty$. Show that $f(x) + g(x) = O(x^\ell)$ as $x \to \infty$, where $\ell = \max\{m, n\}$.

   (c) Let $f\colon \mathbf{Z}_{>0} \to \mathbf{Z}_{\geqslant 0}$ be the function that, for an input m, returns the number of times line 6 from Question 2 is called when the procedure is run on every integer $2m, 2m+1, \ldots, 3m$. Find the smallest function $g(m)$, such that $f(m) = O(g(m))$.

   (d) Do you think there exists an alogrithm that sorts a list of length n, that has running time $O(1)$? Why or why not?

7. Arrange the following functions in order of increasing $O(-)$.

$$\log(n^{10}) \qquad (\log n)^2 \qquad \log(\log(n))$$

$$n\log(n) \qquad \log(n!) \qquad \log(2^n)$$

   That is, if $f(n)$ comes before $g(n)$ in your arrangement, then $f(n)$ is $O(g(n)$.

8. For each function $f(n)$ defined below, find the optimal $g(n)$ such that $f(n)$ is $O(g(n))$. That is, make sure that if $f(n)$ is also $O(h(n))$, then $g(n)$ is $O(h(n))$.

   (a) $f(n) = 1^2 + 2^2 + \ldots + n^2$

   (b) $f(n) = \dfrac{3n - 8 - 4n^3}{2n - 1}$

   (c) $f(n) = \displaystyle\sum_{k=1}^{n} k^3$

   (d) $f(n) = \dfrac{6n + 4n^5 - 4}{7n^2 - 3}$

   (e) $f(n) = \displaystyle\sum_{k=2}^{n} k \cdot (k-1)$

   (f) $f(n) = 3n^2 + 8n + 7$

9. Match the statements in English on the left to the predicate expressions on the right.

[i] Some programs return the correct result for all possible inputs and they never loop indefinitely.

[ii] For any program one can find another program such that it returns the same result for the same inputs as the first one (or loops indefinitely, **iff** the first program does the same).

[iii] There is a program that only loops indefinitely for at most finitely many inputs (or maybe none at all), but for all other inputs it produces the correct result.

[iv] There is at least one Python program that always halts, and for sufficiently large inputs it produces the correct result, but it may err for some small-size inputs.

[v] For a program to produce a correct result for some input i it is strictly necessary to halt.

[vi] A Python program always produces exactly one result for the given input provided that it halts.

[I] $\forall p \in \mathcal{P}\ \forall i \in \mathbb{Z}^+\ \forall r \in \mathbb{Z}^+$
$(A(p,i,r) \wedge C(i,r) \ \rightarrow\ H(p,i))$

[II] $\forall p_1 \in \mathcal{P}\ \exists p_2 \in \mathcal{P}\ \forall i \in \mathbb{Z}^+\ \forall r \in \mathbb{Z}^+$
$\Big((\neg H(p_1,i) \wedge \neg H(p_2,i))$
$\vee (A(p_1,i,r) \leftrightarrow A(p_2,i,r))\Big)$

[III] $\exists p \in \mathcal{P}\ \exists N \in \mathbb{Z}^+\ \forall i \in \mathbb{Z}^+\ \exists r \in \mathbb{Z}^+$
$((i \le N \wedge \neg H(p,i)) \vee (A(p,i,r) \wedge C(i,r)))$

[IV] $\forall p \in \mathcal{P}\ \forall i \in \mathbb{Z}^+\ \forall r_1 \in \mathbb{Z}^+\ \forall r_2 \in \mathbb{Z}^+$
$(H(p,i) \wedge A(p,i,r_1) \wedge A(p,i,r_2) \ \rightarrow\ r_1 = r_2)$

[V] $\exists p \in \mathcal{P}\ \exists N \in \mathbb{Z}^+\ \forall i \in \mathbb{Z}^+\ \forall r \in \mathbb{Z}^+$
$(H(p,i)\ \wedge\ (A(p,i,r) \wedge (i > N)\ \rightarrow\ C(i,r)))$

[VI] $\exists p \in \mathcal{P}\ \forall i \in \mathbb{Z}^+\ \forall r \in \mathbb{Z}^+$
$(H(p,i)\ \wedge\ (A(p,i,r) \rightarrow C(i,r)).$

The functions A, H, C are defined as below.

- $A(p_1, i_2, r_3)$ is true iff Python program $p_1 \in \mathcal{P}$ receives input $i_2 \in \mathbb{Z}^+$ and outputs result $r_3 \in \mathbb{Z}^+$
- $H(p_1, i_2)$ is true iff program $p_1 \in \mathcal{P}$ receives input i_2 and halts (that is, does not loop indefinitely)
- $C(i_1, r_2)$ is true iff for input i_1 the correct result is r_2

10. (a) You have 4 coins with different weights. You can compare any two of them on two-sided balance scales (you can determine, which coin is heavier). The task is to arrange them in increasing order by their weight. Assume that somebody claims that using these scales 4 times is always sufficient. How many outcomes can you have, if you use scales 4 times (and every time there are two possibilities – either one side is heavier or the other).
Find the number of ways 4 coins can be arranged in some order (these ways are called *permutations*).

(b) You have 16 stones; you know that one of them is radioactive. You can put any number of stones in an analyzer and it tells, if any of the stones among those tested were radioactive (but the device does not point to the radioactive stone). Can you find the radioactive stone using the analyzer just 4 times?

11. Consider the set $X = \{1, 2, \ldots, n\}$ and subsets $S_1, \ldots, S_n \subseteq X$. Consider an algorithm A that determines whether or not there is a disjoint pair of subsets $S_i \cap S_j = \emptyset$. The algorithm works in the following way:

    - A loops through the subsets, and for each subset S_i, it loops through all other subsets S_j, and for each of these other subsets S_j, it loops through all elements k in S_i to determine whether k also belongs to S_j.
    - As soon as A finds any two disjoint subsets, it outputs their numbers i and j, and immediately stops.

    Answer the following questions about the algorithm A.

    (a) Write A in pseudocode, using the line "**if** $i \in S_j$: ...".

    (b) Write A in pseudocode, using **for** loops and iterator loops "**foreach** $k \in S_j$" and test elements for equality, instead of using the line from part (a).

    (c) Give a big-O estimate for the number of times the algorithm, as written in part (b), tests element equality.

12. Consider the code in Python below.

```
sum = 0
for i in range(1,n+1):
    for j in range(1,n+1):
        sum += (i*t + j*t + 1)**2
```

    The parameter `n` is a natural number, and `t` is fixed. Let $f(\mathtt{n})$ be the number of operations executed when the above code is run. An "operation" is addition, multiplication, or raising to the power 2. Find the optimal $g(n)$ so that $f(n)$ is $O(g(n))$.

13. Let $f\colon \mathbf{R} \to \mathbf{R}$ and $g\colon \mathbf{R} \to \mathbf{R}$ be functions. Find all of the predicate logic expressions below that are logically equivalent to "The function $f(n)$ is $O(g(n))$".

    (a) $\forall n \in \mathbb{R}\ \exists n_0 \in \mathbb{R}\ \exists C \in \mathbb{R}$, $(n > n_0 \to |f(n)| \leq C \cdot |g(n)|)$

    (b) $\exists n_0 \in \mathbb{R}\ \forall n \in \mathbb{R}\ \exists C \in \mathbb{R}$, $(n > n_0 \to |f(n)| \leq C \cdot |g(n)|)$

    (c) $\exists n_0 \in \mathbb{R}\ \exists C \in \mathbb{R}\ \forall n \in \mathbb{R}$, $(n > n_0 \to |f(n)| \leq C \cdot |g(n)|)$

    (d) $\exists n_0 \in \mathbb{R}\ \exists C \in \mathbb{R}\ \forall n \in \mathbb{R}$, $(n > n_0 \to f(n) \leq C \cdot |g(n)|)$

    (e) $\exists n_0 \in \mathbb{R}\ \exists C \in \mathbb{R}\ \forall n \in \mathbb{R}$, $(n > n_0 \to |f(n)| \leq C \cdot g(n))$

    (f) $\exists n_0 \in \mathbb{R}\ \exists C \in \mathbb{R}\ \forall n \in \mathbb{R}$, $(n \geq n_0 \to |f(n)| < C \cdot |g(n)|)$

    (g) $\exists n_0 \in \mathbb{Z}^+\ \exists C \in \mathbb{Z}^+\ \forall n \in \mathbb{R}$, $(n > n_0 \to |f(n)| \leq C \cdot |g(n)|)$

14. There is a collection of n stone axes of different weights, and the heaviest and the lightest of them are used to commit a crime. You must figure out which axes were used, and you may only compare any two axes on scales to find which is heavier and which is lighter.

    (a) Describe an algorithm, in English or in pseudocode, to find both the heaviest and the lightest stone axe in this collection.

    (b) Is it possible to find the lightest and heaviest axes using less than $2n-2$ comparisons?

    (c) Is it possible to find the lightest and heaviest axes using less than $(3/2)n - 2$ comparisons?

(d) Suppose that you already know what are the lightest and heaviest axes in the collection of n axes. What is the minimum number of comparisons you need to convince someone that these are indeed the lightest and heaviest ones?

15. Download the file `worksheet05-predicates.v` from ORTUS (under Week5). It contains 4 statements with predicates and proofs. (Homework 5 contains more statements to prove.) Run the proofs contained there, make sure that you understand all the proof tactics used. Also consider reading `https://bit.ly/2O36EZ5`, pages 38–43.

    Hint. If Coq IDE crashes when you double-click the file, try opening Coq IDE application without any file and copy-paste the text from `worksheet05-predicates.v` into the editor.