Discrete Structures

Worksheet 21

Spring 2022
March 18

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 1.

---

1. 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 $O(n)$.

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

3. For $X = \{1, 2, \ldots, n\}$ and a a set of subsets $S = \{S_1, \ldots, S_k\}$, $S_i \subseteq X$, there is an algorithm that determines whether or not there are two subsets $S_i, S_j \in S$ with $S_i \cap S_j = \emptyset$. The algorithm works in the following way:
   - For each subset S_i, the algorithm looks at all other subsets S_j, and for each of these other subsets S_j, it looks at every element k in S_i to determine whether k also belongs to S_j.
   - As soon as the algorithm finds any two disjoint subsets, it outputs their numbers i and j, and stops.

   Answer the following questions about the algorithm A.
   (a) Write the algorithm in pseudocode, using the line "**if** $i \in S_j$: …" somewhere.
   (b) Write the algorithm in pseudocode, using **for** loops and "**foreach** $k \in S_j$" loops. 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.

4. Consider the code in Python below.

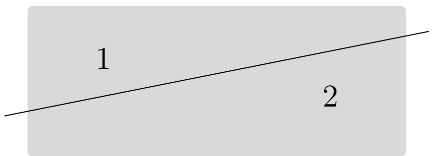
```
sum = 0
for i in range(1,n+1):
    for j in range(1,n+1):
        sum += (i*t + j*t + 1)**2
```

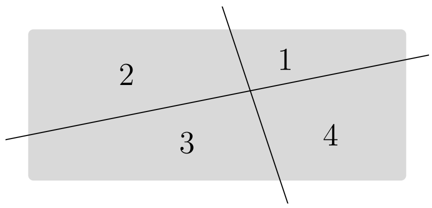
The number `n` is a natural number and `t` is a real number. Let $f(\mathtt{n})$ be the number of operations executed when the above code is run. An "operation" is addition, multiplication, or raising to the power 2. Find the optimal $g(n)$ so that $f(n)$ is $O(g(n))$.

---

5. Complete the following tasks for next lab (Tuesday). They will be presented at the beginning of the lab.

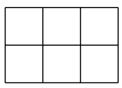
(a) A line in the plane separates the plane into two regions (above and below the line). For two lines, it is four:



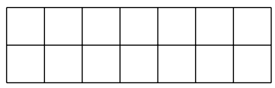


Using a recurrence relation, find the number of regions into which n (non-parallel) lines separate the plane.

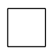
(b) For each $n \in \mathbf{N}$, a $2 \times n$ checker board may be tiled using 1×1 and L-tiles (3 tiles arranged in an L-shape).



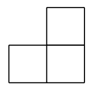
2×3 checker board



2×7 checker board



1×1 tile



L-tile

Using a recurrence relation, find the number of ways a $2 \times n$ checker board may be tiled using these two tiles.

(c) Write the following English sentences using logical symbols. Avoid using the negation symbol $\neg$ by changing the quantifiers and the inequality signs.

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

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

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

(d) 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 $O(g(n)$. Justify your work!