§2.3

WrittenHW#4

TCSS 321

1. [2] Determine whether f is a function from $\mathbb{Z}$ to $\mathbb{R}$ and justify your answer.
   (a) $f(n) = \sqrt{n^2 + 1}$
   (b) $f(n) = \dfrac{1}{n^2 - 9}$
2. [2] Create a function that is not the identity function, from $\mathbb{Z}$ to $\mathbb{Z}$ that has an inverse.
3. Let $f(x) = \lfloor x \rfloor + \lceil x/3 \rceil$ be a map from $\mathbb{R}$ to $\mathbb{R}$.
   (a) [1] Evaluate $f(5)$.
   (b) [1] Evaluate $f(-13)$.
   (c) [2] Identify the image/range of f.
   (d) [2] Graph f. Make sure to clearly indicate any endpoints.

§3.2

WrittenHW #4

TCSS 321

1. [2] Show that x^3 is $O(x^5)$ but x^5 is not $O(x^3)$.
2. [3] Arrange the functions , $\sqrt{n}$, 2^n, $n \log n$, $1000 \log n$, and $\frac{n^2}{10000}$ in a list so that each function is bit-O of the next function. Justify the ordering.
3. [3] Let n be a positive integer, is $1^3 + 2^3 + 3^3 + 4^3 + \ldots + n^3 \in O(n^3)$? $O(n^4)$?
4. [2] Suppose that you have two different algorithms for solving a problem. To solve problem of size n, the first algorithms uses exactly $n^2 2^n$ operations and the second algorithm uses exactly $n!$ operations. As n grows, which algorithm uses fewer operations? Justify yourself.

WrittenHW #4

1. [3] Give a big-θ for the number of additions used in the below segment of an algorithm. note: in industry this is normally called big-O....

    $t := 0$
    for $i = 1$ *to* n **do**
        for $j = 1$ *to* n **do**
            $t := t + i + j$
        end
    end
    return C $\#C = [c_{ij}]$

    Algorithm 1: Matrix Algorithm

2. [3] Consider the following algorithm for evaluating polynomials at the value c. Work through each step of the algorithm with the polynomial $3x^2 + x + 1$ at $x = 2$ showing the values assigned at each assignment step.

    Data: c, a_0, a_1, $....a_n$: real numbers
    $y := a_n$
    for $i = 1$ *to* n **do**
        $y := y * c + a_{n-i}$
    end
    return y $\{y = a_n c^n + ... + a_1 c + a_0\}$

    Algorithm 2: Matrix Algorithm

3. [2] Consider the algorithm described in the above question. Exactly how many multiplications and additions are used by this algorithm to evaluate a polynomial of degree n at $x = c$? (Do not count additions used to increment the loop variable.)

4. [2] How does the number of comparisons change from n to $2n$ with bubble sort?