**CS251 Week 04, Lab Exercises.**

**More Runtime Practice!**

**PART 1:** Spend about 20 minutes working through these exam-style questions to the best of your ability. Debate your answers with your neighbors.

|  |
| --- |
| **QUESTION 1.1:**  Fill in the blanks. Complete each of the statements below with either or whichever is correct and gives the most information about the relationship.  is \_\_\_\_\_\_\_()  is \_\_\_\_\_\_\_\_\_\_\_( )  is \_\_\_\_\_\_\_\_\_()  is \_\_\_\_\_\_\_\_\_\_(  is \_\_\_\_\_\_\_\_\_\_\_\_  is \_\_\_\_\_\_\_\_\_\_\_\_\_  is \_\_\_\_\_\_\_\_\_\_\_\_\_\_( |

|  |
| --- |
| **QUESTION 1.2:**  We are given the following facts:  “Algorithm A’s worst case runtime is ”  “Algorithm B’s worst-case runtime is ”  As far as worst-case asymptotic runtime, which algorithm would we prefer?   1. Algorithm A 2. Algorithm B 3. not enough information 4. depends on the value of |

|  |
| --- |
| **QUESTION 1.3:**  Suppose I have two algorithms X and Y and I know that the **best, worst and average case runtime** properties below hold:  “Algorithm X’s runtime is ”  “Algorithm Y’s runtime is ”  TRUE or FALSE: “Algorithm Y performs faster than Algorithm X **for all input sizes”**   1. TRUE 2. FALSE |

|  |
| --- |
| **INTERMISSION: TA Will Now GIve answers to Part-I** |

**PART 2: now work through as many of the following as you can.**

|  |  |
| --- | --- |
| **QUESTION 2.1:**  Which statement best describes the **worst-case** runtime of the C function below as a function of n?   |  | | --- | | **int foo(int a[], int n) {**  **int i, j, x;**  **x=0;**  **for(i=0; i<n; i++) {**  **if((a[i] % 2) == 0){**  **for(j=1; j<n; j \*= 2){**  **x += a[j];**  **}**  **}**  **else {**  **for(j=0; j<i; j++) {**  **x -= a[j];**  **}**  **}**  **}**  **return x;**  **}** | |

|  |  |
| --- | --- |
| **QUESTION 2.2:** Same function; which statement best describes the **best-case** runtime of foo?   |  | | --- | | **int foo(int a[], int n) {**  **int i, j, x;**  **x=0;**  **for(i=0; i<n; i++) {**  **if(a[i] % 2) == 0){**  **for(j=1; j<n; j \*= 2){**  **x += a[j];**  **}**  **}**  **else {**  **for(j=0; j<i; j++) {**  **x -= a[j];**  **}**  **}**  **}**  **return x;**  **}** | |

|  |  |
| --- | --- |
| **QUESTION 2.3:**  Consider the C function tick below. If we call **tick(4)**, how many times will **TICK** be printed to the screen?   |  | | --- | | **void tick(int n) {**  **int i;**  **if(n <= 0) return;**  **for(i=0; i<n; i++){**  **printf(“TICK\n”);**  **}**  **tick(n/2);**  **}** |  1. 4 2. 16 3. 8 4. 7 |
| **QUESTION 2.4:** Same function. How many TICKs are printed when tick(8) is called?   1. 8 2. 24 3. 15 4. 16 |
| **QUESTION 2.5:** Same function. Let be the number of TICKs printed when tick called with a parameter of . Which of the following best describes   1. is 2. is 3. is 4. is |

**Appendix.**

**Big-Oh Definition**

**Given two integer functions T(n) and f(n), we say that:**

**T(n) is O(f(n)) if and only if there exist constants c>0 (real) and n0>0 (int) such that**

**T(n)≤ cf(n) for *all* n ≥ n0**

**f(n) is an “asymptotic *upper-bound”* on T(n)**

**Big-Omega Definition**

**Given two integer functions T(n) and f(n), we say that:**

**T(n) = Ω(f(n)) if and only if there exist constants c>0 (real) and n0>0 (int) such that**

**T(n) ≥ cf(n) for all n ≥ n0**

**f(n) is an “asymptotic *lower-bound”* on T(n)**

**Big-Theta Definition**

**Given two integer functions T(n) and f(n), we say that:**

**T(n) is Θ(f(n)) if and only if**

**T(n) = O(f(n)) *and***

**T(n) = Ω(f(n))**

**equivalently:**

**T(n) = O(f(n)) and**

**f(n) = O(T(n))**

**T(N) and f(n) are “asymptotically equivalent” -- they grow at the same rate.**