Programming and Data Structures

Week 5 Assignment

**Reminder: All work must be your own!**

**Question 1)** Use the following selection sort algorithm to answer the questions below:

void swap(int \*xp, int \*yp) {

int temp = \*xp;

\*xp = \*yp;

\*yp = temp;

}

// A function to implement selection sort

void selectionSort(int arr[], int n) {

// One by one move boundary of unsorted subarray

for (int i = 0; i < n-1; i++) {

// Find the minimum element in unsorted array

int min\_idx = i;

for (int j = i+1; j < n; j++) {

if (arr[j] < arr[min\_idx]) {

min\_idx = j;

swap(&arr[min\_idx], &arr[i]);

}

}

}

}

a) Identify the straight-line code in the above algorithm. You can describe or underline.

I have underlined the straight-line code in the above algorithm.

b) Fill in the following table that counts the number of times that the innermost piece of code

will be executed.

|  |  |  |
| --- | --- | --- |
| Iteration # | Value of i | # of executions |
| 1 | 0 | n-1 |
| 2 | 1 | n-2 |
| 3 | 2 | n-3 |
| … |  |  |
| n-1 | n-2 | n-(i+1) |
| n | n-1 | 0 |

c) Sum the last column of the table and simplify as much as you can.

The number of executions of the innermost piece of code is equal to . For example, an array of size 8 would execute the innermost piece of code 7+6+5+4+3+2+1 = 28 times.

d) Based on your answer to **c** what is the runtime of the algorithm?

We know that = . Substituting “n-1” for “n” yields . The expression simplifies to . Dropping the constant “” and the lower-ordered term “n”, the runtime of the selectionSort algorithm is .

**Question 2)** For each code snippet, state its runtime in terms of N, you can assume that the ‘...’

represents straight line code.

a) for (int i = N; i >= 0; i -= 4) { … }

b) for (int i = 1; i < N; i \*= 5) { … }

c) for (int i = 0; i < N; i++) {

for (int j = N; j > 0; j /= 2) { … }

}

d) for (int i = 0; i < N; i++) {

for (int j =N; j > i; j−−) { … }

}

|  |  |  |
| --- | --- | --- |
| Iteration | i | Inner Iterations |
| 0 | 0 | N |
| 1 | 1 | N-1 |
| 2 | 2 | N-2 |
| k | k | N-k |
| N | N | N-N = 0 |

e) for (int i = 1; i < N; i∗=2) {

for (int j = 0; j < i; j++) { … }

}

|  |  |  |
| --- | --- | --- |
| Iteration | i | Inner Iterations |
| 0 | 1 | 1 |
| 1 | 2 | 2 |
| 2 | 4 | 4 |
| k |  |  |
|  |  | N |

**Question 3)** For each of the following function pairs (f & g), give an M and x0 that holds that f(x)

∈ O(g(x)). You do not need to write a proof of such, just state an M and x0 that the formula holds

for. For some M and x0, f(x) <= M \* g(x), for all x > x0. For all questions, it is true that f(x) =

O(g(x)).

**Hint:** Consider setting the two formulas equal and solving for x.

a) f(x) = 100x + 10

g(x) = 5x

M = 21

X0 = 2

b) f(x) = 10x

g(x) = 1/2 x2

M = 1

X0 = 20

c) f(x) = 1000x2

g(x) = x3

M = 1

X0 = 1000

**Question 4)** For each of the following function pairs, use the limit rule to determine which of the following options best applies:

i) f(x) ∈ O(g(x))

ii) f(x) ∈ Ω(g(x))

iii) f(x) ∈ θ(g(x))

a) f(x) = 3x2

g(x) = 15x2

Therefore, iii) f(x) ∈ θ(g(x))

b) f(x) = x2

g(x) = 3x3

Therefore, i) f(x) ∈ O(g(x))

c) f(x) = log2(x)

g(x) = log3(x)

Apply L’Hopital’s Rule.

Say which is f(x) (the numerator). Then .

The same can be done for the denominator g(x). Then we have:

Therefore, iii) f(x) ∈ θ(g(x))

d) f(x) = x \* log(x)

g(x) = 5x

Therefore, ii) f(x) ∈ Ω(g(x))

e) f(x) = 2log2(x)

g(x) = 2x

Therefore, iii) f(x) ∈ θ(g(x))