# Individual Portion

## 1. Big-$O$reo [15 points]

Give the tightest big-O estimate for each of the following functions. Justify your answers.

(a) $f(n) = (2^n + n^n) \cdot (n^3 + n \log n^n)$

(b) $g(n) = (n^n + n!) \cdot (n + 1) \quad + \quad (n^3 + 3^n) \cdot (\sqrt{n} + \log n)$

(c) $h(n) = (n^n + n^2) \cdot (n^n + n) \quad + \quad (\log 3 + n^n) \cdot (n^2 + n^n)$

**Solution:**

(a) $n^n$ grows faster than $2^n$ and $n^3$ grows faster than $n \log n^n$. Thus, the tightest big-O estimate for $f(n)$ is $O(n^{n+3})$.

(b) $n^n$ grows faster than $n!$ and $3^n$ grows faster than $n^3$. Thus, the tightest big-O estimate for $g(n)$ is $O(n^{n+1})$.

(c) $n^n$ grows faster than $n^2$ and $n^n$ grows faster than $n$. Thus, the tightest big-O estimate for $h(n)$ is $O(n^{2n})$.

## 2. On the Run [20 points]

Give the tightest big-O estimate for the number of operations (where an operation is arithmetic, a comparison, or an assignment) used in each of the following algorithms. **Explain your reasoning.**

(a)
```
function doubleTrouble(a_1, ..., a_N ∈ ℝ, j ∈ ℝ)
    j ← 1
    for i := 1 to N do
        if i = j then
            j ← 2j
        end if
    end for
    return j
end function
```

(b)
```
function sumSquares(N ∈ ℤ⁺)
    if N = 1 then
        return 1
    end if
```

```
    value ← sumSquares(N − 1) + N^2
    return value
end function
```

(c)
```
function findLTMinProduct(a_1, ..., a_N ∈ ℝ)
    p ← 203
    for i := 1 to N do
        for j := 1 to N do
            if a_i a_j < p then
                p ← a_i a_j
            end if
        end for
    end for
    numLTMinProduct ← 0
    for k := 1 to N do
        if a_k < p then
            numLTMinProduct ← numLTMinProduct + 1
        end if
    end for
    return numLTMinProduct
end function
```

(d)
```
function subtractAndAdd(N ∈ ℤ)
    while N > 0 do
        if N is even then
            N ← N − 3
        end if
        if N is odd then
            N ← N + 1
        end if
    end while
    return N
end function
```

(e)
```
function search(a_1, ..., a_N ∈ ℝ, target ∈ ℝ)
    left ← 1
    right ← N
    while True do
        mid ← ⌊(left+right)/2⌋
        if a_mid = target then
            return mid
        end if
        if right ≤ left then
            return −1
        end if
```

**if** $a_{mid} < target$ **then**
  $left \leftarrow mid + 1$
**end if**
**if** $a_{mid} > target$ **then**
  $right \leftarrow mid - 1$
**end if**
**end while**
**end function**

**Solution:**

(a) O(DoubleTrouble) = O(N). This is because the loop is run N times, and the only operation inside the loop is an assignment.

(b) O(sumSquares) = O(N). This is because the function is called recursively N times, and the only operations inside the function are assignments and arithmetic operations.

(c) O(findLTMinProduct) = O($N^2$). This is because there are two nested loops that run N times each, and the only operations inside the loops are assignments and comparisons.

(d) O(subtractAndAdd) = O(N). This is because the while loop runs N times, and the only operations inside the loop are assignments and comparisons.

(e) O(search) = O(log N). This is because in this binary search the while loop runs log N times, and the only operations inside the loop are assignments and comparisons.

## 3. This one's bound to be fun! [18 points]

You are given the following bounds on functions $f$ and $g$:

- $f(x)$ is $O(203^x x^2)$ and $\Omega(3^x \log x)$
- $g(x)$ is $O(\frac{x!}{2^x})$ and $\Omega(4^x)$

Find the following, simplify your answer as much as possible.

(a) Find the tightest big-O and big-$\Omega$ estimates that can be *guaranteed* of $f(x)(g(x))^2$.

(b) Find the tightest big-O and big-$\Omega$ estimates that can be *guaranteed* of $f(x) + g(x)$.

(c) Let $h(x) = f(x) - g(x)$. Prove or disprove that $h(x)$ is $\Omega(4^x)$.

**Solution:**

$\Omega$ is a lower bound, so we can use the lower bound of $f(x)$ and $g(x)$ to find the lower bound of $h_n(x)$.
$O$ is an upper bound, so we can use the upper bound of $f(x)$ and $g(x)$ to find the upper bound of $h_n(x)$.

(a) Let $h_1(x)$ be $f(x)(g(x))^2$.
Since $f(x)$ is $O(203^x x^2)$ and $g(x)$ is $O(\frac{x!}{2^x})$, we have that $h_1(x)$ is $O(203^x x^2 \left(\frac{x!}{2^x}\right)^2)$.
Since $f(x)$ is $\Omega(3^x \log x)$ and $g(x)$ is $\Omega(4^x)$, we have that $h_1(x)$ is $\Omega(3^x \log x \,(4^x)^2)$.
Thus, the tightest big-O estimate for $h_1(x)$ is $O(203^x x^2 \left(\frac{x!}{2^x}\right)^2)$ and the tightest big-$\Omega$ estimate for $h_1(x)$ is $\Omega(3^x \log x \,(4^x)^2)$.

(b) Let $h_2(x)$ be $f(x) + g(x)$.
Since $f(x)$ is $O(203^x x^2)$ and $g(x)$ is $O(\frac{x!}{2^x})$, we have that $h_2(x)$ is $O(203^x x^2 + \frac{x!}{2^x})$.
Since $f(x)$ is $\Omega(3^x \log x)$ and $g(x)$ is $\Omega(4^x)$, we have that $h_2(x)$ is $\Omega(3^x \log x + 4^x)$.
Thus, the tightest big-O estimate for $h_2(x)$ is $O(203^x x^2 + \frac{x!}{2^x})$ and the tightest big-$\Omega$ estimate for $h_2(x)$ is $\Omega(3^x \log x + 4^x)$.

(c) Disproof by contradiction:
Consider $f(x) = 3^x$ and $g(x) = 4^x$.
Then $h(x) = f(x) - g(x) = 3^x - 4^x$.
Assume $h(x)$ is $\Omega(4^x)$.
Then there exists $c > 0$ and $x_0 > 0$ such that $h(x) \geq c \cdot 4^x$ for all $x \geq x_0$.
However, $3^x - 4^x < 0$ for all $x > 0$.
Thus, $h(x)$ is not $\Omega(4^x)$.

## 4. Big Function Fun [16 points]

Prove or disprove the following:

(a) If $f(x)$ is $O(g(x))$ then $2^{f(x)}$ is $O(2^{g(x)})$.

(b) If $f(x)$ is $O(g(x))$ then $(f(x))^2$ is $O\big((g(x))^2\big)$.

Note that in these proofs you do not need to use the definition of big-O, but can use the properties for combining mathematical functions covered in lecture.

**Solution:**

(a)

(b)

## 5. Roots and Shoots [16 points]

Suppose $f$ satisfies $f(n) = 2f(\sqrt{n}) + \log_2 n$, whenever $n$ is a perfect square greater than 1, and additionally satisfies $f(2) = 1$.

(a) Find $f(16)$.

(b) Find a big-O estimate for $g(m)$ where $g(m) = f(2^m)$.
**Hint:** Make the substitution $m = \log_2 n$.

(c) Find a big-O estimate for $f(n)$.

**Solution:**

(a) We can divide the recurance relation into smaller parts:

$$\begin{aligned} f(16) &= 2f(\sqrt{16}) + \log_2 16 \\ &= 2f(4) + 4 \\ &= 2(2f(2) + 2) + 4 \\ &= 2(2(1) + 2) + 4 \\ &= 2(4) + 4 \\ &= 8 + 4 \\ &= 12 \end{aligned}$$

Thus, $f(16) = 12$.

(b) Let $m = \log_2 n$. Then $n = 2^m$. We can rewrite the recurrence relation as:

$$\begin{aligned} f(2^m) &= 2f(\sqrt{2^m}) + \log_2 2^m \\ &= 2f(2^{m/2}) + m \end{aligned}$$

We can see that $f(2^m) = 2f(2^{m/2}) + m$. To rewrite in terms of $g(m)$, we can substitute $g(m) = f(2^m)$:

$$g(m) = 2g(m/2) + m$$

By the Master Theorem with $a = 2$, $b = 2$, $d = 1$, and $f(n) = n$, $g(m) = O(m \log m)$ since $\frac{a}{b^d} = \frac{2}{2} = 1$.

(c) Using the same substitution, since $m = \log_2 n$ and $g(m) = O(m \log m)$, we have that $f(n) = O(\log_2 n \log \log_2 n)$.

## 6. GG Brown Laboratory [15 points]

What is the tightest big-O bound on the runtime complexity of the following algorithm?

```
function BADSEARCH(n)
    if n ≥ 1 then
        BADSEARCH(⌊n/3⌋)
        for i := 1 to n do
            for j := 1 to ⌊n/2⌋ do
                print "Hello I am lost"
            end for
        end for
        BADSEARCH(⌊n/3⌋)
        print "Nevermind I got it"
    end if
end function
```

**Solution:**

This is a recursive algorithm that calls itself twice with $\frac{n}{3}$ as the argument. The outer loop runs $n$ times and the inner loop runs $\frac{n}{2}$ times. The print statement runs $\frac{n^2}{2}$ times. Thus, we can write this as a recurrence relation:

$$T(n) = 2T\left(\frac{n}{3}\right) + \frac{n^2}{2}$$

Using the Master Theorem, we can see that $a = 2$, $b = 3$, and $f(n) = \frac{n^2}{2}$. Since $\log_b a = \log_3 2 \approx 0.63$ and $f(n) = O(n^c)$ where $c = 2$, we have that $T(n) = O(n^2)$. The runtime complexity of this algorithm is $O(n^2)$.