University of Waterloo
CS 341 — Algorithms
Spring 2014
Assignment 1 Solutions

1. *Show formally, from first principles, (using the definitions of O, Ω, Θ), that $6n(3n-1)(19n+2)$ is $\Theta(n^3)$.*

$$
\begin{aligned}
&f(n) = 6n(3n-1)(19n+2) = 342n^3 - 78n^2 - 12n \\
&\text{Since } n^3 \geq n^2 \text{ and } n^3 \geq n, \forall n \geq 1 \\
&\quad 342n^3 - 78n^2 - 12n \geq 342n^3 - 78n^3 - 12n^3 = 252n^3, \forall n \geq 1 \\
&\quad \text{so choose } c = 252, n_0 = 1 \Rightarrow f(n) \in \Omega(n^3) \\
&342n^3 - 78n^2 - 12n \leq 342n^3, \forall n \geq 1 \\
&\quad \text{so choose } c = 342, n_0 = 1 \Rightarrow f(n) \in O(n^3) \\
&\Rightarrow f(n) \in \Theta(n^3)
\end{aligned}
$$

2. *Prove the following relationships:*

   (a) 2^n *is* $o(3^{n-1})$

$$\lim_{n\to\infty} \frac{2^n}{3^{n-1}} = \lim_{n\to\infty} \frac{3 \cdot 2^n}{3^n} = 3 \cdot \lim_{n\to\infty} \left(\frac{2}{3}\right)^n = 3 \cdot 0 = 0$$

   Thus $2^n \in o(3^{n-1})$

   (b) $\log^2 n$ *is* $O(\sqrt{n})$

$$
\begin{aligned}
\lim_{n\to\infty} \frac{\log^2 n}{\sqrt{n}} &= \lim_{n\to\infty} \frac{\frac{2\log n}{n}}{\frac{1}{2\sqrt{n}}} = \lim_{n\to\infty} \frac{4\log n}{\sqrt{n}} = 4 \cdot \lim_{n\to\infty} \frac{\log n}{\sqrt{n}} \\
&= 4 \cdot \lim_{n\to\infty} \frac{\frac{1}{n}}{\frac{1}{2\sqrt{n}}} = 8 \cdot \lim_{n\to\infty} \frac{1}{\sqrt{n}} = 8 \cdot 0 = 0
\end{aligned}
$$

   Thus $\log^2 n$ is $o(\sqrt{n})$, so $\log^2 n \in O(\sqrt{n})$

   (c) *Is* $log^2 n \in \Theta(\sqrt{n})$*? Briefly justify your answer.*

   No. From part b), since $\log^2 n \in o(\sqrt{n})$, $O(\sqrt{n})$ is **not** a tight bound, so $\log^2 n \notin \Theta(\sqrt{n})$

3. *Analyze the following two pieces of pseudocode and for each of them give a tight (Θ) bound on the running time as a function of n. Show your work and justify each step*

   Note: `lg(n)` *is defined as the discrete logarithm base 2, i.e. the floor of* $log_2(n) + 1$.

(a)
```
x = 0
for i = 1 to ceiling(lg(n))
   for j = 1 to i
      for k = 1 to 10
         x = x + 1
```

$$\sum_{i=1}^{\lceil \log n \rceil} 10i = 10 \cdot \sum_{i=1}^{\lceil \log n \rceil} i = 10 \cdot \left(\frac{\lceil \log n \rceil}{2} + \frac{\lceil \log n \rceil^2}{2} \right) \in \Theta\left((\log n)^2\right)$$

(b)
```
x = 0
for i = 1 to sqr(n)     \\ i.e. n^2
   for j = 1 to ceiling(lg(i))
      x = x + 1
```

$$\sum_{i=1}^{n^2} \lceil \log i \rceil \leq \sum_{i=1}^{n^2} (\log i + 1) \leq n^2 \log n^2 + n^2 = 2n^2 \log n + n^2 \in O\left(n^2 \log n\right)$$

$$\sum_{i=1}^{n^2} \lceil \log i \rceil \geq \sum_{i=1}^{n^2} \log i \geq \sum_{i=\frac{n^2}{2}+1}^{n^2} \log i \geq \sum_{i=\frac{n^2}{2}+1}^{n^2} \log \frac{n^2}{2} \geq \frac{n^2}{2} \log \frac{n^2}{2} \in \Omega\left(n^2 \log n\right)$$

$$\therefore \sum_{i=1}^{n^2} \lceil \log i \rceil \in \Theta\left(n^2 \log n\right)$$

4. *Programming Question - write the following programs in C++.*

   Given an array $A = A[1 \ldots n]$ consisting of n distinct integers and an integer S, determine the number OP of ordered pairs (i, j) such that $A[i] + A[j] = S$.

   The input to your program will be from `stdin`*, in the following order: S, n, followed by the contents of array A starting at the first index. One number per line.*

   The output of your program, OP, should be sent to `stdout`*.*

   (a) *Implement an algorithm that computes OP in $\Theta(nlogn)$.*

```
#include <algorithm>
#include <iostream>
using namespace std;

int main(int argc, char** argv) {
    int s, n;
    cin >> s;
```

```
    cin >> n;

    int a[n];
    for (int i = 0; i < n; ++i) cin >> a[i];

    // sort the input, O(n lg n)
    sort(a, a+n);

    // search for pairs, O(n lg n)
    int op = 0;
    for (int i = 0; i < n; ++i) {
        // binary search for rest of pair, O(lg n)
        if (binary_search(a, a+n, s-a[i])) ++op;
    }

    cout << op << endl;
}
```

(b) *Assuming that the array is sorted (in increasing order), implement an algorithm that computes OP in $\Theta(n)$.*

```
#include <iostream>
using namespace std;

int main(int argc, char** argv) {
    int s, n;
    cin >> s;
    cin >> n;

    int a[n];
    for (int i = 0; i < n; ++i) cin >> a[i];

    // scan for pairs, O(n)
    int op = 0;
    for (int i = 0, j = n-1; i < n && j >= 0; ++i) {
        while (a[i] + a[j] > s && j > 0) --j;
        if (a[i] + a[j] == s) ++op;
    }

    cout << op << endl;
}
```