CS 341, Spring 2020
Semih Salihoglu

Assignment 1 Part 1 (due Sunday, May 24, midnight EST)

Instructions:

- Hand in your assignment using Crowdmark. Detailed instructions are on the course website.
- Give complete legible solutions to all questions.
- Your answers will be marked for clarity as well as correctness.
- For any algorithm you present, you should justify its correctness (if it is not obvious) and analyze the complexity.

1. [*8 marks*] Give a proof from first principles (not using limits) the following statements:

   (a) [*4 marks*]
   $$n^{2.7} - 100n^{2.4} + 1000 \in \omega(n^{2.5})$$

   We want to prove for all constant $c > 0$, there exists a n_0 such that $|n^{2.7} - 100n^{2.4} + 1000| > |n^{2.5}|$ for all $n > n_0$
   If $c > 1$
   Let $n_0 = 10^{10}c^{10}$ and $n \geq n_0$

   $$\begin{aligned}
   |n^{2.7} - 100n^{2.4} + 1000| &= n^{2.5}(n^{0.2} - \frac{100}{n^{0.1}} + \frac{1000}{n^{2.5}}) \\
   &\geq n^{2.5}(10^2c^2 - \frac{100}{n^{0.1}} + \frac{1000}{n^{2.5}}) \\
   &\geq n^{2.5}(10^2c^2 - \frac{100}{10c} + \frac{1000}{n^{2.5}}) \\
   &= n^{2.5}(10^2c^2 - \frac{10}{c} + \frac{1000}{n^{2.5}}) \\
   &> n^{2.5}(10^2c^2 - \frac{10}{c} + 1) \text{ since } \frac{1000}{n^{2.5}} \leq \frac{1000}{10^{25}c^{25}} = \frac{1}{10^{22}c^{25}} < 1 \\
   &> cn^{2.5} \\
   &> 0
   \end{aligned}$$

   If $0 < c \leq 1$ Let $n_0 = 10^{10}$ and $n \geq n_0$

   $$\begin{aligned}
   n^{2.7} - 100n^{2.4} + 1000 &= n^{2.5}(n^{0.2} - \frac{100}{n^{0.1}} + \frac{1000}{n^{2.5}}) \\
   &\geq n^{2.5}(10^2 - \frac{100}{10} + \frac{1000}{n^{2.5}}) \\
   &< n^{2.5}(100 - 10 + 1) \text{ since } \frac{1000}{n^{2.5}} \leq \frac{1}{10^{22}} < 1 \\
   &= 91n^{2.5} \\
   &> c \\
   &> 0
   \end{aligned}$$

Hence, $n^{2.7} - 100n^{2.4} + 1000 \in \omega(n^{2.5})$

(b) [*4 marks*] Let $f(n)$ and $g(n)$ be positive-valued functions. Then:

$$\max\{f(n), g(n)\} = \Theta(f(n) + g(n))$$

We want to prove that there exists $c_1 > 0, c_2 > 0$ such that

$$c_1|(f(n) + g(n))| \leq |\max\{f(n), g(n)\}| \leq |c_2(f(n) + g(n))|$$

for all $n \leq n_0$ Let $c_1 = \frac{1}{2}, c_2 = 1$

$$2\max\{f(n), g(n)\} \leq f(n) + g(n)$$
$$\text{Hence, } \max\{f(n), g(n)\} \leq \frac{1}{2}(f(n) + g(n)) \text{ for all n}$$

If $\max\{f(n), g(n)\} = f(n)$,
$\max\{f(n), g(n)\} = f(n) < f(n) + g(n)$ since $g(n) > 0$ for all n
If $\max\{f(n), g(n)\} = g(n)$,
$\max\{f(n), g(n)\} = g(n) < f(n) + g(n)$ since $f(n) > 0$ for all n
Therefore $\max\{f(n), g(n)\} = \Theta(f(n) + g(n))$

2. [*12 marks*] For each pair of functions $f(n)$ and $g(n)$, fill in the correct asymptotic notation among Θ, o, and ω in the statement $f(n) \in \sqcup\, (g(n))$. Formal proofs are not necessary, but provide brief justifications for all of your answers. (The default base in logarithms is 2.)

(a) $f(n) = (8n)^{250} + (3n + 1000)^{500}$ vs. $g(n) = n^{500} + (n + 1000)^{400}$
$f(n) \in \Theta(g(n))$
$\lim_{n\to\infty} \frac{f(n)}{g(n)} = 3^{500}$

(b) $f(n) = n^{1.5}2^n$ vs. $g(n) = (n)^{100}1.99^n$.
$f(n) \in \omega(g(n))$

$$\lim_{n\to\infty} \frac{n^{1.5}2^n}{n^{100}1.99^n} = \frac{(\frac{2}{1.99})^n}{n^{98.5}} \stackrel{L'H}{=} \infty$$

(c) $f(n) = (256)^{n/4}$ vs. $g(n) = (125)^{n/3}$
$f(n) \in o(g(n))$

$$\lim_{n\to\infty} \frac{256^{n/4}}{125^{n/3}} = (\frac{4}{5})^n = 0$$

(d) $f(n) = 2^{\log(n)\cdot\log(n)}$ vs. $g(n) = n^{2012}$
$f(n) \in \omega(g(n))$
$f(n) = 2^{\log(n)\cdot\log(n)} = n^{\log(n)}$
$\lim_{n\to\infty} \frac{n^{\log(n)}}{n^{2012}} > \lim_{n\to\infty} \frac{n^{2013}}{n^{2012}} = \lim_{n\to\infty} n = \infty$

3. [*10 marks*] Analyze the following pseudocodes and give a tight Θ bound on the running time as a function of n. Carefully show your work.

   (a) [*5 marks*]

```
for i = 1 to n do
    A[i] = true
for i = 1 to n do
    j = i
    while j ≤ n do
        A[j] = false
        j = j + i
```

$$\begin{aligned} T(n) &= n + \sum_{i=1}^{n} \lfloor \frac{2n}{i} \rfloor \\ &\leq n + \sum_{i=1}^{n} \frac{2n}{i} \\ &= n + 2n\Theta(\log n) \\ &\in O(n \log n) \end{aligned}$$

Also,

$$\begin{aligned} T(n) &= n + \sum_{i=1}^{n} \lfloor \frac{2n}{i} \rfloor \\ &\geq n + \sum_{i=1}^{n} \frac{1}{i} \\ &= n + n\Theta(\log n) \\ &\in \Omega(n \log n) \end{aligned}$$

Hence, $T(n) \in \Theta(n \log n)$

   (b) [*5 marks*] The following is a sorting algorithm that sorts an array A of n integers, where each integer $e_i \in A$ is $0 \leq e_i \leq m-1$. Go through the code and verify that this algorithm indeed sorts A correctly.

```
for i = 0 to m − 1 do
    counts[i] = 0
for i = 0 to n − 1 do
    counts[A[i]] + +
k = 0
for i = 0 to m − 1 do
    for j = 0 to counts[i] − 1 do
        A[k] = i, k = k + 1
```

line 2 execuates m times, line 4 execuates n times, line 8 execuates exactly n time since A has size n.
Putiing all together,

$$\begin{aligned} T(n) &= m + n + 1 + n \\ &\in \Theta(\max(m, n)) \end{aligned}$$

4. [*12 marks*] Given a string $s = a_1 a_2 ... a_n$ of length n, where $a_1 a_2 ... a_n \in \{0, 1\}$, decide whether s is the kth *power* of a sub-string t, i.e., $s = t^k$, for some $k > 1$ and string t. Here, t^k denotes the string t repeated k times. For example, 01000100, 10101010, and 000000, are all perfect powers (e.g. $01000100 = 0100^2$) but 01000110 is not.

   Give an algorithm that solves this problem in $O(n^{3/2})$ time. Describe your algorithm, provide the pseudocode, and analyze the run-time of your algorithm.

   Hint: Observe that if $s = t^k$, and t has length ℓ, then $n = \ell k$. This implies that ℓ and k cannot both be greater than $\sqrt{n}$.

IsPerfectPwoer(s) first loops through 2 to $\sqrt{n}$ to find all possible values of l that divides n. For each valid value l, it checks whether each sub-string with length l is actually equal. Then, it switches the value of l and k, i.e. let $l = n/l$ and do the exactly same process to check if all substring are equal.

Algorithm 1: IsPerfectPwoer(s)

```
1  for i = 2 to √n do
2      l = i
3      if l divides n then
4          for j = 1 to n/l − 1 do
5              if s[j × l ... (j + 1) × l] != s[0 ... l − 1] then
6                  break
7              if j == n/l − 1 then
8                  return True
9          l = n/l
10         for j = 1 to n/l − 1 do
11             if s[j × l ... (j + 1) × l] != s[0 ... l − 1] then
12                 break
13             if j == n/l − 1 then
14                 return True
15 return False
```

Run-time analysis: The outer loop(line 1-14) iterates at most $\sqrt{n}$ times. Both inner loops(line 4-8, 10-14) iterates at most $n/l - 1$ times.

Hence,

$$\begin{aligned}
T(n) &\leq \sqrt{n} \times 2(\frac{n}{l} - 1) \\
&\leq 2n^{\frac{3}{2}} - n^{\frac{1}{2}} \text{ , since } l \geq 1 \\
&\in O(n^{\frac{3}{2}})
\end{aligned}$$