Name: Kevina Wong
ID: 109179049

CSCI 3104, Algorithms **Profs. Chen & Grochow**
Problem Set 2 – Due Thurs Jan 30 11:55pm **Spring 2020, CU-Boulder**

Advice 1: For every problem in this class, you must justify your answer: show how you arrived at it and why it is correct. If there are assumptions you need to make along the way, state those clearly.

Advice 2: Informal reasoning is typically insufficient for full credit. Instead, write a logical argument, in the style of a mathematical proof.

Instructions for submitting your solutions:

- The solutions **should be typed** and we cannot accept hand-written solutions. Here's a short intro to LaTeX.
- You should submit your work through the **class Canvas page** only.
- You may not need a full page for your solutions; pagebreaks are there to help Gradescope automatically find where each problem is. Even if you do not attempt every problem, please submit this template of at least 9 pages (or Gradescope has issues with it).

Quicklinks: 1 2a 2b 2c 2d 3a 3b 3c

---

1. *Name (a) one advantage, (b) one disadvantage, and (c) one alternative to worst-case analysis. For (a) and (b) you should use full sentences.*

 - **(a) Advantage:** It provides an upper guarantee of running time and generally captures complexity in practice.
 - **(b) Disadvantage:** The worse case input may occur rarely.
 - **(c) Althernative:** Average Case Analysis

Name: $\boxed{\text{Kevina Wong}}$
ID: $\boxed{109179049}$

CSCI 3104, Algorithms **Profs. Chen & Grochow**
Problem Set 2 – Due Thurs Jan 30 11:55pm **Spring 2020, CU-Boulder**

2. *For each part of this question, put the growth rates in order, from slowest-growing to fastest. That is, if your answer is* $f_1(n), f_2(n), \ldots, f_k(n)$*, then* $f_i(n) < O(f_{i+1}(n))$ *for all* i*. If two adjacent ones are asymptotically the same (that is,* $f_i(n) = \Theta(f_{i+1}(n))$*), you must specify this as well.*

   Justify your answer (show your work). You may assume transitivity: if $f(n) < O(g(n))$ *and* $g(n) < O(h(n))$*, then* $f(n) < O(h(n))$*, and similarly for little-oh, etc.*

   (a) *Polynomials.*

$$n+1 \qquad n^4 \qquad 1/n \qquad 1 \qquad n^2+2n-4 \qquad n^2 \qquad \sqrt{n} \qquad 10^{100}$$

Answer:

$$1/n \;<\; 1 \;=\; 10^{100} \;<\; \sqrt{n} \;<\; n+1 \;<\; n^2 \;=\; n^2+2n+4 \;<\; n^4$$

The slowest growth rate is $1/n$. This is because the growth rate can be rewritten as n^{-1}. Rather than the function growing, it actually gets smaller and smaller.

Math Work Shown (2a):

- $\lim_{n\to\infty} \frac{1/n}{1} = 0$ because the limit can be rewritten as $\lim_{n\to\infty} \frac{1}{n} = 0$. Therefore, we can conclude that 1/n grows slower than 1.
- $\lim_{n\to\infty} \frac{1}{10^{100}} = \frac{1}{10^{100}} \neq 0$ and the limit L is a finite number. Therefore, they grow at the same rate.
- $\lim_{n\to\infty} \frac{10^{100}}{\sqrt{n}} = 0$. This limit can be rewritten as $10^{100} \times \lim_{x\to\infty} \frac{1}{\sqrt{n}} = 0$. Therefore, we can conclue that 10^{100} grows slower than $\sqrt{n}$.
- $\lim_{n\to\infty} \frac{\sqrt{n}}{n+1} = 0$. This limit can be rewritten as $\lim_{n\to\infty} \frac{n^{1/2}}{n+1} = 0$. Therefore, we can conclude that $\sqrt{n}$ grows slower than n+1.
- $\lim_{n\to\infty} \frac{n+1}{n^2} = 0$. Since the denominator has the highter term, we can conclude that this limit will be 0. Thus, n+1 grows slower than n^2.
- $\lim_{n\to\infty} \frac{n^2}{n^2+2n+4} = 1$. Since the terms are equal between the numerator and the denomator, the limit, L, is 1. Since L $\neq 0$ and is a finite number, we can conclude that the two have the same growth rate.
- $\lim_{n\to\infty} \frac{n^2+2n+4}{n^4} = 0$. Since the denominator has a higher term than the numerator, we can conclude that the limit is 0, and thus, conclude that n^2+2n+4 has a slower growth rate than n^4.

(b) *Logarithms and related functions.*

$$(\log_2 n)^2 \qquad \log_2(n) \qquad \log_3(n) \qquad \sqrt{n} \qquad \log_{1.5}(n) \qquad \log_2(n^2)$$

Answer:

$$\log_{1.5}(n) \;=\; \log_2(n) \;=\; \log_3(n) \;=\; \log_2(n^2) \;<\; (\log_2(n))^2 \;<\; \sqrt{n}$$

Math Work Shown (2b):

- $\lim_{n\to\infty} \frac{\log_{1.5}(n)}{\log_2(n)} = \frac{ln(1.5)}{ln(2)}$. L'Hopital's rule can be applied to the limit. $\lim_{n\to\infty} \frac{\log_{1.5}(n)}{\log_2(n)} = \lim_{n\to\infty} \frac{\frac{1}{nln(1.5)}}{\frac{1}{nln(2)}} = \lim_{n\to\infty} \frac{ln(2)}{ln(1.5)} = \frac{ln(2)}{ln(1.5)}$. The limit L $\neq 0$ and is also a finite number. Because of this, we can conclude that the two have the same growth rate.
- $\lim_{n\to\infty} \frac{\log_2(n)}{\log_3(n)} = \frac{ln(3)}{ln(2)}$. L'Hopital's rule can be applied to the limit. $\lim_{n\to\infty} \frac{\log_2(n)}{\log_3(n)} = \lim_{n\to\infty} \frac{\frac{1}{nln(2)}}{\frac{1}{nln(3)}} = \lim_{n\to\infty} \frac{ln(3)}{ln(2)} = \frac{ln(3)}{ln(2)}$. The limit L $\neq 0$ and is also a finite number. Because of this, we can conclude that the two have the same growth rate.
- $\lim_{n\to\infty} \frac{\log_3(n)}{\log_2(n^2)} = \frac{ln(2)}{2ln(3)}$. L'Hopital's rule can be applied to the limit. $\lim_{n\to\infty} \frac{\log_2(n)}{\log_3(n)} = \lim_{n\to\infty} \frac{\frac{1}{nln(2)}}{\frac{1}{nln(3)}} = \lim_{n\to\infty} \frac{ln(3)}{ln(2)} = \frac{ln(3)}{ln(2)}$. The limit L $\neq 0$ and is also a finite number. Because of this, we can conclude that the two have the same growth rate.
- $\lim_{n\to\infty} \frac{log_2(n^2)}{(log_2(n))^2} = 0$. L'Hopital's rule can be applied to this limit. $\lim_{n\to\infty} \frac{log_2(n^2)}{(log_2(n))^2} = \lim_{n\to\infty} \frac{\frac{2}{nln2}}{\frac{2log_2(n)}{nln2}} = \lim_{n\to\infty} \frac{1}{log_2(n)} = 0$. Because the limit is 0, we can conclude that $log_2(n^2)$ has a slower growth rate than $(log_2(n))^2$.
- $\lim_{n\to\infty} \frac{(log_2(n))^2}{\sqrt{n}} = 0$. L'Hopital's rule can be applied to this limit. $\lim_{n\to\infty} \frac{(log_2(n))^2}{\sqrt{n}} = \lim_{n\to\infty} \frac{\frac{2log_2(n)}{nln(2)}}{\frac{1}{2\sqrt{n}}}$. Then, L'Hopital's rule can be applied again. $\lim_{n\to\infty} \frac{\frac{2log_2(n)}{nln(2)}}{\frac{1}{2\sqrt{n}}} = \lim_{n\to\infty} \frac{4log_2(n)}{\sqrt{n}ln(2)} = \lim_{n\to\infty} \frac{8}{ln^2(2)\sqrt{n}} = 0$. Because the limit is 0, we can conclude that the growth rate of $(log_2(n))^2$ is slower than the growth rate of $\sqrt{n}$.

(c) *Logarithms in exponents.*

$$n^{\log_3(n)} \qquad n^{\log_2 n} \qquad n^{1/\log_2(n)} \qquad n \qquad 1$$

Answer:

$$n^{1/log_2(n)} \ = \ 1 \ < \ n \ < \ n^{log_3(n)} \ < \ n^{log_2(n)}$$

Math Work Shown (2c):

- $\lim_{n\to\infty} \frac{n^{1/log_2(n)}}{1} = 2$. This limit can be split up into two limits. $\lim_{n\to\infty} 1 = 1$ and $\lim_{n\to\infty} n^{1/log_2(n)} = 2$. Therefore, the limit, L, is $\frac{2}{1}$. L $\neq 0$ and is a finite number. From this, we can conclude that 1 and $n^{1/log_2(n)}$ have the same growth rate.
- $\lim_{n\to\infty} \frac{1}{n} = 0$. Therefore, we can conclude that the growth rate of 1 is slower than the growth rate of n.
- $\lim_{n\to\infty} \frac{n}{n^{log_3(n)}} = 0$. L'Hopital's rule can be applied to this limit. $\lim_{n\to\infty} \frac{n}{n^{log_3(n)}} = \lim_{n\to\infty} \frac{1}{\frac{2n^{log_3(n)-1}ln(n)}{ln(3)}} = ln(3) \lim_{n\to\infty} \frac{1}{2n^{log_3(n)-1}ln(n)} = 0$. Therefore, we can conclude the n has a slower growth rate than $n^{log_3(n)}$.
- $\lim_{n\to\infty} \frac{n^{log_3(n)}}{n^{log_2(n)}} = 0$. Due to exponent rules, this limit can be rewritten as $\lim_{n\to\infty} n^{log_3(n)-log_2(n)} = 0$. Since this limit evaluates to 0, we can conclude that the growth rate of $n^{log_3(n)}$ is smaller than the growth rate of $n^{log_2(n)}$.

Name: $\boxed{\text{Kevina Wong}}$
ID: $\boxed{109179049}$

CSCI 3104, Algorithms **Profs. Chen & Grochow**
Problem Set 2 – Due Thurs Jan 30 11:55pm **Spring 2020, CU-Boulder**

(d) *Exponentials.* Hint: *Recall Stirling's approximation, which says that* $n! \sim \left(\frac{n}{e}\right)^n \sqrt{2\pi n}$, *i.e.* $\lim_{n\to\infty} \frac{n!}{\left(\frac{n}{e}\right)^n \sqrt{2\pi n}} = 1$.

$$n! \qquad 2^n \qquad 2^{2n} \qquad 2^{n\log_2(n)} \qquad 2^{n+7}$$

Answer:

$$2^n \;=\; 2^{n+7} \;<\; 2^{2n} \;<\; n! \;<\; n^{nlog_2 n}$$

Math Work Shown (2d):

- $\lim_{n\to\infty} \frac{2^n}{2^{n+7}} = \frac{1}{128}$. The exponent rule can be applied to this limit. $\lim_{n\to\infty} \frac{2^n}{2^{n+7}} = \lim_{n\to\infty} \frac{1}{2^{n+7-n}} = \lim_{n\to\infty} \frac{1}{2^7} = \frac{1}{128}$. Therefore, since the limit, L, is a contant and $\neq 0$, we can conclude that 2^n and 2^{n+7} have the same growth rate.
- $\lim_{n\to\infty} \frac{2^{n+7}}{2^{2n}} = 0$. The exponent rule can be applied to this limit. $\lim_{n\to\infty} \frac{1}{2^{2n-(n+7)}} = \frac{1}{2^{n-7}} = 0$. Because the limit evaluates to 0, we can conclude that 2^{n+7} has a slower growth rate than 2^{2n}.
- $\lim_{n\to\infty} \frac{2^{2n}}{n!} = 0$. This limit can be simplified to $\lim_{n\to\infty} \frac{2^{2n}}{n!} = \lim_{n\to\infty} \frac{4^n}{n!}$. From here, this limit can be represented as $\frac{4\times4\times4...4\times4...}{1\times2\times3\times4...(n-1)\times(n)}$. Based on this, we can see that the denominator grows significantally faster than the numerator, which allows us to conclude that the limit is 0, and further conclude that 2^{2n} has a slower growth rate that $n!$.
- $\lim_{n\to\infty} \frac{n!}{n^{nlog_2(n)}} = 0$. Stirling's approximation can be applied to this problem. $\lim_{n\to\infty} \frac{n!}{n^{nlog_2(n)}} = \lim_{n\to\infty} \frac{(\frac{n}{e})^n\sqrt{2\pi n}}{n^{nlog_2(n)}}$. Then, the constants can be taken outside of the limit to make the evalation simpler. $\sqrt{2\pi}\lim_{n\to\infty} \frac{(\frac{n}{e})^n\sqrt{n}}{n^{nlog_2(n)}}$. Then, using exponent rules, parts of the numerator can be moved to the denomator.$\sqrt{2\pi}\lim_{n\to\infty} \frac{1}{e^n n^{nlog_2(n)-n-1/2}}$. We can then conclude that this limit evalutes to 0, and thus conclude that the growth rate of $n!$ is slower than the growth rate of $n^{nlog_2(n)}$.

3. *For each of the following algorithms, analyze the worst-case running time. You should give your answer in big-Oh notation. You* do not *need to give an input which achieves your worst-case bound, but you should try to give as tight a bound as possible.*

   Justify your answer (show your work). This likely means discussing the number of atomic operations in each line, and how many times it runs, writing out a formal summation for the runtime complexity $T(n)$ of each algorithm, and then simplifying your summation.

   (a)
```
1  f(A): // A is a square, 2D array; indexed starting from 1
2    let d be a copy of A
3    for i = 1 to len(A):
4      d[i][i] = 0
5
6    for i = 1 to len(A):
7      for j = 1 to len(A):
8        for k = 1 to len(A):
9          if (d[i][k] + d[k][j]) < d[i][j]:
10           d[i][j] = d[i][k] + d[k][j]
11
12   return d
```

Answer:

Line Number	Cost	Time
2	c_1	1
3	c_2	n+1
4	c_3	n
6	c_4	n+1
7	c_5	$\sum_{i=1}^{n}(n+1) = n(n+1) = n^2+n$
8	c_6	$\sum_{j=1}^{n}\sum_{i=1}^{n}(n+1) = n^2(n+1) = n^3+n^2$
9	c_7	$\sum_{j=1}^{n}\sum_{i=1}^{n}(n) = n^3$
10	c_8	$\sum_{j=1}^{n}\sum_{i=1}^{n}(n) = n^3$
12	c_9	1

Worst Case Running Time:

$T(n) = (c_6+c_7+c_8)(n^3) + (c_5+c_6)(n^2) + (c_2+c_3+c_4+c_5)(n) + (c_1+c_2+c_4+c_9)$
$=$
$\mathcal{O}(n^3)$

(b)
```
1 g(A): // A is a list of integers
2   for i = 1 to len(A):
3     for j = 1 to len(A)-i:
4       if A[j+1] > A[j]:
5         // swap A[j+1] and A[j]
6         tmp = A[j+1]
7         A[j+1] = A[j]
8         A[j] = tmp
9   return A
```

Answer:

Line Number	Cost	Time
2	c_1	n+1
3	c_2	$\sum_{i=1}^{n}(n+1-i) = \frac{n^2+n}{2}$
4	c_3	$\sum_{i=1}^{n}(n-i) = \frac{n^2-n}{2}$
6	c_4	$\sum_{i=1}^{n}(n-i) = \frac{n^2-n}{2}$
7	c_5	$\sum_{i=1}^{n}(n-i) = \frac{n^2-n}{2}$
8	c_6	1

Worst Case Running Time:

$T(n) = (c_2 + c_3 + c_4 + c_5)(\frac{n^2}{2}) + (c_2)(\frac{n}{2}) + (c_3 + c_4 + c_5)(\frac{-n}{2}) + (c_1)(n) + (c_1 + c_6)$
$=$
$\mathcal{O}(n^2)$

(c) Here, `abs(n)` returns the absolute value of its argument, and can be treated as an atomic operation

```
1 h(A): // A is a list of integers, of length at least 2, first index is 1
2   min = abs(A[1] - A[2])
3   for i = 1 to len(A):
4     for j = i+1 to len(A):
5       if abs(A[i] - A[j]) < min:
6         min = abs(A[i] - A[j])
7   return min
```

Answer:

Line Number	Cost	Time
2	c_1	1
3	c_2	n+1
4	c_3	$\sum_{i=1}^{n}\sum_{j=i+1}^{n+1} 1 = \frac{n^2+n}{2}$
5	c_4	$\sum_{i=1}^{n}\sum_{j=i+1}^{n} 1 = \frac{n^2-n}{2}$
6	c_5	$\sum_{i=1}^{n}\sum_{j=i+1}^{n} 1 = \frac{n^2-n}{2}$
7	c_6	1

Worst Case Running Time:

$T(n) = (c_3 + c_4 + c_5)(\frac{n^2}{2}) + (c_2)(n) + (c_2)(\frac{n}{2}) + (c_4 + c_5)(\frac{-n}{2}) + (c_2 + c_3 + c_6)$
$=$
$\mathcal{O}(n^2)$

References:

- Office hours with other stdudents and CA.
- Prof. Chen's Week 2 notes.