CS 6363.005: Design and Analysis of Algorithms
Exam #1.B September 30, 2019
Professor D.T. Huynh

Student Name:	KEY

General Remarks. This exam comprises 4 problems:

Problem ♯1 is assigned 25 points,

Problem ♯2 is assigned 25 points,

Problem ♯3 is assigned 25 points, and

Problem ♯4 is assigned 25 points.

Thus, the maximum score is 100 points.

Unless explicitly stated, *no correctness proofs* are required for your algorithms and (time) complexity means worst-case complexity.

Provide clean answers on the exam booklet. Use aditional paper only when necessary.

This is a **closed-book** exam

Exam time: 10:00 – 11:20 am

Good Luck!

♯1	♯2	♯3	♯4	Total

Problem ♯ 1.

1. Compare the order of magnitude of the following pair of functions. In each case determine whether $f(n) = o(g(n)), f(n) = \omega(g(n))$. Justify your answers!

   (a) $f(n) = 100n^{1.02} + lgn, \quad g(n) = 200n^{1.01} lg^k n$ where $k > 0$ is a fixed integer.

   Your justification: $f(n) = \Theta(n^{1.02}) \quad g(n) = \Theta(n^{1.01} lg^k n)$

   5

   $\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{n^{1.02}}{n^{1.01} lg^k n} = \lim_{n\to\infty} \frac{n^{.01}}{lg^k n} = \infty$

   Your answer: $f(n) =$ $\omega(g(n))$

   (b) $f(n) = (nlgn)^{lgn}, \quad g(n) = (lgn)^n$

   Your justification: $lg(f(n)) = lgn(lgn + lglgn)$; $lg(g(n)) = n.lglgn$
   $= \Theta(lg^2 n)$ $\qquad = \Theta(nlglgn)$

   5

   $\Rightarrow \lim_{n\to\infty} \frac{f(n)}{g(n)} = lg^2 n / n.lglgn = 0$

   Your answer: $f(n) =$ $o(g(n))$

2. (a) State the Master Theorem , and (b) use it to derive a tight bound for $T(n) = 5T(n/3) + n^2$.

   (State the Master Theorem on back page and do part (b) here)

   (b) *Show your work here:* $a = 5, b = 3, \quad f(n) = n^2$

   $n^{\log_3 5} = n^{1.\square} \Rightarrow f(n) = \Omega(n^{\log_3 5 + \varepsilon})$

   7

   Also: $af(\frac{n}{b}) = 5(\frac{n}{3})^2 = \frac{5}{9}n^2$
   $\leq cn^2$ where $c = \frac{5}{9} < 1$

   $\Rightarrow$ Case (3)

   Your answer: $T(n) =$ $\Theta(n^2)$

Statement of Master Theorem : see class notes

8

Problem ♯ 2. (Maximum Selection Sort)

```
Max-Selection (A[1..n])
  j = n
  while j > 1
        k = j; max = j; S = A[j]
        while k > 1
              k = k − 1
              if A[k] > S then
                    max = k;  S = A[k]
      swap A[max] with A[j]
      j = j − 1
```

$j = n, n-1, \dots, 2$

$k = j, \dots, 1 \rightarrow \Theta(j)$

$\Longrightarrow \sum_{j=2}^{n} \Theta(j) = \Theta(n^2)$

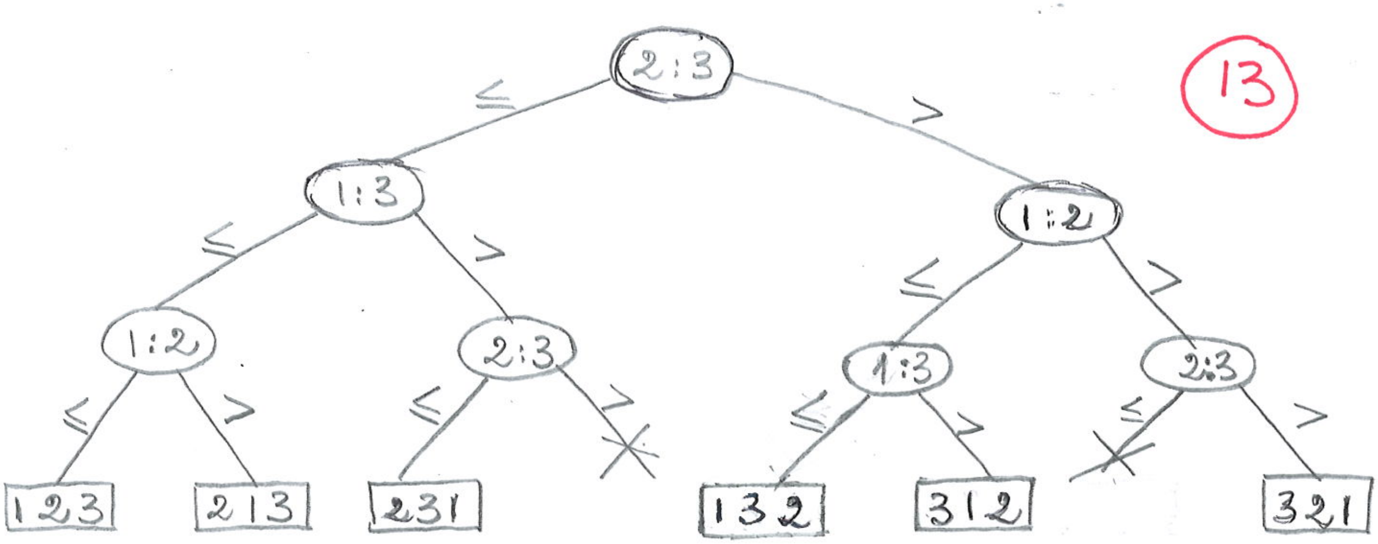
7

1. Analyze the worst-case complexity of your algorithm using the Θ notation. On what kind of arrays does Max-Selection perform worst?

Max - Selection works equally worst on all arrays

5

2. Draw the decision tree obtained from Max-Selection for an array $A[1..3]$ of size 3.



13

Problem ♯3 (Linear Time Selection)

1. Describe the linear selection algorithm.

See class notes

10

2. The linear time selection algorithm is modified by dividing the input array A into $\lceil n/9 \rceil$ groups of 9 elements each.

   For this modified version of Selection let x denote the *median* of the $\lceil n/9 \rceil$ medians of the 9-element groups. Derive (a) a lower bound for the number of elements in A that are greater than x, (b) an upper bound for the number of elements in A that are less than x, and (c) a recurrence for the running time. And, show that (d) the modified version of Select runs in linear time.

   (a)

3

$$5\left(\left\lceil \frac{1}{2}\left\lceil \frac{n}{9} \right\rceil \right\rceil - 2\right) \geq \frac{5n}{18} - 10$$

   (b)

$$\leq n - \left(\frac{5n}{18} - 10\right) = \frac{13n}{18} + 10$$

3

   (c)

$$T(n) \leq \begin{cases} O(1) & \text{for small } n \\ T\left(\left\lceil \frac{n}{9} \right\rceil\right) + T\left(\frac{13n}{18} + 10\right) + O(n) & \text{for suff. large } n \end{cases}$$

3

   (d)

Claim $T(n) \leq cn$ for some c and large n

Pf.

$$\begin{aligned} T(n) &\leq c \cdot \left\lceil \frac{n}{9} \right\rceil + c\left(\frac{13n}{18} + 10\right) + c_1 n \\ &\leq c\frac{n}{9} + c + \frac{13cn}{18} + 10c + c_1 n \\ &= \frac{15cn}{18} + 11c + c_1 n \\ &= cn - \underbrace{\left(\frac{3cn}{18} - c_1 n - 11c\right)}_{\geq 0 \text{ for suff. large } c} \\ &\leq cn \end{aligned}$$

6

Problem ♯4 (Rod Cutting)

1. Describe the $O(n^2)$ dynamic programming algorithm for rod cutting and perform the last iteration to calculate the optimal revenue for the following rod of length 7: (Display all steps of the last iteration!)

i	1	2	3	4	5	6	7
$p(i)$	1	4	7	8	10	13	15
$r(i)$	1	4	8	9	11	16	17
			7	8		14	15

$i=1$: $p[1]+r[6] = 1+16 = 17$ ✓

14 15 ✓

$i=2$: $p[2]+r[5] = 4+11 = 15$

$i=3$: $p[3]+r[4] = 7+9 = 16$

8 15

$i=4$: $p[4]+r[3] = 8+8 = 16$

7 15

$i=5$: $p[5]+r[2] = 10+4 = 14$

$i=6$: $p[6]+r[1] = 13+1 = 14$

$i=7$: $p[7]+r[0] = 15+0 = 15$

5

Algorithm

n = rod length

$p[1..n]$ = prices

$r[0..n]$ = revenues

$r[0] = 0$

for $j=1$ to n

$$r[j] = \underset{1 \le i \le j}{\text{Max}} \{p[i]+r[j-i]\}$$

10

2. Suppose each cut operation incurs a cost of c. In this case the revenue associate with each solution is (the sum of the prices of the pieces) minus ($c \times$ the number of cut operations). Give a dynamic programming algorithm to solve this modified problem.

As above:

n = rod length

$p[1..n]$ = prices

$r[0..n]$ = revenues

c = cost of a cut operation

$r[0]=0$; $r[1] = p[1]$

10

for $j=2$ to n

$$r[j] = \text{Max}\left[\bigcup_{1\le i \le j-1} \{p[i]+r[j-i]-c\} \cup \{p[j]\} \right]$$

cost of cut at location i

whole piece of length j (no cut)