CS 374 HW 4 Problem 1

quddus2, Hieu Huynh, Aldo Sanjoto

TOTAL POINTS

100 / 100

QUESTION 1

1 1A **25 / 25**

✓ **- 0 pts** Correct

- 25 pts Logic impossible to follow, or completely erroneous answer

- 15 pts Failing to specifically argue that each successive quarter of the array ends up in the right sub-array (without mentioning it)

- 10 pts Failing to argue in detail that each successive quarter of the array ends up in the right sub-array (but do mention it)

- 10 pts Failing to set up a correct induction proof

- 5 pts Minor flaws or typos in the proof

- 18.62 pts IDK

QUESTION 2

2 1B **25 / 25**

✓ **- 0 pts** Correct

- 25 pts Logic impossible to follow, or completely erroneous answer

- 5 pts Slight mistake or typo

- 18.75 pts IDK

- 0 pts Missing constant work factor when n > 16

QUESTION 3

3 1C **25 / 25**

✓ **- 0 pts** Correct

- 25 pts Logic impossible to follow, or completely erroneous answer

- 0 pts Failing to account for the (constant) work at every node / level of recursion.

- 15 pts Stating an incorrect (non-constant) amount of work at each node / level of recursion.

- 5 pts Concluding a runtime other than 6^(log2 n), but otherwise essentially correct

- 18.75 pts IDK

QUESTION 4

4 1D **25 / 25**

✓ **- 0 pts** Correct

- 25 pts Logic impossible to follow, or completely erroneous answer

- 10 pts Failing to mention that the no-repeated-swapping property is a result of insertion sort

- 15 pts Failure to argue that no pair of entries can be swapped more than once

- 5 pts Almost correct, but minor flaw or typo

- 18.75 pts IDK

Q1)

a) Induction on length of the array. (n)

- Base case: $2 \leq n \leq 16$, and $n = 2^k$.

  The array is sorted by the InsertionSort

  $\Rightarrow$ The base case is correct.

- IH: Assume the algorithm is correct for array of size $n \leq 2^k$ (with $k > 0$).

- Let $n = 2^{k+1} = 2 \cdot 2^k$.

  The Algorithm calls bogiSort on sub-array of size 2^k six times: $(i=0; j=2)$; $(i=0, j=1)$; $(i=0; j=0)$; $(i=1; j=2)$; $(i=1; j=1)$; $(i=2; j=2)$

- For $i=0; j=2$: The call to BogiSort $(A[\frac{n}{2}, \dots, (n-1)])$ sorted the ~~pa~~ sub-array $A[\frac{n}{2}, \dots (n-1)]$, by IH.

  $\Rightarrow$ The $\frac{n}{4}$ smallest values of $A[\frac{n}{2}, \dots, (n-1)]$ is located in $A[\frac{n}{2}, \dots, (\frac{3n}{4} - 1)]$

- For $i=0; j=1$: The call to BogiSort $(A[\frac{n}{4}, \dots (\frac{3n}{4} - 1)])$ sorted the subarray $A[\frac{n}{4}, \dots, (\frac{3n}{4} - 1)]$, by IH

  $\Rightarrow$ The $\frac{n}{4}$ smallest values of $A[\frac{n}{4}, \dots, (\frac{3n}{4} - 1)]$ is located in $A[\frac{n}{4}, \dots, (\frac{n}{2} - 1)]$. These $\frac{n}{4}$ values are also the $\frac{n}{4}$ smallest values in $A[\frac{n}{4}, \dots, n-1]$ because the $\frac{n}{4}$ smallest values of $A[\frac{n}{2}, \dots (n-1)]$ are located at $A[\frac{n}{2}, \dots, (\frac{3n}{4} - 1)]$ before sort.

- For $i=0, j=0$: The call to BogiSort$(A[0, \dots, \frac{n}{2} - 1])$ sorted the subarray $A[0, \dots, \frac{n}{2} - 1]$, by IH.

  $\Rightarrow$ The $\frac{n}{4}$ smallest values of $A[0, \dots, \frac{n}{2} - 1]$ is located ~~in~~ in $A[0, \dots, \frac{n}{4} - 1]$. These $\frac{n}{4}$ values are also the $\frac{n}{4}$ smallest values of $A[0, \dots, n-1]$ because the $\frac{n}{4}$ smallest values of $A[\frac{n}{4}, \dots, n-1]$ are located at $A[\frac{n}{4}, \dots, \frac{n}{2} - 1]$ before the call to sort

- Therefore, after three calls to Bogi Sort for $(i=0, j=2)$; $(i=0; j=1)$; $(i=0; j=0)$, the $\frac{n}{4}$ smallest values of the array $A[0, \dots (n-1)]$ is located at $A[0, \dots (\frac{n}{4}-1)]$

For $i=1, j=2$: The call to BogiSort$(A[\frac{n}{2}, \dots, (n-1)])$ sorted the sub-array $A[\frac{n}{2}, \dots (n-1)]$ by IH

$\Rightarrow$ The $\frac{n}{4}$ smallest values of $A[\frac{n}{2}, \dots (n-1)]$ is located at $A[\frac{n}{2}, \dots (\frac{3n}{4}-1)]$

For $i=1, j=1$: the call to BogiSort$(A[\frac{n}{4}, \dots, \frac{3n}{4}-1]$ sorted the sub-array $A[\frac{n}{4}, \dots, \frac{3n}{4}-1]$ by IH

$\Rightarrow$ the $\frac{n}{4}$ smallest values of $A[\frac{n}{4}, \dots, \frac{3n}{4}-1]$ is located in $A[\frac{n}{4}, \dots, \frac{n}{2}-1]$. These $\frac{n}{4}$ values are also the $\frac{n}{4}$ smallest values of $A[\frac{n}{4}, \dots, (n-1)]$ because the $\frac{n}{4}$ smallest values of $A[\frac{n}{2}, \dots, n-1]$ are located at $A[\frac{n}{2}, \dots, \frac{3n}{4}-1]$ before the call to bogi Sort

Therefore, after two calls to Bogi Sort for $(i=1, j=2)$; $(i=1, j=1)$ the $\frac{n}{4}$ smallest values of $A[\frac{n}{4}, \dots (n-1)]$ are located at $A[\frac{n}{4}, \dots, \frac{n}{2}-1]$

- We already have $\frac{n}{4}$ smallest values of $A[0, \dots (n-1)]$ located at $A[0, \dots, \frac{n}{4}-1]$

Therefore, we have $\frac{n}{2}$ smallest values of $A[0, \dots (n-1)]$ in sorted order located at $A[0, \dots, \frac{n}{2}-1]$

$\Rightarrow$ The $\frac{n}{2}$ greatest values of $A[0, \dots, (n-1)]$ are located at $A[\frac{n}{2}, \dots, (n-1)]$.

- For $i=2, j=2$: the call to Bogi Sort $(A[\frac{n}{2}, \dots, (n-1)]$ sorted the sub-array $A[\frac{n}{2}, \dots (n-1)]$, by IH

Therefore, the whole array is sorted

Therefore, BogiSort() algorithm is correct for all array of size $n = 2^i$ $(i > 0)$

1 1A 25 / 25

✓ - 0 pts Correct

- 25 pts Logic impossible to follow, or completely erroneous answer

- 15 pts Failing to specifically argue that each successive quarter of the array ends up in the right sub-array (without mentioning it)

- 10 pts Failing to argue in detail that each successive quarter of the array ends up in the right sub-array (but do mention it)

- 10 pts Failing to set up a correct induction proof

- 5 pts Minor flaws or typos in the proof

- 18.62 pts IDK

1B.)

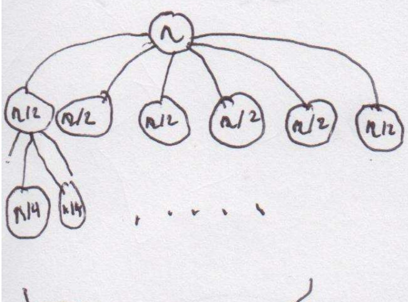
$$T(n) = \begin{cases} 6T(n/2) + d & \text{if } n > 16 \\ c & \text{if } n \leq 16 \end{cases}$$

n = size of the array

d = is some constant

1C.)

Solviagiag the reccurrence relation gives us the following



$$\left.\begin{matrix} 6 \\ \downarrow \\ 36 \\ \downarrow \\ 216 \end{matrix}\right\} 6^i$$

$\Rightarrow$ Number of recursive calls on $T(n)$

$$6^1 + 6^2 + \ldots 6^{\log n} \Rightarrow \sum_{i=1}^{\log n} 6^i \Rightarrow \frac{1-6^{\log_2 n+1}}{1-6}$$

hieght of tree is $\log_2(n)$

$T(n)$ is bounded by $O\left(\frac{1-6^{\log_2 n+1}}{1-6}\right)$

$$O\left(6^{\log_2 n}\right) \Rightarrow \boxed{T(n) = 6^{\log_2 n} \cdot c}$$

Proof ON NEXT page $\Rightarrow$

2 1B 25 / 25

✓ - 0 pts Correct

- 25 pts Logic impossible to follow, or completely erroneous answer

- 5 pts Slight mistake or typo

- 18.75 pts IDK

- 0 pts Missing constant work factor when n > 16

1B.)

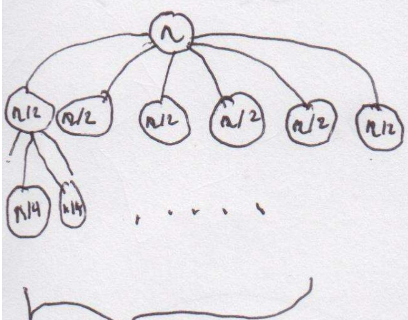
$$T(n) = \begin{cases} 6T(n/2) + d & \text{if } n > 16 \\ c & \text{if } n \le 16 \end{cases}$$

n = Size of the array

d = is some constant

1C.)

Solviagiag the reccurrence relation gives us the following



$$\left.\begin{matrix} 6 \\ \downarrow \\ 36 \\ \downarrow \\ 216 \end{matrix}\right\} 6^i$$

hieght of tree is $\log_2(n)$

$\Rightarrow$ Number of recursive calls on $T(n)$

$$6^1 + 6^2 + \dots 6^{\log n} \Rightarrow \sum_{i=1}^{\log n} 6^i \Rightarrow \frac{1 - 6^{\log_2 n + 1}}{1 - 6}$$

$T(n)$ is bounded by $O\left(\frac{1 - 6^{\log_2 n + 1}}{1 - 6}\right)$

$$O\left(6^{\log_2 n}\right) \Rightarrow \boxed{T(n) = 6^{\log_2 n} \cdot c}$$

Proof ON NEXT Page $\Rightarrow$

Base case: $n = 16 \Rightarrow T(n) = c$

Inductive hypothesis: assume for $n \leq 2^i$ that $T(n) = 6^{\log_2(n)} \cdot c$

(where $2^i > 16$)

let $n = 2^{i+1}$ where $2^{i+1} > 16$

then $T(n) = 6 \cdot T(n/2)$ // definition of reccurrence

$=$

$= 6 \cdot 6^{\log_2(n/2)} \cdot c$ // inductive hypothysis

$= 6 \cdot 6^{\log_2(n)-1} \cdot c$ // log identities

$= 6^{\log_2(n)} \cdot c$ // simplification

$T(n) = 6^{\log_2(n)} \cdot c$ where # of comparisons is $O(6^{\log_2(n)})$

3 1C 25 / 25

✓ - 0 pts Correct

- 25 pts Logic impossible to follow, or completely erroneous answer

- 0 pts Failing to account for the (constant) work at every node / level of recursion.

- 15 pts Stating an incorrect (non-constant) amount of work at each node / level of recursion.

- 5 pts Concluding a runtime other than 6^(log2 n), but otherwise essentially correct

- 18.75 pts IDK

1D.) The number of swaps ~~to~~ is at most $\binom{n}{2}$ because if we consider the worst case where we have an array of n elements which is ordered by decreasing order then for us to have this array in order (i.e array is in increasing order) we will have to preform $\frac{n(n-1)}{2}$ total swaps which is the same as $\binom{n}{2} \Rightarrow \frac{n(n-1)\cancel{(n-2)!}}{2\,\cancel{(n-2)!}}$. Also once we sort the array insertion sort will not do anything if we were to try and call insertion sort on it again. So keeping these two facts in mind bogi sort algorithm cannot exceed more than $\binom{n}{2}$ swaps because even if recursively call insertion sort on sub-arrays of the orginal we cannot have more than $\binom{n}{2}$ swaps since the total number of swaps that may occur on an unsorted array (decreasing order) is $\frac{n(n-1)}{2}$ and once the array is sorted insertion sort will not do anything nor will do anymore swaps since the array is already sorted.

4 1D 25 / 25

✓ - 0 pts Correct

- 25 pts Logic impossible to follow, or completely erroneous answer

- 10 pts Failing to mention that the no-repeated-swapping property is a result of insertion sort

- 15 pts Failure to argue that no pair of entries can be swapped more than once

- 5 pts Almost correct, but minor flaw or typo

- 18.75 pts IDK