Algorithm Analysis
Mid Term 1

Ephraim Wang
wang0037

1 a) Prove that initial heaping takes $O(n)$ operations.

After implementing a Binary Tree from an unsorted array, heapify is applied to the last non-leaf node, located at index $\left[\frac{n}{2}-1\right]$. This way, we implement heapify from the last non-leaf node up to the root.

Number of Nodes at depth d from root : 2^d

Number of Nodes at height h from bottom : $\frac{n}{2^{h+1}}$

At height h, each node takes at most $O(h)$ time complexity.

$$\therefore T(n) = \sum_{h=0}^{\log n}\left(\frac{n}{2^{h+1}} \cdot O(h)\right) = O(n) \cdot \sum_{h=0}^{\log n} \frac{n}{2^h}$$

$$= O(n) \cdot O(2)$$

$$= O(n) \cdot O(1)$$

$$= O(n)$$

b) Average case complexity focuses on the typical* time complexity of algorithms by dividing the sum of time complexities for all possible inputs by the number of possible inputs.

Quick Sort :

Let $T(n)$ be average case time complexity on array size n.

$$T(n) = T(n_L) + T(n_R) + O(n) \quad \text{for } n > 1.$$

Where possible sizes of n_L & n_R are equally possible from $[0, n-1]$, each w $\frac{1}{n}$ possibility.

same

Hence, $T(n_L) = T(n_R) = T(n-1-n_L)$

$$T(n) = n + \frac{1}{n}\sum_{k=0}^{n-1}\left[T(k) + T(n-1-k)\right]$$

$$nT(n) = n^2 + 2\sum_{k=0}^{n-1} T(k)$$

$$(n-1)T(n-1) = 2\left[\sum_{k=0}^{n-2} T(k)\right] + (n-1)^2$$

$$n \cdot T(n) - (n-1)T(n-1) = n^2 - (n^2 - 2n + 1) + 2\left[\sum_{k=0}^{n-1} T(k)\right] - 2\left[\sum_{k=0}^{n-2} T(k)\right]$$

$$= 2n - 1 + 2T(n-1)$$

$$nT(n) = 2n - 1 + 2T(n-1) + (n-1)T(n-1)$$

$$= 2n - 1 + (n+1)T(n-1)$$

↓ Divide by $(n+1)$

$$\frac{T(n)}{(n+1)} = \frac{2n-1}{n(n+1)} + \frac{T(n-1)}{n}$$

Let $\frac{T(n)}{n+1} = S(n)$ where $S(n) = S(n-1) + \frac{2n-1}{n(n+1)}$

$$S(n) = \sum_{i=2}^{n} \frac{1}{i} \approx \log n$$

$$\therefore T(n) = (n+1)(\log n) = n\log n + \log n$$

$$= O(n \log n)$$

2 a) $T(n) = 3T\left(\frac{n}{2}\right) + n$, $T(1) = 1$, $n = 2^k$

$$T(n) = 3\left[3T\left(\frac{n}{4}\right) + \frac{n}{2}\right] + n$$

$$= 3^2\left[T\left(\frac{n}{2^2}\right)\right] + 3^1\left(\frac{n}{2^1}\right) + 3^0\left(\frac{n}{2^0}\right)$$

$$= 3^2\left[3T\left(\frac{n}{2^3}\right) + \frac{n}{2^2}\right] + 3^1\left(\frac{n}{2^1}\right) + 3^0\left(\frac{n}{2^0}\right)$$

$$= 3^3\left[T\left(\frac{n}{2^3}\right)\right] + 3^2\left(\frac{n}{2^2}\right) + 3^1\left(\frac{n}{2^1}\right) + 3^0\left(\frac{n}{2^0}\right)$$

$$\therefore T(n) = 3^m\left(\frac{n}{2^m}\right) + \sum_{i=0}^{m-1} 3^i\left(\frac{n}{2^i}\right)$$

$$= 3^m\left(\frac{n}{2^m}\right) + n\sum_{i=0}^{m-1}\left(\frac{3}{2}\right)^i$$

Since $n = 2^k$, we expand till $\frac{n}{2^m} = 1 \iff n = 2^m$

Geometric Series : $r = 3/2$

$m = \log_2 n$

$$T(n) = 3^{\log_2 n}\cdot T\left(\frac{n}{2^{\log_2 n}}\right) + n\sum_{i=0}^{\log_2 n - 1}\left(\frac{3}{2}\right)^i$$

$$= 3^{\log_2 n}\cdot T\left(\frac{n}{n}\right) + n\cdot\frac{1-(3/2)^{\log_2 n}}{1 - 3/2}$$

$$= 3^{\log_2 n}\cdot 1 + n\cdot 2\left(\left(\frac{3}{2}\right)^{\log_2 n} - 1\right)$$

$$= 3^{\log_2 n} + 2n\left(n^{\log_2 3/2} - 1\right)$$

$$= 3^{\log_2 n} + 2n^{(\log_2 3/2)+1} - 2n$$

$$= n^{\log_2 3} + 2n^{\log_2 3} - 2n$$

$$T(n) = O(n^{\log_2 3})$$

2b) $T(n) = 2T\left(\frac{n}{2}\right) + n^2$, $T(1) = 1$, $n = 2^k$

$$T(n) = 2\left[2T\left(\frac{n}{2^2}\right) + \left(\frac{n}{2}\right)^2\right] + n^2$$
$$= 2^2\left[T\left(\frac{n}{2^2}\right)\right] + 2\left(\frac{n}{2}\right)^2 + n^2$$
$$= 2^2\left[2T\left(\frac{n}{2^3}\right) + \left(\frac{n}{2^2}\right)^2\right] + 2\left(\frac{n}{2}\right)^2 + n^2$$
$$= 2^3\left[T\left(\frac{n}{2^3}\right)\right] + 2^2\left(\frac{n}{2^2}\right)^2 + 2^1\left(\frac{n}{2^1}\right)^2 + 2^0\left(\frac{n}{2^0}\right)^2$$
$$\therefore T(n) = 2^m\left[T\left(\frac{n}{2^m}\right)\right] + \sum_{i=0}^{m-1} 2^i\left(\frac{n}{2^i}\right)^2$$
$$= 2^m\left[T\left(\frac{n}{2^m}\right)\right] + n^2\sum_{i=0}^{m-1}\frac{1}{2^i}$$

Since $n = 2^k$, we expand till $\frac{n}{2^m} = 1$

Geometric Series: $r = \frac{1}{2}$

$n = 2^m \iff m = \log_2 n$

$$T(n) = 2^{\log_2 n}\left[T\left(\frac{n}{2^{\log_2 n}}\right)\right] + n^2\sum_{i=0}^{\log_2 n - 1}\frac{1}{2^i}$$
$$= n\left[T\left(\frac{n}{n}\right)\right] + n^2 \cdot \left[\frac{1 - \left(\frac{1}{2}\right)^{\log_2 n}}{1 - \frac{1}{2}}\right]$$
$$= n + n^2\left[\frac{1 - \frac{1}{n}}{\frac{1}{2}}\right]$$
$$= n + 2n^2 - 2n = 2n^2 - n$$

$T(n) = O(n^2)$ ☆

Q3

Cost for chain l.

$l = 2$

$$A_1 \cdot A_2 = 15 \cdot 5 \cdot 10 = 750$$

$$A_2 \cdot A_3 = 5 \cdot 10 \cdot 20 = 1\,000$$

$$A_3 \cdot A_4 = 10 \cdot 20 \cdot 25 = 5\,000$$

$$A_4 \cdot A_5 = 20 \cdot 25 \cdot 10 = 5\,000$$

$l = 3$

$$A_1A_2A_3 = \min\left[\overset{A_2A_3}{1000} + \overset{A_1(A_2A_3)}{15\cdot5\cdot20},\ \overset{A_1A_2}{750} + \overset{(A_1A_2)A_3}{15\cdot10\cdot20}\right] = 2500$$

$$A_2A_3A_4 = \min\left[\overset{A_3A_4}{5000} + \overset{A_2(A_3A_4)}{5\cdot10\cdot25},\ \overset{A_2A_3}{1000} + \overset{(A_2A_3)A_4}{5\cdot20\cdot25}\right] = 3500$$

$$A_3A_4A_5 = \min\left[\overset{A_4A_5}{5000} + \overset{A_3(A_4A_5)}{10\cdot20\cdot10},\ \overset{A_3A_4}{5000} + \overset{(A_3A_4)A_5}{10\cdot25\cdot10}\right] = 7\,000$$

$l = 4$

$$A_1A_2A_3A_4 = \min\left[\overset{A_2A_3A_4}{3500} + \overset{A_1(A_2A_3A_4)}{15\cdot5\cdot25},\ \overset{A_1A_2}{750} + \overset{A_3A_4}{5000} + \overset{(A_1A_2)(A_3A_4)}{15\cdot10\cdot25},\ \overset{A_1A_2A_3}{2500} + \overset{(A_1A_2A_3)A_4}{15\cdot20\cdot25}\right] = 5375$$

$$A_2A_3A_4A_5 = \min\left[\overset{A_3A_4A_5}{7000} + \overset{A_2(A_3A_4A_5)}{5\cdot10\cdot10},\ \overset{A_2A_3}{1000} + \overset{A_4A_5}{5000} + \overset{(A_2A_3)(A_4A_5)}{5\cdot20\cdot10},\ \overset{A_2A_3A_4}{3500} + \overset{(A_2A_3A_4)A_5}{5\cdot25\cdot10}\right] = 4750$$

$l = 5$

$$\min\left[\overset{A_2A_3A_4A_5}{4750} + \overset{A_1(A_2A_3A_4A_5)}{15\cdot5\cdot10},\ \overset{A_1A_2}{750} + \overset{A_3A_4A_5}{7000} + \overset{(A_1A_2)(A_3A_4A_5)}{15\cdot10\cdot10},\ \overset{A_1A_2A_3}{2500} + \overset{A_4A_5}{5000} + \overset{(A_1A_2A_3)(A_4A_5)}{15\cdot20\cdot10},\ \overset{A_1A_2A_3A_4}{5375} + \overset{(A_1A_2A_3A_4)A_5}{15\cdot25\cdot10}\right]$$

$$= 5500_{\#}$$

Optimal Multiplication: $A_1\left[\left((A_2A_3)A_4\right)A_5\right]$

4

An inversion in permutation is a pair of indices (a, b) such that : $a < b$ and $Value[a] > Value[b]$. In other words, a situation where larger elements appear earlier than smaller elements in an array.

Since bubble sort works by swapping adjacent elements w the larger element being swapped towards the end of the list. This algorithm uses a twice nested for-loop to ensure that each element is "bubbled" or swapped to its sorted position in the array. When the outer for-loop is finished, all inversions in an array should be eliminated, resulting in a sorted array.

Therefore, the more inversion in an array, the greater the number of swaps required, resulting in greater time complexity.

Best case: $O(n)$

Worst case: $O(n^2)$

$$\begin{array}{c} idx \\ Val \end{array} \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 5 & 3 & 4 & 1 & 2 & 7 & 6 \end{bmatrix}$$

	Swaps:
1st pass [3 4 1 2 5 6 7]	5
2nd pass [3 1 2 4 5 6 7]	2
3rd pass [1 2 3 4 5 6 7]	2

$\left.\begin{array}{c} 5 \\ 2 \\ 2 \end{array}\right\}$ 9 inversions #