Colby Frison
OUID: 113568816

CS-4413 || Midterm 1

02/28/2025

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

- the height of the heap is $(\log_2 n)$
- at most $(n/2)$ non-leaf nodes in the heap
- each level 'h' of the heap has $(n/2^{h+1})$ nodes
  - ↳ each node at level h has height

$$\sum_{h=0}^{\log_2 n} (\text{num nodes at lvl } h) \cdot O(h)$$

↓ substitute

$$\sum_{h=0}^{\log_2 n} \left(\frac{n}{2^{h+1}}\right) \cdot O(h) \to O\left(n \sum_{h=0}^{\log_2 n} \frac{h}{2^h}\right) \to \sum_{h=0}^{\infty} \frac{h}{2^h} = 1 \quad O(n)$$

b) Define Average Case Complexity. Derive an expression for the average case complexity of Quick sort?

Average Case Complexity is the expected running time of an algorithm averaged over all possible inputs of a given size

Derivation:
- $T(n)$ = average-case running time of quick sort
  - ↳ Depends on time to partition which = $O(n)$
  - ↳ Depends on time to recursively sort the two arrays

when pivot is chosen randomly creating sub arrays $(i-1)$ & $(n-i)$ where i is the pivot index

Items to the left of pivot; Items to right of pivot

Quicksort recursively sorts the two subarrays with times $T(i-1)$ & $T(n-i)$

Since pivot is chosen randomly, each pivot i has a $1/n$ probability of being selected

Combining every thing:

$$T(n) = \underbrace{\frac{1}{n}}_{\text{pick pivot}} \sum_{i=1}^{n} (\underbrace{T(i-1)}_{\text{left}} + \underbrace{T(n-i)}_{\text{right}}) + \underbrace{O(n)}_{\text{pivot selection time}}$$

The average-case running time $T(n)$ is the weighted average of the times for all possible choices of the pivot

$T(i-1)$ & $T(n-i)$ are symmetric because

↳ when $i=1$: $T(i-1) = T(0)$ & $T(n-i) = T(n-1)$; when $i=n$: $T(n-i) = T(0)$ & $T(i-1) = T(n-1)$ → we can now $\sum_{i=1}^{n} T(n-1) = \sum_{i=1}^{n} T(n-i)$

we can rewrite as:

$$T(n) = \frac{2}{n}\sum_{i=1}^{n} T(i-1) + O(n) \xrightarrow[\text{summation}]{\text{solve with}} T(n) = O(n \log n)$$

2. Solve the following recurrence using substitution method:

a) $T(n) = 3T(n/2) + n,\ T(1) = 1,\ n = 2^k$

sub $T(n/2)$: $T(n) = 3(3T(n/4) + n/2) + n = 3^2 T(n/4) + 3 \cdot n/2 + n$

sub $T(n/4)$: $T(n) = 3^2(3T(n/8) + n/4) + 3 \cdot n/2 + n = 3^3 T(n/8) + 3^2 \cdot n/4 + 3 \cdot n/2 + n$

Generalize substitution pattern: $T(n) = 3^k T(n/2^k) + n\left(1 + 3/2 + 3^2/2^2 + \dots + \frac{3^{k-1}}{2^{k-1}}\right)$

Base case: $T(1) = 1$

$\frac{n}{2^k} = 1 \Rightarrow n = 2^k \Rightarrow k = \log_2 n$

$T(n) = 3^{\log_2 n}\, \cancel{T(1)} + n\left(1 + 3/2 + 3^2/2^2 + \dots + \frac{3^{\log_2 n - 1}}{2^{\log_2 n - 1}}\right)$ geometric series with ratio $= 3/2$

simplify: $a^{\log_b c} = c^{\log_b a}$; $3^{\log_2 n} = n^{\log_2 3}$

$S = \text{"} \quad \dots \quad \text{"} \Rightarrow S = \frac{r^k - 1}{r - 1}$

sub r $\Rightarrow S = \frac{(3/2)^{\log_2 n} - 1}{3/2 - 1} = 2\left((3/2)^{\log_2 n} - 1\right) = 2\left(n^{\log_2 \frac{3}{2}} - 1\right)$

Combine terms:

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

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

↳ $T(n) = 3n^{\log_2 3} - 2n$ → Dominant term ∴ $O(n^{\log_2 3})$

b) $T(n) = 2T(n/2) + n^2,\ T(1) = 1,\ n = 2^k$

sub $T(n/2)$: $T(n) = 2(2T(n/4) + (n/2)^2) + n^2 = 2^2 T(n/4) + 2\, n^2/4 + n^2$

sub $T(n/4)$: $T(n) = 2^2(2T(n/8) + (n/4)^2) + 2\,n^2/4 + n^2 = 2^3 T(n/8) + 2^2\, n^2/4^2 + 2\, n^2/4 + n^2$

→ $T(n) = 2^k T(n/2^k) + n^2\left(1 + 1/2 + 1/2^2 + \dots + 1/2^{k-1}\right)$

sub ↳ $T(n) = 2^{\log_2 n}\, T(1) + n^2\left(1 + 1/2 + 1/2^2 + \dots + \frac{1}{2^{\log_2 n - 1}}\right)$ → $2^{\log_2 n}$

↳ $T(n) = n + n^2\left(1 + 1/2 + 1/2^2 + \dots + 1/2^{\log_2 n - 1}\right)$ → $S = \frac{1 - r^k}{1 - r}$

$S = \frac{1 - (1/2)^{\log_2 n}}{1 - 1/2} = 2(1 - 1/n)$

Combine:

$T(n) = n + n^2 \cdot 2(1 - 1/n) \Rightarrow T(n) = n + 2n^2 - 2n = 2n^2 - n \Rightarrow T(n) = O(n^2)$

3. Compute the optimal cost and optimal parenthesis for multiplying the following matricies:

$A_1|_{15 \times 5}$, $A_2|_{5 \times 10}$, $A_3|_{10 \times 20}$, $A_4|_{20 \times 25}$, $A_5|_{25 \times 10}$

$[15, 5, 10, 20, 25, 10]$ (A_1, A_2, A_3, A_4, A_5)

2 groupings

$A_1A_2 = 15 \cdot 5 \cdot 10 = 750$
$A_2A_3 = 5 \cdot 10 \cdot 20 = 1000$
$A_3A_4 = 10 \cdot 20 \cdot 25 = 5000$
$A_4A_5 = 20 \cdot 25 \cdot 10 = 5000$

3 groupings:

$A_1A_2A_3 =$
- $A_1(A_2A_3) = A_1 + A_2A_3 + (15 \cdot 5 \cdot 20) = 1000 + 1500 = 2500$
- $(A_1A_2)A_3 = A_1A_2 + A_3 + (15 \cdot 10 \cdot 20) = 750 + 3000 = 3750$

lower term ∴ $A_1A_2A_3 = 2500$

$A_2A_3A_4 =$
- $A_2(A_3A_4) = A_2 + A_3A_4 + (5 \cdot 10 \cdot 25) = 5000 + 1250 = 6250$
- $(A_2A_3)A_4 = A_2A_3 + A_4 + (5 \cdot 20 \cdot 25) = 1000 + 2500 = 3500$

∴ $A_2A_3A_4 = 3500$

$A_3A_4A_5 =$
- $A_3(A_4A_5) = A_3 + A_4A_5 + (10 \cdot 20 \cdot 10) = 5000 + 2000 = 7000$
- $(A_3A_4)A_5 = A_3A_4 + A_5 + (10 \cdot 25 \cdot 10) = 5000 + 2500 = 7500$

∴ $A_3A_4A_5 = 7000$

4 grouping

$A_1A_2A_3A_4 =$
- $A_1(A_2A_3A_4) = A_1 + A_2A_3A_4 + (15 \cdot 5 \cdot 25) = 3500 + 1875 = 5375$
- $(A_1A_2)(A_3A_4) = A_1A_2 + A_3A_4 + (15 \cdot 10 \cdot 25) = 750 + 5000 + 3750 = 9500$
- $(A_1A_2A_3)A_4 = A_1A_2A_3 + A_4 + (15 \cdot 20 \cdot 25) = 2500 + 7500 = 10\,000$

∴ $A_1A_2A_3A_4 = 5375$

$A_2A_3A_4A_5 =$
- $A_2(A_3A_4A_5) = A_2 + A_3A_4A_5 + (5 \cdot 10 \cdot 10) = 7000 + 500 = 7500$
- $(A_2A_3)(A_4A_5) = A_2A_3 + A_4A_5 + (5 \cdot 20 \cdot 10) = 1000 + 5000 + 1000 = 7000$
- $(A_2A_3A_4)A_5 = A_2A_3A_4 + A_5 + (5 \cdot 25 \cdot 10) = 3500 + 1250 = 4750$

∴ $A_1A_2A_3A_4 = 4750$

5 group

$A_1A_2A_3A_4A_5 =$
- $A_1(A_2A_3A_4A_5) = A_1 + A_2A_3A_4A_5 + (15 \cdot 5 \cdot 10) = 4750 + 750 = 5500$
- $(A_1A_2)(A_3A_4A_5) = A_1A_2 + A_3A_4A_5 + (15 \cdot 10 \cdot 10) = 750 + 7000 + 1500 = 9250$
- $(A_1A_2A_3)(A_4A_5) = A_1A_2A_3 + A_4A_5 + (15 \cdot 20 \cdot 10) = 2500 + 5000 + 3000 = 10500$
- $(A_1A_2A_3A_4)A_5 = A_1A_2A_3A_4 + A_5 + (15 \cdot 25 \cdot 10) = 5375 + 3750 = 9125$

∴ $A_1A_2A_3A_4A_5 = 5500$

optimal cases:

$(A_1(A_2A_3A_4A_5))$
$((A_2A_3A_4)A_5)$
$((A_2A_3)A_4)$ ⇒ combine

Optimal cost

$\boxed{5500}$

Optimal Parenthesization

$(A_1(((A_2A_3)A_4)A_5))$

4. a) What is an inversion in a permutation?

An **Inversion in a permutation** is a pair of indicies (pair(i, j)) where $i < j$ and $A[i] > A[j]$

b) What is the relationship between the running time of bubble sort & number of inversions in the input array?

Bubble sort swaps adjacent elements in an array until sorted ; $O(n^2)$ time complexity.

each swap in the array is essentially fixing one **inversion** of the set

So, the time complexity of any given set has a direct relationship with the number of inversions needed:

we will call the total number of swaps/inversions needed 'd'

Each pass through the array takes $O(n)$ time

In the worst case for bubble sort $d = \frac{n(n-1)}{2}$

so the time complexity is $O(n + d)$

which will become $O(n^2)$ for larger sets

$O\left(n + \frac{n(n-1)}{2}\right)$
$O\left(n + \frac{n^2 - n}{2}\right)$
$O\left(\frac{2n}{2} + \frac{n^2 - n}{2}\right)$
$O\left(\frac{n^2 + n}{2}\right)$

→ Drop constants since they're inconsequential in larger sets

$O(n^2 + n)$

→ n^2 is dominant in comparison to n so n is dropped

$O(n^2)$

c) Find the number of inversions in: $\begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 5 & 3 & 4 & 1 & 2 & 7 & 6 \end{bmatrix}$

natural order of elements: [1 2 3 4 5 6 7]
unsorted elements: [5 3 4 1 2 7 6]

inversions

first pass
- 1: 5 > 3 4 1 2 7 6
- 2: 3 5 > 4 1 2 7 6
- 3: 3 4 5 > 1 2 7 6
- 4: 3 4 1 5 > 2 7 6
- ×: 3 4 1 2 5 < 7 6
- 5: 3 4 1 2 5 7 > 6
- ✓: 3 4 1 2 5 6 7

second pass
- ×: 3 < 4 1 2 5 6 7
- 6: 3 4 1 2 5 6 7
- 7: 3 1 4 2 5 6 7
- ✓: 3 1 2 4 5 6 7

third pass
- 8: 3 1 2 4 5 6 7
- 9: 1 3 2 4 5 6 7
- ✓: 1 2 3 4 5 6 7

↓ 1 more pass to check last elements
- ×: 1 < 2 3 4 5 6 7
- ✓: 1 2 3 4 5 6 7

$\boxed{9 \text{ inversions}}$