Quiz 2B

CS/CE 412/471 Algorithms: Design and Analysis, Spring 2025

17 Feb, 2025. 3 questions, 40 points, 6 printed sides

Instructions:

1. Enter your name and ID below and at the top of every side of your solution.
2. You have 60 minutes to complete this quiz.
3. You may keep a calculator with you. Submit all other devices with your bag at the front of the classroom.
4. Solve the problems by hand in clear and legible handwriting on your own paper.
5. Provide precise and concise solutions.
6. Consulting or copying from another student constitutes a violation of academic integrity. Any infractions will result in disciplinary action, including a failing grade for the quiz.
7. For your solution to be graded, submit this problem sheet along with your solution.

Student Name: ______________________________

Student ID: ______________________________

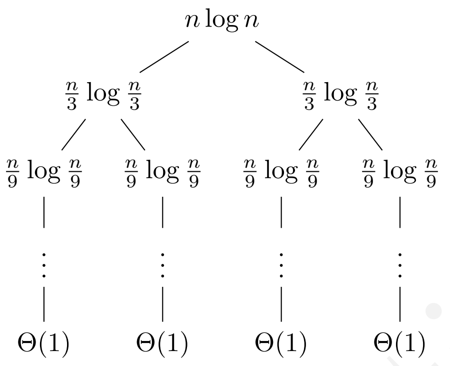
viel Spaß!

1. Recursion Trees and Master Theorem [15 points]

The questions below refer to the recurrence: $T(n) = 2T\left(\frac{n}{3}\right) + n \log n$.

(a) $\boxed{\text{5 points}}$ Construct a recursion tree and indicate the total cost at each level.

Solution:


$n \log n$
$\frac{n}{3} \log \frac{n}{3}$
$\frac{n}{3} \log \frac{n}{3}$
$\frac{n}{9} \log \frac{n}{9}$
$\frac{n}{9} \log \frac{n}{9}$
$\frac{n}{9} \log \frac{n}{9}$
$\frac{n}{9} \log \frac{n}{9}$
$\Theta(1)$
$\Theta(1)$
$\Theta(1)$
$\Theta(1)$

Level	Total Cost
0	$n \log n$
1	$\frac{2}{3} n \log \frac{n}{3}$
2	$\frac{2^2}{3^2} n \log \frac{n}{3^2}$
...	...
i	$\frac{2^i}{3^i} n \log \frac{n}{3^i}$
...	...
$\log_3 n$	$2^{\log_3 n}\Theta(1) = n^{\log_3 2}\Theta(1) = \Theta(n^{\log_3 2})$

(b) $\boxed{\text{5 points}}$ Use the recursion tree to determine the asymptotic complexity of $T(n)$.

Solution:

Proof. $T(n) = \Omega(n \log n)$.
Summing all levels:

$$T(n) = n \sum_{i=0}^{\log_3 n - 1} \left(\frac{2}{3}\right)^i \log \frac{n}{3^i} + \Theta(n^{\log_3 2})$$

Applying $\log \frac{a}{b} = \log a - \log b$, $\log a^b = b \log a$, and taking out unaffected terms from the summation:

$$T(n) = n \log n \sum_{i=0}^{\log_3 n - 1} \left(\frac{2}{3}\right)^i - n \log 3 \sum_{i=0}^{\log_3 n - 1} \left(\frac{2}{3}\right)^i i + \Theta(n^{\log_3 2})$$

Using the geometric series sum:

$$\begin{aligned} T(n) &= n\log n\left(\frac{3}{n}(n-n^{\log_3 2})\right) - n\log 3\sum_{i=0}^{\log_3 n-1}\left(\frac{2}{3}\right)^i i + \Theta(n^{\log_3 2}) \\ &= 3\log n(n-n^{\log_3 2}) - n\log 3\sum_{i=0}^{\log_3 n-1}\left(\frac{2}{3}\right)^i i + \Theta(n^{\log_3 2}) \end{aligned}$$

Rearranging and highlighting a tricky sum:

$$T(n) = 3\log n(n-n^{\log_3 2}) + \Theta(n^{\log_3 2}) - n\log 3 \sum_{i=0}^{\log_3 n-1}\left(\tfrac{2}{3}\right)^i i$$

Working on the tricky sum and using $i \geq 0$ and the sum of an arithmetic series:

$$\begin{aligned} \left(\frac{2}{3}\right)^i &\leq \left(\frac{3}{3}\right)^i \\ \implies \left(\frac{2}{3}\right)^i i &\leq \left(\frac{3}{3}\right)^i i = i \\ \implies \sum_{i=0}^{\log_3 n-1}\left(\frac{2}{3}\right)^i i &\leq \sum_{i=0}^{\log_3 n-1} i = \frac{\log_3 n}{2}(\log_3 n - 1) \end{aligned}$$

Substituting this inequality and accounting for the negative sign before the summation in the expression for $T(n)$:

$$\begin{aligned} &T(n) \geq 3\log n(n-n^{\log_3 2}) + \Theta(n^{\log_3 2}) - n\log 3\frac{\log_3 n}{2}(\log_3 n - 1) \\ &T(n) \geq \Theta(n\log n) \end{aligned}$$

Thus, $T(n) = \Omega(n\log n)$. □

(c) $\boxed{\text{5 points}}$ Verify your answer above using the Master Theorem (below). Show which case applies and how.

The solution to an algorithmic recurrence of the form, $T(n) = aT\left(\frac{n}{b}\right) + f(n)$, is

$$T(n) = \begin{cases} \Theta(n^{\log_b a}) & \exists\epsilon > 0: \ f(n) = O(n^{\log_b a-\epsilon}) \\ \Theta(n^{\log_b a}\lg^{k+1} n) & \exists k \geq 0: \ f(n) = \Theta(n^{\log_b a}\lg^k n) \\ \Theta(f(n)) & \exists\epsilon > 0: \ f(n) = \Omega(n^{\log_b a+\epsilon}), \exists c < 1: af(\frac{n}{b}) \leq cf(n) \end{cases}$$

Solution:

Proof. Case 3 applies and $T(n) = \Theta(n\log n)$.
For the given recurrence, $a = 2, b = 3, f(n) = n\log n$, and $n^{\log_b a} = n^{\log_3 2} \approxeq n^{0.6}$.

Then $f(n) = n \log n = \Omega(n^{\log_3 2 + 1 - \log_3 2}) = \Omega(n)$.
The third case seems to apply with $\epsilon = 1 - \log_3 2 \approxeq 0.4$.
For the third case to apply, there must also be a c such that

$$\begin{aligned} af(\frac{n}{b}) &\leq cf(n) \\ \implies \frac{2n}{3} \log \frac{n}{3} &\leq cn \log n \\ \implies \frac{2}{3} n \log n - \frac{2}{3} n \log 3 &\leq cn \log n \end{aligned}$$

As $n > 0$, one possible value of c is $\frac{2}{3}$.
Then, as per Case 3, $T(n) = \Theta(f(n)) = \Theta(n \log n)$ which agrees with the finding from the previous part, i.e. $T(n) = \Omega(n \log n)$. $\square$

2. Divide and Conquer Vector Operations [10 points]

An n-dimensional vector, v, is a matrix containing n elements and of the form $\begin{bmatrix} v_1 v_2 \dots v_n \end{bmatrix}^\top$. Given n-dimensional vectors, a and b, their difference, c, is an n-dimensional vector given by $c_i = a_i - b_i, 1 \leq i \leq n$.

Provide a divide and conquer algorithm to compute the difference of two n-dimensional vectors, a and b.

Solution:

```
def vector_diff(a, b, c, low, high):
    # populate c[low:high]. low is inclusive, high is exclusive.
    if low == high - 1:
        c[low] = a[low] - b[low]
    else:
        mid = (low + high) // 2
        vector_diff(a, b, c, low, mid)
        vector_diff(a, b, c, mid, high)
```

3. Maximum Subarray Problem [15 points]

Given the following array: $A = [-2, 1, -3, 4, -1, 2]$.

(a) $\boxed{\text{5 points}}$ Find the maximum subarray sum using the brute-force approach.

Solution:

Subarray	Sum	Subarray	Sum	Subarray	Sum
$\langle 2 \rangle$	2	$\langle 4 \rangle$	4	$\langle -3 \rangle$	-3
$\langle -1 \rangle$	-1	$\langle 4, -1 \rangle$	3	$\langle -3, 4 \rangle$	1
$\langle -1, 2 \rangle$	1	$\langle 4, -1, 2 \rangle$	5	$\langle -3, 4, -1 \rangle$	0
				$\langle -3, 4, -1, 2 \rangle$	2

Subarray	Sum
$\langle 1 \rangle$	1
$\langle 1, -3 \rangle$	-2
$\langle 1, -3, 4 \rangle$	2
$\langle 1, -3, 4, -1 \rangle$	1
$\langle 1, -3, 4, -1, 2 \rangle$	3

Subarray	Sum
$\langle -2 \rangle$	-2
$\langle -2, 1 \rangle$	-1
$\langle -2, 1, -3 \rangle$	-4
$\langle -2, 1, -3, 4 \rangle$	0
$\langle -2, 1, -3, 4, -1 \rangle$	-1
$\langle -2, 1, -3, 4, -1, 2 \rangle$	1

The maximum subarray sum is 5.

(b) $\boxed{\text{5 points}}$ Apply the divide-and-conquer approach to find the maximum subarray and show the steps.

Solution:

```
mss([-2,1,-3,4,-1,2])
├─ left = mss([-2,1,-3])
│  ├─ left = mss([-2,1])
│  │  ├─ left = mss([-2]) = -2
│  │  ├─ right = mss([1]) = 1
│  │  ├─ cross = sum([-2,1]) = -1
│  │  └─ return max(-2,1,-1) = 1
│  ├─ right = mss([-3]) = -3
│  ├─ cross = sum([1,-3]) = -2
│  └─ return max(1,-3,-2) = 1
├─ right = mss([4,-1,2])
│  ├─ left = mss([4,-1])
│  │  ├─ left = mss([4]) = 4
│  │  ├─ right = mss([-1]) = -1
│  │  ├─ cross = sum([4,-1]) = 3
│  │  └─ return max(4,-1,3) = 4
│  ├─ right = mss([2]) = 2
│  ├─ cross = sum([4,-1,2]) = 5
│  └─ return max(4,2,5) = 5
├─ cross = sum([1,-3,4,-1,2]) = 3
└─ return max(1,5,3) = 5
```

(c) $\boxed{\text{5 points}}$ Argue about the time complexity of each method.

Solution: The brute force method checks all the pairs of indexes. There are $\Theta(n^2)$ pairs. For each pair, it computes the sum of all the elements between the indices, which, unless optimized, is a $\Theta(n)$ operating. This approach thus takes $\Theta(n^3)$ time. The divide-and-conquer approach solves 2 halves and then performs a linear scan. This leads to the recurrence, $T(n) = T(\frac{n}{2}) + \Theta(n)$, which solves to $\Theta(n \log n)$.