Quiz 2C

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

17 Feb, 2025. 3 questions, 40 points, 5 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: ____________________

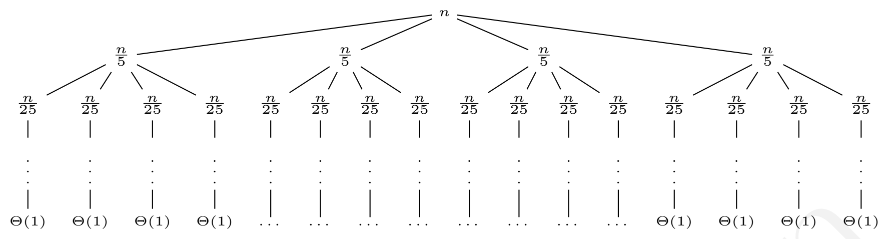
viel Spaß!

1. Recursion Trees and Master Theorem [15 points]

The questions below refer to the recurrence: $T(n) = 4T\left(\frac{n}{5}\right) + n$.

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

Solution:



Level	Total Cost
0	n
1	$\frac{4}{5}n$
2	$\left(\frac{4}{5}\right)^2 n$
$\dots$	$\dots$
i	$\left(\frac{4}{5}\right)^i n$
$\dots$	$\dots$
$\log_5 n$	$4^{\log_5 n}\Theta(1) = n^{\log_5 4}\Theta(1) = \Theta(n^{\log_5 4})$

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

Solution:

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

$$T(n) = n \sum_{i=0}^{\log_5 n - 1} \left(\frac{4}{5}\right)^i + \Theta(n^{\log_5 4})$$

Using the geometric series sum:

$$T(n) = 5(n - n^{\log_5 4}) + \Theta(n^{\log_5 4})$$

Thus, $T(n) = \Theta(n)$. □

(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 $T(n) = \Theta(n^2)$.
For the given recurrence, $a = 4, b = 5, f(n) = n$, and $n^{\log_b a} = n^{\log_5 4} \approx n^{0.9}$.
Then $f(n) = n = \Omega(n^{\log_5 4 + 1 - \log_5 4}) = \Omega(n)$.
The third case seems to apply with $\epsilon = 1 - \log_5 4 \approx 0.1$.
For the third case to apply, there must also be a c such that $af(\frac{n}{b}) \le cf(n) \implies \frac{4n}{5} \le cn$.
One possible value of c is $\frac{4}{5}$.
Then, as per Case 3, $T(n) = \Theta(f(n)) = \Theta(n)$. □

2. Divide and Conquer Vector Operations [10 points]

An n-dimensional vector, v, is a matrix containing n elements and of the form $\left[v_1 v_2 \dots v_n\right]^\top$. Given n-dimensional vectors, a and b, their *scalar product*, c, is a number given by $c = \sum_{i=1}^{n} a_i b_i$ where $a_i b_i$ is the product of a_i and b_i.

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

Solution:

```
def vector_mult(a, b, low, high):
    # return the dot product of the vectors represented by
    # a[low:high] and a[low:high].
    # low is inclusive, high is exclusive.
    if low == high - 1:
        return a[low]
    else:
        mid = (low + high) // 2
        return vector_mult(a, b, low, mid) + vector_mult(a, b, mid, high)
```

3. Maximum Subarray Problem [15 points]

Given the following array: $A = [1, -2, 3, 10, -4, 7]$.

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

Solution:

Subarray	Sum
$\langle 7 \rangle$	7
$\langle -4 \rangle$	−4
$\langle -4, 7 \rangle$	3

Subarray	Sum
$\langle 10 \rangle$	10
$\langle 10, -4 \rangle$	6
$\langle 10, -4, 7 \rangle$	13

Subarray	Sum
$\langle 3 \rangle$	3
$\langle 3, 10 \rangle$	13
$\langle 3, 10, -4 \rangle$	9
$\langle 3, 10, -4, 7 \rangle$	16

Subarray	Sum
$\langle -2 \rangle$	−2
$\langle -2, 3 \rangle$	1
$\langle -2, 3, 10 \rangle$	11
$\langle -2, 3, 10, -4 \rangle$	7
$\langle -2, 3, 10, -4, 7 \rangle$	14

Subarray	Sum
$\langle 1 \rangle$	1
$\langle 1, -2 \rangle$	−1
$\langle 1, -2, 3 \rangle$	2
$\langle 1, -2, 3, 10 \rangle$	12
$\langle 1, -2, 3, 10, -4 \rangle$	8
$\langle 1, -2, 3, 10, -4, 7 \rangle$	15

The maximum subarray sum is 16.

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

Solution:

```
mss([1,-2,3,10,-4,7])
│
├─ left = mss([1,-2,3])
│  │
│  ├─ left = mss([1,-2])
│  │  │
│  │  ├─ left = mss([1]) = 1
│  │  │
│  │  ├─ right = mss([-2]) = -2
│  │  │
│  │  ├─ cross = sum([1,-2]) = -1
│  │  │
│  │  └─ return max(1,-2,-1) = 1
│  │
│  ├─ right = mss([3]) = 3
│  │
│  ├─ cross = sum([1,-2,3]) = 2
│  │
│  └─ return max(1,3,2) = 3
│
├─ right = mss([10,-4,7])
│  │
│  ├─ left = mss([10,-4])
│  │  │
│  │  ├─ left = mss([10]) = 10
│  │  │
│  │  ├─ right = mss([-4]) = -4
│  │  │
│  │  ├─ cross = sum([10,-4]) = 6
│  │  │
│  │  └─ return max(10,-4,6) = 10
│  │
│  ├─ right = mss([7]) = 7
│  │
│  ├─ cross = sum([10,-4,7]) = 13
│  │
│  └─ return max(10,7,13) = 13
│
├─ cross = sum([3,10,-4,7]) = 16
│
└─ return max(3,13,16) = 16
```

(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 $\Theta(n \log n)$.