COMP3821/COMP9801 Workshop Week 8

Divide and Conquer

Jeremy Le and Xueyi Chee

2025T3

Announcements

- Problem Set 7 Submission due on Friday Week 8.
- Problem Set 8 Submission due on Friday Week 9.
  - Submissions on Gradescope.
- Help sessions on Friday 11am - 1pm in K17 G05.

Master Theorem

$T(n) = 2T(n/2) + O(n)$
$= O(n \log n)$

Suppose $T(n) = aT(n/b) + f(n)$, where $f(n) > 0$ is a non-decreasing function on the positive integers and $a \geq 1$, $b > 1$, where $a, b \in \mathbb{R}$. The critical exponent is $c^* = \log_b a$.

Master Theorem

(1) If $f(n) = O(n^{c^*-\varepsilon})$ for some $\varepsilon > 0$, then $T(n) = \Theta(n^{c^*})$.

(2) If $f(n) = \Theta(n^{c^*})$, then $T(n) = \Theta(n^{c^*} \log n)$.

(3) If $f(n) = \Omega(n^{c^*+\varepsilon})$ for some $\varepsilon > 0$, and for some $k < 1$ and n_0, the *regularity condition*

$$af(n/b) \leq kf(n)$$

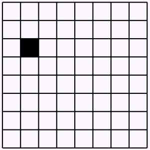
holds for all $n > n_0$, then $T(n) = \Theta(f(n))$.

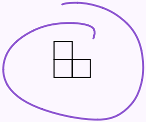
Intuitively, the critical polynomial n^{c^*} represents the number of leaves in the recursion tree.

Guided Problem 1

Guided Problem

You are given an $n \times n$ board with one of its cells missing (i.e. the board has a hole). You are given a supply of L-trominoes, each covering three cells as pictured below.



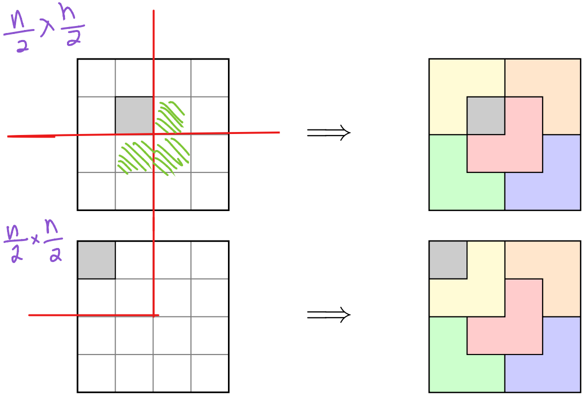


Describe and analyse a polynomial-time algorithm (in terms of n) to cover the entire board with the L-shaped trominoes.

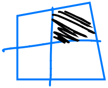
Note. *You may assume that n is a power of two and you know where the missing cell is in advance.*

Guided Problem 1 - Example

$n \times n$



Guided Problem 1 - Working



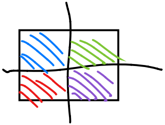
Given $n \times n$ board

Divide : 4 $\frac{n}{2} \times \frac{n}{2}$ boards through the centre

Base Case: 2×2 grid - can be ~~sol~~ solved trivially

Conquer: Place a tromino at the centre such that it contains one piece in the three quadrants without the missing cell.

Combine :



As each quadrant was filled independently combine tilings of each quadrant to fill ~~the~~ the $n \times n$ board.

Guided Problem 1 - Working

~~Correct~~ Correctness

Base Case: 2×2 – done trivially

IH: Assume true for $\frac{n}{2} \times n/2$ $n \times n$ board with one missing tile

Prove: true for a $n \times n$ $2n \times 2n$ board "

① Split into 4 $n \times n$ boards ← "Missing tiles must be in one quadrant.

→ ② By I.H. we can tile this. Place tromino to get equiv subproblem

③ Independently tiled, combine to fill $2n \times 2n$ board.

Guided Problem 1 - Working

Time Complexity

$$T(n) = 4T\left(\frac{n}{2}\right) + O(1)$$

$$c^{*} = \log_b a = \log_2 4 = 2 \qquad n^2$$

By case 1 of the Master Theorem since

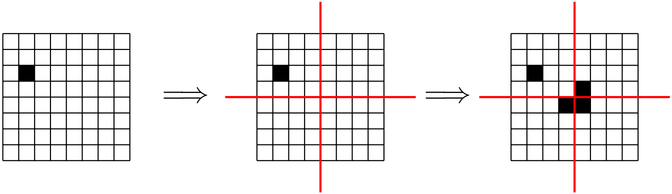
$f(n) = O(n^{2-\varepsilon})$ for $\varepsilon > 0$ we have

$$T(n) = \Theta(n^2)$$

Guided Problem 1 - Solution

Observation

- If we divide the problem into four boards of size $n/2 \times n/2$, then one of these boards must contain the missing tile. We place a tromino in the centre such that each board now contains a "missing" piece. i.e. In the three quadrants not containing the mising tile.



- This helps us create equivalent subproblems!

Guided Problem 1 - Solution

Algorithm

- If $n = 2$, there is only one way to fill a 2×2 board (with a missing cell) with a tromino.
- For $n > 2$, we divide our $n \times n$ board into four $n/2 \times n/2$ boards.
  - We place a tromino at the centre such that the tromino occupies the three remaining boards withou a mising cell.
  - Now that these subproblems are equivalent, we recurse on each board until we arrive at the base case.
- Finally combine these tromino placements to obtain the original board being filled with trominos.

Guided Problem 1 - Solution

Correctness (Induction)

- *Base Case.* Any 2×2 board can trivially be filled by a tromino and missing cell.
- *Inductive Hypothesis.* Assume that we can tile a $n/2 \times n/2$ board with a missing cell.
- *Inductive Step.* Consider a $n \times n$ board with a missing cell.
  - By our algorithm, we divide the board into four $n/2 \times n/2$ boards and place a tromino such that all board now obtain a missing cell.
  - Then by our hypothesis, we assumed we can tile such a board.
  - We can then combine these four tilings without overlapping and completes the inductive step.
- Hence by mathematical induction, our algorithm is correct.

Guided Problem 1 - Solution

Time Complexity

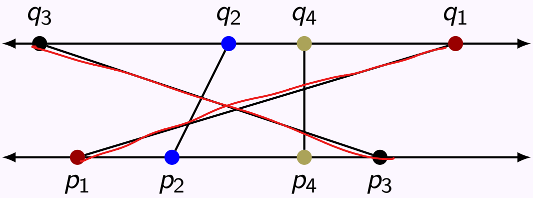
- Let $T(n)$ be a running time of the algorithm on a $n \times n$ board.
- Our algorithm, solves four subproblems of size $n/2$ at each stage with an takes $O(1)$ time to combine solutions.
- Thus our recurrence becomes $T(n) = 4T(\frac{n}{2}) + O(1)$.
- Now define $c^* = \log_2 4 = 2$ and since $f(n) = O(1)$, $T(n)$ satisfies Case 1 of the Master Theorem.
- Therefore, the running time is $\Theta(n^{c^*}) = \Theta(n^2)$.

Guided Problem 2

Guided Problem

You are given two lists of n points, one list $P = [p_1, \ldots, p_n]$ lies on the line $y = 0$ and the other list $Q = [q_1, \ldots, q_n]$ lies on the line $y = 1$. We construct n line segments by connecting p_i to q_i for each $i = 1, \ldots, n$. You may assume that the numbers in P are distinct and the numbers in Q are also distinct. Design an $O(n \log n)$ algorithm to return the number of intersections between every pair of distinct line segments.

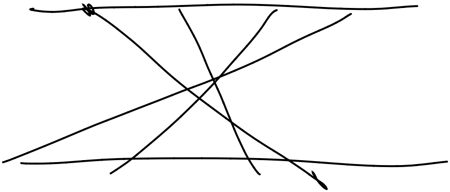
For example, the following instance



should return 5 since there are five intersections.

Guided Problem 2 Ideas / Observations

$O(n^2)$



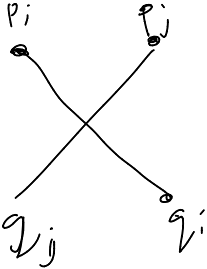
$O(n^2)$ inter.

Guided Problem 2 Ideas / Observations 2

$p_i \ q_i \qquad p_j \ q_j$

$p_i < p_j$

$q_i > q_j$



$A[1 \ldots n]$

$i < j$

$A[i] > A[j]$

Guided Problem 2 Ideas / Observations 2

idea: 1) sort lines by increasing P

2) look at Q in $\uparrow$ order

Claim # inversions in Q array

= # intersections

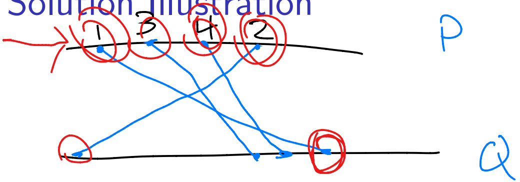
Guided Problem 2 Solution

- Sort lines in increasing P values
- In resulting Q values, count the number of inversions
- report number of inversions found as the number of line intersections

Guided Problem 2 Solution Illustration

P: [1, 5, 3, 4]

Q: [10, 2, 7, 9]



$Q' = [10, 7, 9, 2]$

Guided Problem 2 Proof Sketch

Consider an inversion in the Q'

$i < j$

$Q'[i] > Q'[j]$

$\Leftrightarrow$ $P[i] < P[j]$

$Q'[i] > Q'[j]$

The Karatsuba Multiplication Trick

The Karatsuba Multiplication Trick

- When multiplying two integers A and B of k digits, write

$$A = A_1 2^{\frac{k}{2}} + A_0$$

$$B = B_1 2^{\frac{k}{2}} + B_0$$

  Then with

  - $X = A_0 B_0$
  - $W = A_1 B_1$
  - $V = (A_1 + A_0)(B_1 + B_0)$

The Karatsuba Multiplication Trick (Cont.)

The Karatsuba Multiplication Trick

- Then
$$A \times B = W2^k + (V - W - X)2^{\frac{k}{2}} + X$$
- Using linear bit shifts and linear additions, we can multiply two k-digit numbers in time
$$T(k) = 3T\left(\frac{k}{2}\right) + O(k),$$
which is $\Theta(k^{\log_2 3}) = O(k^{1.59})$.

Extensions of the Karatsuba Multiplication Trick

Extensions of the Karatsuba Multiplication Trick

If we have a k-digit integer and an ℓ-digit integer where $\ell > k$, break the ℓ-digit integer into $\lceil \frac{\ell}{k} \rceil$ blocks of at most k digits. Using bit shifts, multiplication, and addition, each segment can be combined with the k-digit integer in $O(k^{1.59})$ time. In total, the time complexity is

$$O\left(\frac{\ell}{k} \cdot k^{1.59}\right) = O(\ell\, k^{0.59}).$$

$A :$

$B :$ $\qquad O(K^{1.59})$

$\dfrac{\ell}{k}$

$A =$ $\qquad \ell$

$\dots \qquad A_2 \qquad A_1 \qquad A_0$

$B =$ $\qquad k$

$$A = A_0 + 2^k A_1 + 2^{2k} A_2 \dots + 2^{\frac{\ell}{k}-1} A_{\dots}$$

$$B = B$$

$$AB = \underbrace{A_0 B}_{O(K^{1.59})} + 2^k \underbrace{A_1 B}_{O(K^{1.59})} + 2^{2k} A_2 B \dots + 2^{\frac{\ell}{k}-1} A_{\dots} B$$

$$O\left(\frac{l}{k}\right) \times O\left(k^{1.59}\right) + O\left(\frac{l}{k}\right)$$

$$= O\left(l k^{0.59}\right)$$