

Algorithms in the Time of COVID19
HW1, Divide and Conquer, Sorting¹ - Recitations² 3-4

October 9, 2020

¹Contact: Liangzu Peng (lp2528@nyu.edu).

²Hand-written notes available at: <https://www.dropbox.com/sh/x1z104c22d51pox/AACiJdDSKe2SDZw3qNljNApka?dl=0>

- ▶ Homework 1
- ▶ Divide & Conquer
- ▶ Sorting
 - ▶ Randomized quicksort
- ▶ Exercises

Homework 1

Solution (Problem 1)

$$\log_2 n \leq n^3$$

IT

$$n^{1/\log_2 n} = O(\log_2 \log_2 n) \underbrace{= O(\sqrt{\log_2 n}) = O(n^3) = O(n^{\log_2 n})}.$$

$$\Theta(1)$$

$$\log_2 k \leq \sqrt{k}, \quad k \geq 4$$
$$\Rightarrow \log_2 \log_2 n = O(\sqrt{\log_2 n}),$$

when $\log_2 n \geq 4$.

$$3 \leq \log_2 n, \quad \forall n \geq 8,$$
$$\Rightarrow n^3 = O(n^{\log_2 n})$$

Homework 1

$$T(n) = O(n^2 \log n),$$

$$\exists c: T(n) \leq C n^2 \log n,$$

$$\log n \text{ (multiplied by)} \\ n^2$$

Problem 2. Find an asymptotically tight bound for

$$T(n) = \theta(n^2 \log n)$$

Induction hypo. $T(n) = 2T(2n/3) + T(n/3) + n^2.$

2. ① $T(n) \leq C n^2 \log n, \forall n < m$

② $T(m) \stackrel{1}{=} 2T(2m/3) + T(m/3) + m^2$

$\stackrel{2}{\leq} 2 \left[C \left(\frac{2}{3} m \right)^2 \log \left(\frac{2}{3} m \right) \right] + C \left(\frac{m}{3} \right)^2 \log \frac{m}{3} + m^2$

rearrangement { 3. $= \frac{8}{9} C m^2 \left[\log m + \log \frac{2}{3} \right] + \frac{1}{9} C m^2 \left[\log m + \log \frac{1}{3} \right] + m^2$

4. $= \underbrace{C m^2 \log m}_{\text{green}} + \frac{8}{9} C m^2 \log \frac{2}{3} + \frac{1}{9} C m^2 \log \frac{1}{3} + m^2$

Homework 1 $\leq cm^2 \log m$, when $\frac{8}{9}cm^2 \log \frac{2}{3} + \frac{1}{9}cm^2 \log \frac{1}{3} + m^2 \leq 0$

$$\Leftrightarrow \frac{8}{9}C \cdot \log \frac{2}{3} + \frac{1}{9}C \log \frac{1}{3} + 1 \leq 0$$

$$\Leftrightarrow 1 \leq \left(\frac{8}{9} \log \frac{2}{3} + \frac{1}{9} \log \frac{1}{3} \right) C$$

$$\Leftrightarrow C \geq \frac{1}{\frac{8}{9} \log \frac{2}{3} + \frac{1}{9} \log \frac{1}{3}}$$

Problem 2 Find an explicit formula for

$$C_n = n + 1 + \frac{2}{n} \sum_{k=0}^{n-1} C_k \quad (1)$$

with $C_0 = 0$.

$$C_{n+1} = n+2 + \frac{2}{n+1} \sum_{k=0}^n C_k \quad (2)$$

$$\Leftrightarrow (2) \cdot (n+1) - (1) \cdot n =$$

$$\Rightarrow (C_{n+1}) \cdot (n+1) - \underline{n C_n} = \underline{(n+2)(n+1)(n+1)} - n+2 C_n$$

$$\Leftrightarrow (n+1)C_{n+1} = 2(n+1) + (n+2)C_n$$

Homework 1 $\Leftrightarrow \frac{C_{n+1}}{n+2} = \frac{2}{n+2} + \frac{C_n}{n+1}$

$D_n = \frac{C_n}{n+1} \Rightarrow D_{n+1} = D_n + \frac{2}{n+2}$

Problem 4(a) of Homework 1.

```
int F(int x) {
    assert(x >= 1);

    if (x == 1 || x == 2)
        return 1;
    else
        return 2 * F(x-1) - F(x-2);
}
```

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

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

$$\begin{aligned} \rightarrow T(n) &= T(n-1) + T(n-2) \\ \uparrow & T(n) = T(n-1) + T(n-2) + 1 \end{aligned}$$

Homework 1

Given A, B, C, D, n .

~~Find $a \in A$~~

Find $a \in A, b \in B, c \in C, d \in D$
s.t. $a+b+c+d=0$

Problem 5(b) of Homework 1.

$$n^2 \begin{cases} \textcircled{1} \underline{A+B} = \{a+b: a \in A, b \in B\} \\ \textcircled{2} \underline{C+D} = \{ \dots \} \end{cases}$$

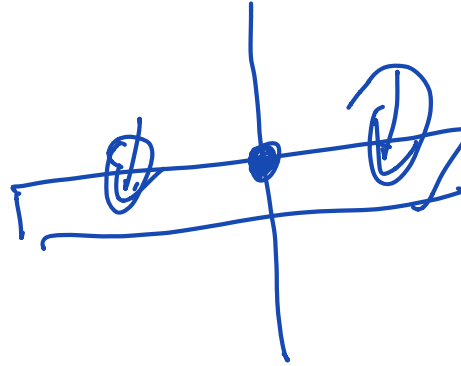
$$O(n^2 \log n) = n^2 \log(n^2)$$

1. Sorting + Linear scan
2. Sort $A+B$, binary search on $C+D$
3. Hashing.

1 million #

$$O(n^2)$$

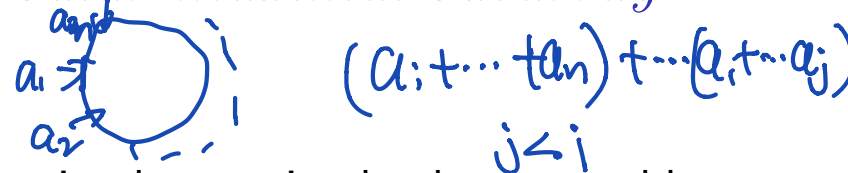
Divide & Conquer (Lecture 5)



- ▶ ^{2D} Counting inversions
- ▶ The maximal subarray
- ▶ Median-of-medians³

³This algorithm was from the paper "Time Bounds for Selection," by Manuel Blum, Robert W. Floyd, Vaughan Pratt, Ronald L. Rivest, and Robert E. Tarjan (Journal of Computer and System Sciences, 1973). Four of the authors won the Turing Award for their contributions in different subfields of computer science (google them).

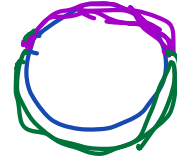
The circular maximal subarray $\in \dots \dots \dots \rightarrow$ $\sum_{i < j} a_i + \dots + a_j$



$(a_1 + \dots + a_n) + \dots + (a_1 + \dots + a_j)$
 $j < i$

In the circular maximal subarray problem, one is given n numbers, arranged on a circle C , and aims to find the maximal sum of the consecutive numbers.

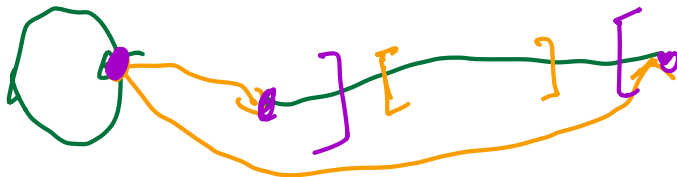
Problem 3. Prove the following lemma.



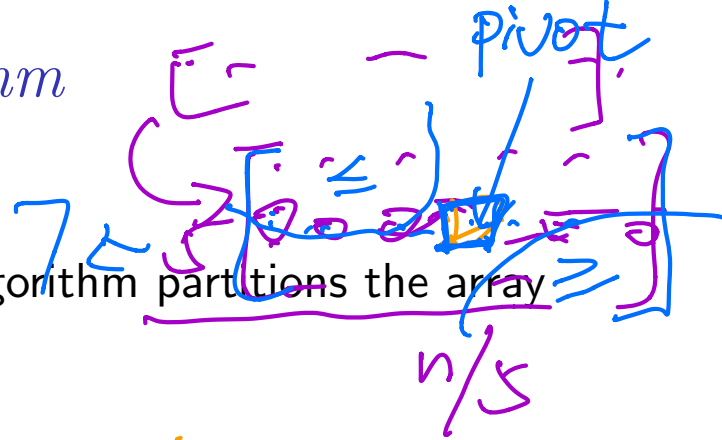
Lemma

The sum of a sequence S of consecutive numbers on the circle C is maximal if and only if the sum of the rest consecutive numbers $(C \setminus S)$ is minimal.

Problem 4. With the above lemma, give an $O(n \log n)$ algorithm that solves the circular maximal subarray problem.



The median-of-medians algorithm



Recall that the median-of-medians algorithm partitions the array into subarrays of size 5.

Problem 5.



What is the running time of the ~~median-of-medians~~ ^{k-selection} algorithm if the array is partitioned into subarrays of size 7?

- ▶ What is the running time of the ~~median-of-medians~~ ^{k-selection} algorithm if the array is partitioned into subarrays of size

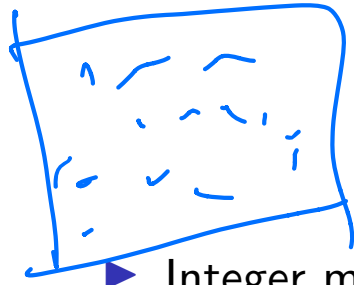
3?⁴ $2(n \log n)$

20/5

⁴See Problem 9.3-1 in CLRS. See also the paper “Select with Groups of 3 or 4,” by Ke Chen and Adrian Dumitrescu, available at:

https://link.springer.com/chapter/10.1007/978-3-319-21840-3_16.

Divide & Conquer (Lecture 6)



► Integer multiplication

► Matrix multiplication⁵

► The closest pair of points⁶

11969
 $\Theta(n^{\log_2 7})$

$\Theta(n^w)$

$w=3$
↓
 $w=\log_2 7$
↓
⋮
↓
 $w \leq 2.4$
↓
?

$w=2$

Conjecture

⁵Read the first page of this paper:

<https://dl.acm.org/doi/pdf/10.1145/2213977.2214056>

⁶This is a problem of computational geometry. If you are interested in this subject, start with this course:

<https://www.edx.org/course/computational-geometry>

Matrix multiplication

$$\begin{bmatrix} A_1 & A_2 \\ A_3 & A_4 \end{bmatrix} \begin{bmatrix} B_1 & B_2 \\ B_3 & B_4 \end{bmatrix} \xrightarrow{8} \begin{bmatrix} A_1 B_1 + A_2 B_2 & \dots \\ \dots & \dots \end{bmatrix}$$

Problem 6. Modify Strassen's algorithm to multiply $n \times n$ matrices where n is not an exact power of 2. The resulting algorithm should be of complexity $O(n^{\log 7})$.

make it an exact power of 2.

The closest pair of points

Algorithm

- Find the dividing line by computing the median using the x-value. $\leftarrow O(n)$.
- Recursively solve closest pair in Q , and closest pair in R . $\leftarrow 2T(\frac{n}{2})$.
- Using a linear scan, we remove all points not within the narrow region. $\leftarrow O(n)$
- Then, we sort the points by their y-value in increasing order. $\leftarrow O(n \log n)$
- For each point, we just compute distances with the next eleven points in this ordering. $\leftarrow O(n)$
(Two points within two layers must be within 11 points in the y-order, i.e. at most 10 points between them)
- Return the minimum.

Bottleneck.

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

Problem 7. Let $T(n) = 2T(n/2) + O(n \log n)$. Prove that

$$T(n) = O(n(\log n)^2) \quad (1)$$

Sorting

Comparison-based sorting algorithms⁷

- ▶ Insertion sort
- ▶ Merge sort
- ▶ Quicksort
- ▶ Heapsort

⁷By “comparison-based sorting algorithms,” we mean algorithms that accesses the input array only via comparisons between pairs of elements, and never directly accesses the value of an element.

Randomized quicksort

- ▶ Given n numbers x_1, \dots, x_n assume that y_1, \dots, y_n are the same values in increasing order.

Randomized quicksort

- ▶ Given n numbers x_1, \dots, x_n assume that y_1, \dots, y_n are the same values in increasing order.
- ▶ Over the course of quicksort we instead choose the pivot element *independently and uniformly at random* to which we will refer as the *randomized* quicksort algorithm.
 - ▶ In the standard quicksort algorithm the first element in the subarray is chosen as the pivot.

Randomized quicksort

- ▶ Given n numbers x_1, \dots, x_n assume that y_1, \dots, y_n are the same values in increasing order.
- ▶ Over the course of quicksort we instead choose the pivot element *independently and uniformly at random* to which we will refer as the *randomized* quicksort algorithm.
 - ▶ In the standard quicksort algorithm the first element in the subarray is chosen as the pivot.
- ▶ **Goal:** we are going to show that the *expected* running time of this randomized quicksort algorithm is $O(n \log n)$.

Randomized quicksort (cont.)

- ▶ Let X be the total number of comparisons.
- ▶ The expected running time = the expectation $\mathbb{E}[X]$ of X .
- ▶ **Goal:** prove $\mathbb{E}[X] = O(n \log n)$.

Randomized quicksort (cont.)

- ▶ Let X be the total number of comparisons.
 - ▶ The expected running time = the expectation $\mathbb{E}[X]$ of X .
 - ▶ **Goal:** prove $\mathbb{E}[X] = O(n \log n)$.
- ▶ For $i < j$ let X_{ij} be a random variable that takes on the value 1 if y_i and y_j are compared at any time during the randomized quicksort algorithm, and 0 otherwise.

Randomized quicksort (cont.)

- ▶ Let X be the total number of comparisons.
 - ▶ The expected running time = the expectation $\mathbb{E}[X]$ of X .
 - ▶ **Goal:** prove $\mathbb{E}[X] = O(n \log n)$.
- ▶ For $i < j$ let X_{ij} be a random variable that takes on the value 1 if y_i and y_j are compared at any time during the randomized quicksort algorithm, and 0 otherwise.

Problem 12. What is the relation between X and X_{ij} 's?

Randomized quicksort (cont.)

- ▶ Let X be the total number of comparisons.
 - ▶ The expected running time = the expectation $\mathbb{E}[X]$ of X .
 - ▶ **Goal:** prove $\mathbb{E}[X] = O(n \log n)$.
- ▶ For $i < j$ let X_{ij} be a random variable that takes on the value 1 if y_i and y_j are compared at any time during the randomized quicksort algorithm, and 0 otherwise.

Problem 14. What is the relation between X and X_{ij} 's?

Problem 15. What is the relation between $\mathbb{E}[X]$ and $\mathbb{E}[X_{ij}]$'s?

Randomized quicksort (cont.)

- ▶ Since X_{ij} is an indicator random variable which can only be 0 or 1, $E[X_{ij}]$ is equal to the probability that $X_{ij} = 1$.
 - ▶ i.e., the probability that y_i and y_j are compared during the algorithm.

Randomized quicksort (cont.)

Recall that $y_1 \leq \cdots \leq y_i \leq \cdots \leq y_j \leq \cdots \leq y_n$.

Problem 16. Show that y_i and y_j are compared *if and only if* either y_i or y_j is the first pivot selected by the randomized quicksort algorithm from the set $Y^{ij} = \{y_i, \dots, y_j\}$.

Randomized quicksort (cont.)

Recall that the pivots are chosen independently and uniformly at random.

Problem 17. What is the probability that y_i or y_j is the first pivot selected from $Y^{ij} = \{y_i, \dots, y_j\}$?

Problem 18. What is $\mathbb{E}[X_{ij}]$?

Randomized quicksort (cont.)

Problem 19. Prove that

$$\mathbb{E}[X] = \sum_{i=1}^{n-1} \sum_{j=i+1}^n \frac{2}{j-i+1} = (2n+2) \sum_{k=1}^n \frac{1}{k} - 4n. \quad (2)$$

Remark 1. The last problem in Recitation 1 gives

$$\sum_{k=1}^n k = \Theta(\log n). \quad (3)$$

Remark 2. Compare the formula of C_n in Homework 1 with $\mathbb{E}[X]$.

Exercises: Lower bound for comparison-based sorting

Problem 20. Prove the following theorem.

Theorem

Every deterministic comparison-based sorting algorithm performs at least $c \cdot n \log_2 n$ comparisons on some input array of length $n \geq 1$, where $c > 0$ is a constant.

Hint: Let $f(n)$ be the number of comparisons made by a deterministic comparison-based sorting algorithm when sorting an array of length n . Prove that $2^{f(n)} \geq n!$. Then prove that $f(n) = \Omega(n \log_2 n)$.

Faster sorting algorithms - stronger assumptions on data

- ▶ Counting sort
- ▶ Bucket sort

Exercises: Bucket sort

Expected Cost

Sort each $B[i]$ with insertion sort

$$|B[0]| + \dots + |B[n-1]| = n$$

$$\text{Cost: } |B[0]|^2 + |B[1]|^2 + \dots + |B[n-1]|^2$$

n numbers are independently and uniformly drawn from $[0,1)$

$$E[|B[i]|] = 1$$

$$E[|B[i]|^2] = 2 - 1/n$$

Expected cost $O(n)$

Problem 21. Prove $\mathbb{E}[n_i^2] = 2 - 1/n$, where $n_i := |B_i|$.

Exercises: The maximal subarray

Problem 22. Give a linear time algorithm for the maximal subarray problem.

Exercises: comparison

Problem 23. Show that you need exactly $n - 1$ comparisons to find the smallest element out of n elements. Prove that the second smallest of n elements can be found with $n + \lceil \log n \rceil - 2$ comparisons in the worst case.

Exercise: maximum and minimum

Problem 24. Assume that n is even. Show that the maximum and minimum of n numbers can be found by $3n/2 - 2$ comparisons.

Exercises: deterministic quicksort in $O(n \log n)$ time

Problem 25. Give a variant of the quicksort algorithm which runs in $O(n \log n)$ time in the worst case. Your algorithm should be deterministic.

Exercises: complex numbers

Problem 26. Give an algorithm that multiply the complex numbers $a + bi$ and $c + di$ using only three multiplications of real numbers. The algorithm takes a, b, c , and d as inputs and outputs the real component $ac - bd$ and the imaginary component $ad + bc$.

Exercises: the maximal rectangle sum

Problem 27. Given an $n \times n$ 2D-array A where each entry $A[i][j]$ is an integer, the maximal rectangle sum problem requires to find a rectangle, specified by four indexes i_1, i_2, j_1, j_2 , so as to maximize

$$\sum_{i=i_1}^{i_2} \sum_{j=j_1}^{j_2} A[i][j].$$

Give an algorithm of complexity no more than $O(n^3 \log n)$ to solve this maximal rectangle sum problem.