Algorithms (Sep 2020 - Jan 2021)

Given a problem, we would like to design an algorithm that solves instances of this problem efficiently.

<u>Example</u>: Sorting numbers
Input: a set S of numbers
Output: the elements of S in sorted order

S: | s_1 | s_2 | ... | s_n | S is described as an array

<u>Mergesort</u>: divide the array into 2 halves
- sort $s_1, \dots, s_{n/2}$: let S_1 be this sorted list
- sort $s_{n/2+1}, \dots, s_n$: let S_2 be this sorted list
- merge S_1 and S_2 to obtain the overall sorted list

This approach is called Divide-and-Conquer.

(1) divide the problem into a number of subproblems.
(2) conquer the subproblems by solving them recursively
(3) combine the solutions of the subproblems into the solution for the original problem.

<u>Finding the minimum of n numbers</u>
Given an array $A[1..n]$ of n numbers, determine $\min_{1 \le i \le n} A[i]$.
- We know this can be computed using $n-1$ comparisons.

<u>Exercise</u>: Show that we need $n-1$ comparisons.
That is we cannot compute $\min_{1 \le i \le n} A[i]$ using less than $n-1$ comparisons.

Finding the median

We are again given an array $A[1..n]$ of n numbers.
For convenience, assume n to be odd.
We need to find the "middle" element in sorted order.

Ex. 130, 83, 220, 50, 2, 29, 7, 13, 98, 30, 66, 115, 300, 27, 92.

Let us partition the input into buckets of size k, where k is a small constant

Let $k = 3$.

130	50	7	30	300
83	2	13	66	27
220	29	98	115	92

– Sort each bucket.

83	2	7	30	27	
130	29	13	66	92	→ set S of medians
220	50	98	115	300	

– Recursively find the median of S

In our example above, this would be 66.
Call this element m.

Exercise. Show that at least $n/4$ elements are $\leq m$ in the input. Similarly, show that at least $n/4$ elmts are $\geq m$ in the input.

Hint Consider this table with k rows and n/k columns.

7	2	30	27	83
13	29	66	92	130
98	50	115	300	220

this is the set S in sorted order.

- Let us compare every number with m and get the following partition:

$$\underset{\text{numbers} < m}{\mathcal{L}} \qquad \underset{\substack{\text{all occurrences} \\ \text{of } m}}{M} \qquad \underset{\text{numbers} > m}{R}$$

The exercise above tells us that $|\mathcal{L}| \leq 3n/4$ and $|R| \leq 3n/4$.

* if $|\mathcal{L}| \geq n/2$ then find the $\left(\frac{n+1}{2}\right)$-th largest element of $\mathcal{L}$.
* if $|\mathcal{L}| + |M| < \frac{n+1}{2}$ then find the $\left(\frac{n+1}{2} - |\mathcal{L}| - |M|\right)$-th largest element of R.
* else return m.

- Please check that the algorithm is correct.

What is the time taken by our algorithm?

$$T(n) \leq k.\log k \cdot \frac{n}{k} + T\left(\frac{n}{k}\right) + n + T\left(\frac{3n}{4}\right)$$

$$= cn + T\left(\frac{n}{k}\right) + T\left(\frac{3n}{4}\right) \quad \text{where } c \text{ is a constant}$$

What happens when $k = 3$? when $k = 4$?

<u>Exercise</u> Show that $T(n)$ is $O(n)$ when $k = 5$.

<u>A more general problem</u>: Find-Element(A, t)

Here we want to find the t-th element in sorted order in A, where $1 \leq t \leq n$.

1. Partition A into $n/5$ buckets, where each bucket has 5 elements
   - Sort each bucket.
   - Let S = set of medians $(|S| = n/5)$.

2. Call Find-Element(S, $\lceil \frac{|S|}{2} \rceil$).
Let m be the value returned.

3. Use m as a pivot to partition A into L, M, R where $L = \{\text{elements} < m\}$ and $R = \{\text{elements} > m\}$.
Note that L and R are multisets here.

4. Depending on $|L|$, $|M|$ call Find-Element(L, t) or Find-Element$(R, t - |L| - |M|)$ or return m.

$$T(n) \leq T\left(\frac{n}{5}\right) + T\left(\frac{3n}{4}\right) + 4n \quad \text{for } n \geq 5$$
$$= O(1) \quad \text{for } n < 5$$

This solves to $T(n) = O(n)$.

Multiplying Two Polynomials

Let $A(x)$ and $B(x)$ be 2 polynomials of degree $(n-1)$ each with complex coefficients.

Let $A(x) = a_{n-1} x^{n-1} + \dots + a_1 x + a_0$ where $a_i \in \mathbb{C} \ \forall i$

$B(x) = b_{n-1} x^{n-1} + \dots + b_1 x + b_0$ where $b_i \in \mathbb{C} \ \forall i$.

The product of $A(x)$ and $B(x)$ is the polynomial?

$$P(x) = \sum_{i=0}^{2(n-1)} p_i x^i \quad \text{where } p_i = \sum_{j=0}^{i} a_j b_{i-j}$$

What is the time needed to compute $P(x)$?
– The high school algorithm takes $\Theta(n^2)$ time.

Can we do better?

Another way of representing a polynomial is the point-value representation.

- a point-value representation of $A(x)$ is a set of n point-value pairs $\{(x_0, y_0), \ldots, (x_{n-1}, y_{n-1})\}$, where x_i's are distinct and $y_i = A(x_i)$ for all i.

If $A(x)$ and $B(x)$ are given in point-value representation evaluated at the same $2n-1$ points, their product can be obtained easily.

$$A(x): \{(x_0, y_0), \ldots, (x_{2n-2}, y_{2n-2})\}$$

$$B(x): \{(x_0, z_0), \ldots, (x_{2n-2}, z_{2n-2})\}$$

$$P(x): \{(x_0, y_0 z_0), \ldots, (x_{2n-2}, y_{2n-2} z_{2n-2})\}$$

Let us consider the following approach:

Step 1: Convert $A(x)$ and $B(x)$ to point-value representation.

Step 2: Obtain their product $P(x)$ in point-value representation.

Step 3: Convert $P(x)$ back to coefficient form.

How do we evaluate a polynomial $A(x)$ at some point $x = x_0$?
- use Horner's rule

$$(((a_{n-1} x_0 + a_{n-2}) x_0 + a_{n-3}) x_0 + a_{n-4}) \ldots$$

- takes $\Theta(n)$ time.

So Step 1 seems to take $\Theta(n^2)$ time.
Step 2 takes $O(n)$ time.
Step 3 : ?