OH TODAY: 10pm-11pm ET.

Exam: Tue., March 02.
- Details will be posted by Tuesday next week.

**Simplified Master Theorem.**

Let $a \geq 1$, $b > 1$, and $k > 0$ be constants and let

$$T(n) = aT\left(\frac{n}{b}\right) + \Theta(n^k)$$

defined for $n \geq 0$. The base case $T(1)$ is some constant. Then

\[
\begin{cases}
\text{Case I:} & \text{if } a > b^k \text{ then } T(n) = \Theta(n^{\log_b a}) \\
\text{Case II:} & \text{if } a = b^k \text{ then } T(n) = \Theta(n^k \log n) \\
\text{Case III:} & \text{if } a < b^k \text{ then } T(n) = \Theta(n^k)
\end{cases}
\]
Examples

(i) \( T(n) = 4 T\left(\frac{n}{2}\right) + n^1 \)

\( a = 4, \ b = 2, \ k = 1 \quad \therefore \ a > b^k \)

\( \therefore T(n) = \Theta\left( n^{\frac{\log_2 4}{2}} \right) = \Theta(n^2). \)

(ii) \( T(n) = T\left(\frac{n}{3}\right) + n \)

\( a = 1, \ b = 3, \ k = 1 \quad \therefore \text{By Con. III of the Master theorem,} \)

\( T(n) = \Theta(n). \)

(iii) \( T(n) = 9 T\left(\frac{n}{3}\right) + n^{2.5} \)

\( a = 9, \ b = 3, \ k = 2.5 \quad \therefore \ a < b^k \)

\( \therefore T(n) = \Theta(n^{2.5}). \)

Selection

Input: Array \( A \) of \( n \) distinct nos.

Objective: To find the \( i^{th} \) smallest element in \( A \).
CIS 110 way: Sort A & then find the i-th smallest element.

Running time: $O(n \log n)$.

Target running: $O(n)$.

Algorithm $(\text{Select} (A, i))$.

1. Divide A into $\left\lceil \frac{n}{5} \right\rceil$ groups of 5 elements in each group (except the last group). $\Theta(n)$.

2. Find the median element in each group.

3. Let $S$ be the set of their medians.

4. Partition the array A around x. Let rank(x) = k.

5. If $i = k$ then return x. $\Theta(1)$
6. else if \( i < k \) then

find the \( i \)th smallest element in the left partition.

\[
\text{Select} \ (A[1 \cdots k-1], i) \quad T \left( \frac{7n}{10} + 6 \right)
\]

7. else

find the \( (i-k) \)th smallest element in the right partition.

\[
\text{Select} \ (A[k+1, n], i-k) \quad T \left( \frac{7n}{10} + 6 \right) \quad \frac{1}{2} \left( \frac{n}{3} \right)
\]

\[
\# \text{elements} > x \quad \geq 3 \left( \frac{1}{2} \left( \left\lceil \frac{n}{5} \right\rceil - 2 \right) \right) = \left( \frac{3n}{10} - 6 \right)
\]

\[
\# \text{elements} < x
\]
The side that I will recurse on in Steps 6 or 7 will have

\[ n - \left( \frac{3n}{10} - 6 \right) = \frac{7n}{10} + 6. \]

Runtime recurrence:

\[ T(n) \leq \begin{cases} O(1), & n < n_0 = 140 \\ T\left( \left\lceil \frac{n}{5} \right\rceil \right) + T\left( \frac{7n}{10} + 6 \right) + an, & n \geq n_0 \\ \end{cases} \]

where a is some const.

Induction on \( n \). That is, we will show that for some constant \( c \), \( T(n) \leq c \cdot n \), \( \forall n \geq n_0 \).

IH: Assume that \( T(j) \leq c_j, \forall 0 \leq j < k \).

BC: \( T(j) \leq c \cdot j \), \( \forall j \leq n_0 \). (we can choose \( c \) to be sufficiently large).
\( IS: \) Want to prove that the claim when \( n = k. \)

\[
T(k+1) \leq T\left(\frac{k}{5}\right) + T\left(\frac{7k}{10} + 6\right) + a(k+1). \quad \text{let} \quad k' = k+1
\]

\[
T(k') \leq T\left(\left\lfloor \frac{k}{5} \right\rfloor\right) + T\left(\frac{7k}{10} + 6\right) + a k'
\]

\[
\leq c \left(\frac{k}{5} + 1\right) + c \left(\frac{7k}{10} + 6\right) + ak \quad \text{(By IH)}
\]

\[
= \frac{ck}{5} + c + \frac{7ck}{10} + 6c + ak
\]

\[
= \frac{9ck}{10} + 7c + ak
\]

\[
\leq ck + \left(\frac{c}{10} + 7c + ak\right)
\]

\[
\text{only if} \quad -\frac{ck}{10} + 7c + ak \leq 0.
\]

\[
\leq \frac{ck}{10}
\]

\[
ck' \leq c \left(\frac{k}{10} - 7\right) \geq ak
\]

\[
c \left(\frac{k - 70}{10}\right) \geq ak
\]
\[ T(n) = T \left( \frac{95n}{100} \right) + n. \]

\[ a = 1 \quad b = \frac{100}{95} \quad k = 1 \]

\[ a < b^k \quad \therefore \quad T(n) = \Theta(n). \]

\[ T(n) = 3T \left( \frac{n}{2} \right) + n \]

\underline{Stacks & Queues.}
Abstract Data Type (ADT)

- Data
- Operation

Operation for Stack: Push & Pop.

LIFO

Incremental Strategy:

- Everytime the array becomes full, we increment the size of the array by c.
- Initial size of the array: c

Push (obj)

// s: size of the stack. (# of elem)
// a: array size
// c: increment & initial size
\[ A[s] \leftarrow \text{obj} \]

\[ s \leftarrow s + 1 \]

\[ \text{if } s = a \text{ then} \]

\[ a \leftarrow a + c \]

Copy contents of \( A \) into the new array \( a + c \).

Sequence of operations: What is the time for

- a seq. of \( n \) operations?
- sequence of \( n \) push operations.

\[ c = 2 \]

Total cost of \( n \) Push operations
\[ = O(n) \times \]

\[ = \text{cost to add } n \text{ elements } + \text{cost to copy } + \text{cost array allocation.} \]

\[ = n + (c + 2c + 3c + \cdots + n) + \]

\[ (c + 2c + 3c + \cdots + n + n + c) \]

\[ \sum_{i=1}^{k} i = \frac{k(k+1)}{2} \]

\[ = n + c \left( 1 + 2 + 3 + \cdots + \frac{n}{c} \right) + c \left( 1 + 2 + \cdots + \frac{n}{c} + \frac{n}{c} + 1 \right) \]

\[ = n + c \left( \frac{\frac{n}{c}(\frac{n}{c} + 1)}{2} \right) + c \left( \frac{\frac{n}{c} + 1}{2} \left( \frac{n}{c} + 2 \right) \right) \]

\[ = \Theta \left( n^2 \right) \]