COL 351: Analysis and Design of Algorithms

Lecture 24

Divide and Conquer

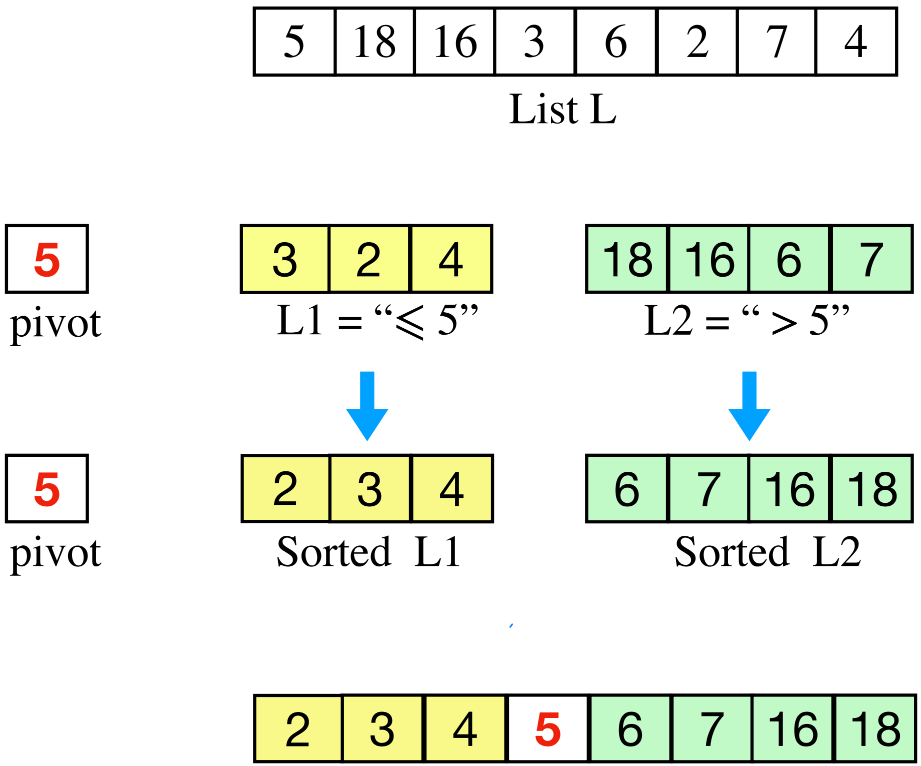
1. **Divide the main problem into smaller subproblems.**

2. **Solve the sub-problems recursively**

3. **Combine**

Quick Sort

Worst Case = $O(n^2)$
if we pick PIVOT naively



1. Divide into subproblems
2. Solve the sub-problems recursively
3. Combine

Quick Sort Variations

1. Deterministic Quick Sort:
   We take Pivot to be Median, i.e. $\left(\dfrac{n}{2}\right)^{th}$ element in SORTED list L.

2. Randomized Quick Sort:
   We take Pivot to be a RANDOM element of the list L.

$T(n) = 2T(n/2) + \text{Time to find Median}$

Deterministic Quick Sort

DetQuickSort(L)

x = Median of list L; /* pivot is Median*/

Initialise L1 and L2 to be empty lists;

For each $(y \in L \setminus x)$:

If $(y \leqslant x)$: L1.append(y);

If $(y > x)$: L2.append(y);

Return *DetQuickSort*(L1) $\circ\, x \circ$ *DetQuickSort*(L2);

How efficiently can we find median?

New Problem: Computing kth-smallest element

Given: List L of size n and an integer $k \in [1,n]$.

Find: The element at **index** k after we **sort** the list L.

Example:

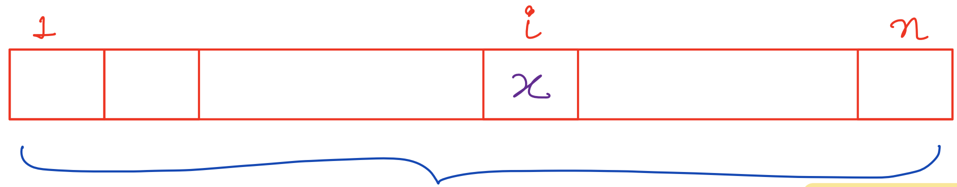
$L = [77, 33, 88, 66, 44, 11, 22, 55]$

If $k = 2$ then we need to return 22.

Sorted L: | 11 | 22 | 33 | 44 | - - - | 88 |

Computing kth-smallest element

Sorted L



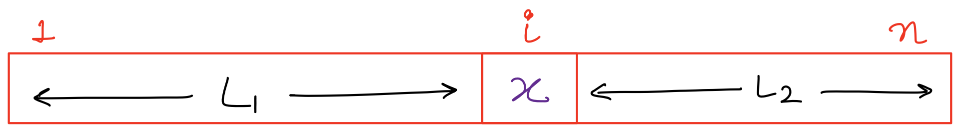
Pick an element x from L

$i = \text{Rank}(x)$

How to find x? SEE Next Pg.

$L_1 = (L - x)_{\leq x}$

$L_2 = L_{>x}$



If $k < i$

Search(L_1, k)

If $k = i$

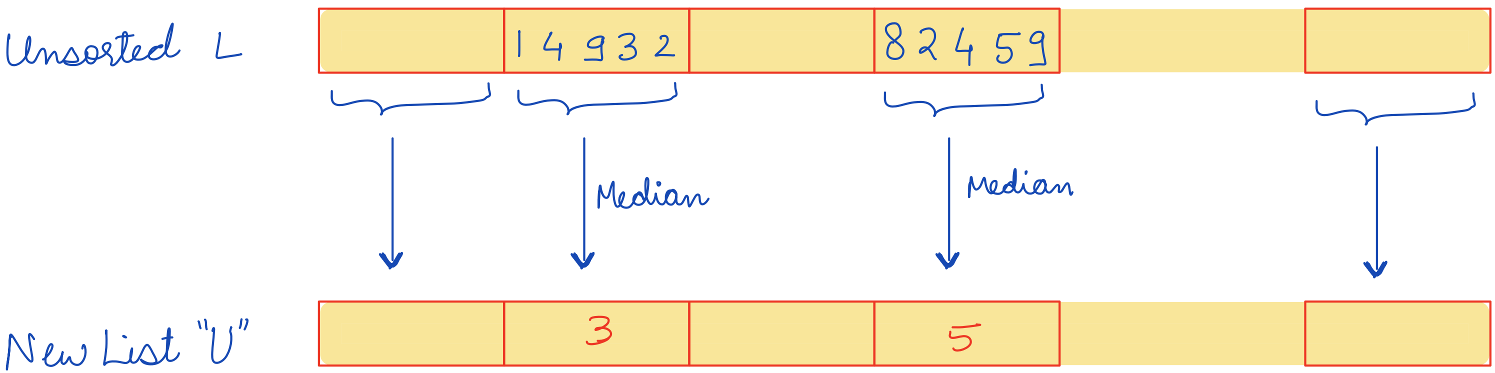
STOP

If $k > i$

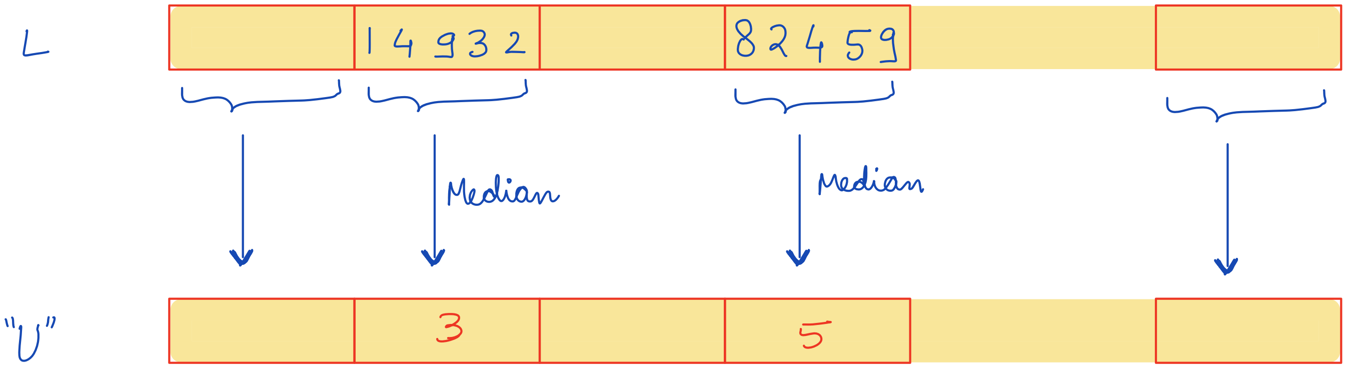
Search$(L_2, k-i)$

HOW TO FIND GOOD x?

$\lceil n/5 \rceil$ chunks of size 5



$x = \text{Median}(U)$



CLAIM: $\text{Rank}(x = \text{Median of } U) \in \left[\frac{3n}{10}, \frac{7n}{10}\right]$

Proof:

In U, $n/10$ elements are larger than x

$\Rightarrow$ In L, $3n/10$ elements are larger than x

$\Rightarrow |L_1| \leq \frac{7n}{10}$ Similarly, $|L_2| \leq \frac{7n}{10}$

Computing k$^{\text{th}}$-smallest element

Search(L, k)

1. Divide L into $\lceil n/5 \rceil$ groups each of size 5.
2. U = List containing medians of all $\lceil n/5 \rceil$ groups.
3. $x = Order(U, \lceil n/10 \rceil)$.

4. Rank = position of x in sorted L.
5. If (Rank $= k$) return x.

6. If (Rank $> k$):

   L_1 = Elements of $L \backslash x$ smaller than x

   return *Search*$(L_1,\ k)$.

7. If (Rank $< k$):

   L_2 = Elements of $L \backslash x$ larger than x

   return *Search*$(L_2,\ k - \text{Rank})$.

Steps 1-3: $O(n) + T(n/5)$

Steps 4-5: $O(n)$

Steps 6-7: ? $T(7n/10)$

Note: $n/10$ elements in U are $\geq x$.

So,
at least $3(n/10)$ elements in L are $\geq x$.

Therefore, $|L_1| \leq 7(n/10)$

Similarly, $|L_2| \leq 7(n/10)$

Time complexity follows the relation

$$T(n) \leq T(n/5) + T(7n/10) + cn$$

SUBSTITUTE

$$\boxed{T(n) \leq \alpha n}$$

<u>RHS</u>

$$\left(\frac{\alpha n}{5} + \alpha \frac{7n}{10} + Cn\right) \leq \alpha n$$

$$\frac{9n}{10}\alpha + Cn \leq \alpha n$$

$$\Rightarrow \alpha = 10c$$

Computing k^{th}-smallest element

Lemma: The k^{th} smallest element of a list L of size n is computable in $O(n)$ time.

Corollary: The Median of a list L of size n is computable in $O(n)$ time.

Time complexity of Deterministic Quick Sort follows the relation

$$T(n) \leq 2T(n/2) + cn$$

Randomized Quick Sort

$RandQuickSort(L)$

x = Random element of list L;

Initialise L1 and L2 to be empty lists;

For each ($y \in L \backslash x$):

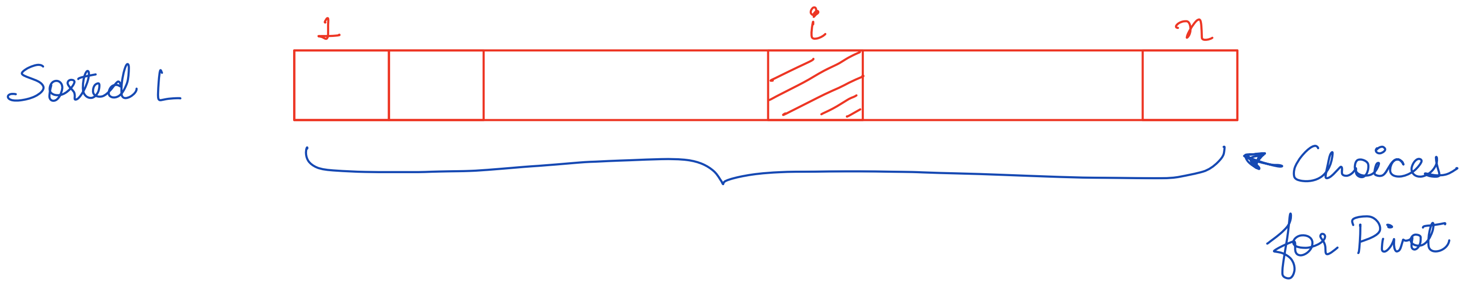
If ($y \leqslant x$) : L1.append(y);

If ($y > x$) : L2.append(y);

Return $RandQuickSort$(L1) $\circ$ x $\circ$ $RandQuickSort$(L2);

EXPECTED TIME TO SORT?

$$\text{Expected time} = \frac{1}{n} \sum_{i=1}^{n} \Big(E\big(T(i-1)\big) + E\big(T(n-i)\big) \Big) + cn$$



If Pivot has rank i then

$$\text{size}(L1) = i-1$$

$$\text{size}(L2) = n-i$$

Homework: What is $E(T(n))$?