Chapter 2. Sorting & Order Statistics

We discuss:

- Sorting algor. (HeapSort, QuickSort)
- Median & Order Statistics
- A Lower Bound on Sorting.

Selection Sort.

We've discussed Insertion Sort: it's in-place and has time complexity $O(n^2)$.

Idea of Selection Sort:

- Remove largest element and add it to output sequence until input sequence is exhausted.

```
SELECTION (A[1..n])
    j = n
    while  j > 1
        k = j ; max = j ; S = A[j]
        while  k > 1
            k = k-1
```

if $A[k] > S$ **then**

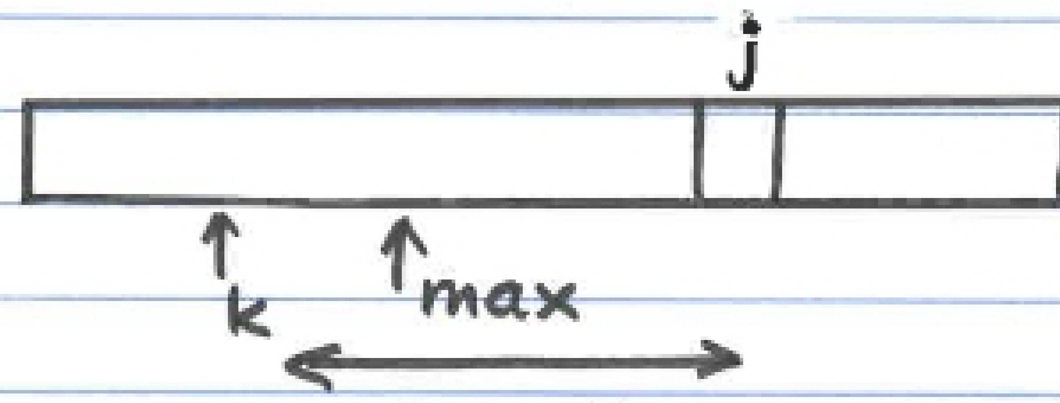
$max = k$

$S = A[k]$

swap $A[max]$ with $A[j]$

$j = j-1$

Illustration



$A[max] =$ max of $A[k+1 .. j]$

Complexity of SELECTION:

- Searching for max in $A[1..j]$ takes $O(j)$ steps
- Outer loop is executed for values $j = n, n-1, \dots, 2$.

Thus, $T(n) = O\left(\sum_{j=2}^{n} O(j)\right)$

$= O(n^2)$.

Heapsort

<u>Observation:</u> If we can find max faster, we can improve on SELECTION.

$\Rightarrow$ We organize $A[1..n]$ as a <u>heap</u>

A <u>heap</u> is an array object that can be identified as an almost complete binary tree: $\text{parent}(i) = \lfloor i/2 \rfloor$

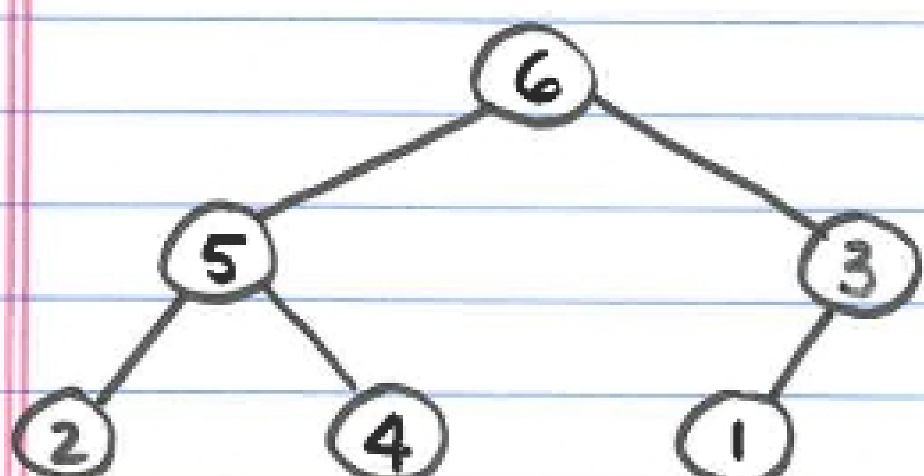
$\text{left}(i) = 2i$ and $\text{right}(i) = 2i+1$

A heap is a <u>max heap</u> if

$$A[\text{parent}(i)] \geq A[i]$$

(<u>min heap</u> if $A[\text{parent}(i)] \leq A[i]$)

<u>Example:</u> (max heap)



6	5	3	2	4	1