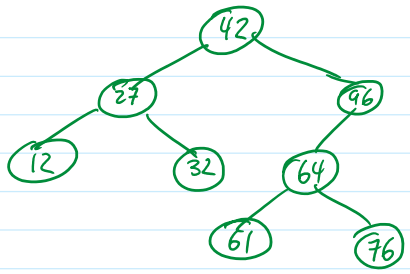
6.3 Applications of Decrease-by-a-Variable Factor

Tuesday, April 8, 2025 1:35 PM

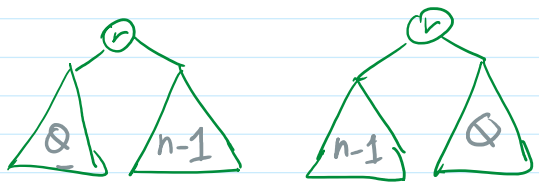
- Example: Insertion and Search in Binary Search Tree




```
Search (T, x):
    r ← root of T
    if x < r then search( left(T), x)
  elseif x > r then Search ( right(T), x)
  else
      x = r !  found!!
```

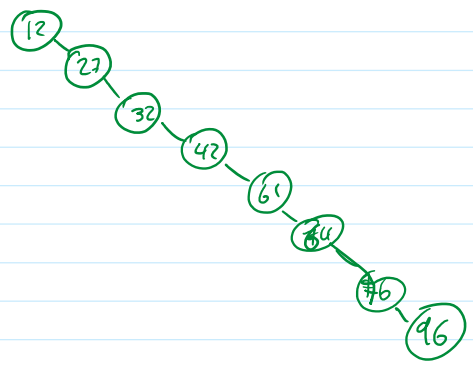
- The size of the problem decreases, but you do not know by how much.

Let $x < r$



best worst

degenerate tree



Analysis

basic operation < comparison

$C_{best}(n) = 1 \in \Theta(1)$

$C_{worst}(n) = n \in \Theta(n)$

Issue: Worst case analysis does not always gives you the whole picture.
It is better to do average or amortized analysis.

$C_{avg}(n) \in \Theta(\log n)$

T: set of all possible trees of n elements

$$\frac{C(n) \text{ for every } t \in T}{|T|}$$

$$\frac{C(n) \text{ for every } t \in T'}{|T|}$$

- Example: The Selection Problem. (The Median Problem)
  - Given an unsorted array $A[0 \ldots n-1]$ find the k'th largest element.
  - Special case : $k = \lfloor \frac{n}{2} \rfloor$ i.e. the median

idea #1

1:- Sort the array
2:- look at the $A[n-k-1]$ element
⊖ you are sorting the whole array.

idea #2

1:- use an incremental sort.
⊖ you still sorting ~half~ of the array

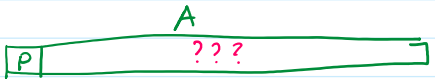
idea #3

1:- scan array
2:- maintain the k biggest elements you have seen so far.
⊖ if k is large, the table becomes difficult to maintain.

idea #4:

Use a concept called "partitioning"

Given:

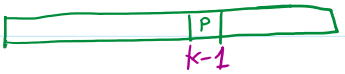


Partitioning by p gives you

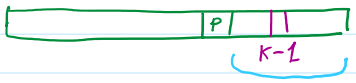
NOTE: partitioning is not sorting
- P is in its "sorted" place.

Algorithm: To find the K'th largest element.

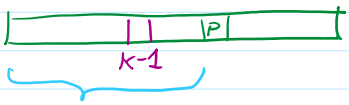
1) partition Array
2) - case:



Return P.

- case: 

partition right subarray

- case: 

partition left subarray

e.g Trace: [20, 31, 4, 25, 6, 3, 29, 7, 33] $n=9$, $k=3$

p

[4, 6, 3, 7, 20, 31, 25, 29, 33]

p

[~~~~~~~~~~~~, 25, 29, 31, 33]

p

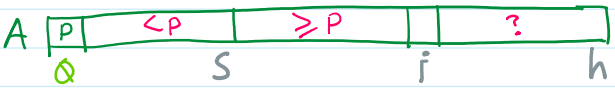
[~~~~~~~~~~~~ 25, 29 ~~~~~~~~~~~~]

[~~~~~~~~~~~~ 29, ~~~~~~~~~~~~]

How to partition? Lomuto Partitioning.

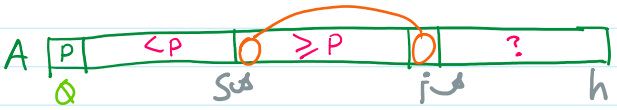
Intuition:

consider a partial partition

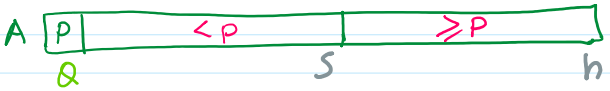


what to do with $A[i]$?

Cases
- $A[i] \geq p$ advance i
- $A[i] < p$
  - swap $A[i]$ with $A[s+1]$
  - advance s
  - advance i



Eventually:



then: swap $A[0]$ with $A[s]$

```
FUNCTION LomutoPartition (A[l..h])
    p ← A[l]
    s ← l
    FOR i ← l+1 TO h DO
        IF A[i] < p THEN
            swap(A[i], A[s+1])
            s ← s+1
    Swap(A[l], A[s])
```

```
    s ← s+1
    Swap(A[l], A[s])
    RETURN s
```

Let's use Lomuto Partition to find the k'th element.

```
FUNCTION   QuickSelect(A[l..h], k)
    s ← Lomuto Partition(A[l..h])
    IF s = k-1 THEN
        return A[s]
    ELSE
        IF s > l+k-1 THEN
            QuickSelect(A[l...s-1], k)
        ELSE
            QuickSelect(A[s+1, h], k)
```

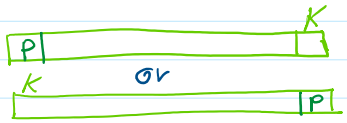
Analysis:

basic operation comparisons inside Lomuto Partition:

best:

$C_{best}(n) = n-1 \in \Theta(n)$ linear.

worst:



$$C_{worst}(n) = (n-1) + (n-2) + (n-3) + (n-4) + \dots + 1$$
$$= \frac{(n-1)n}{2} \approx \frac{1}{2}n^2 \in \Theta(n^2) \text{ quadratic.}$$

- More sophisticated analysis required:

$$C_{Avg}(n) \in \Theta(n)$$

Eof