06/03/19

## Binary Search:

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 3 | 6 | 8 | 12 | 14 | 17 | 25 | 29 | 31 | 36 | 42 | 47 | 53 | 55 | 62 |
| ↑ l | | | | | | | | | | | | | | ↑ h |

Mid = $\frac{l+h}{2}$

Two pointers – (l, h).

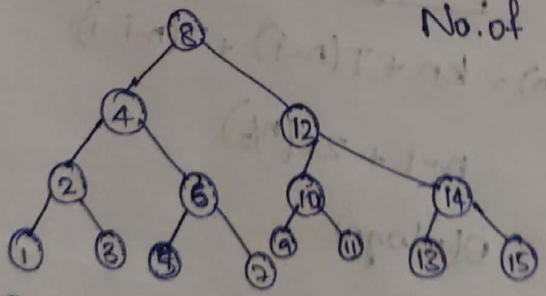
→ Compare with mid and proceed.

→ If h<l - element is not present in the list.

Algorithm: BinarySearch (a[], n, key){

```
l=1; h=n;
while (l<=h){
    mid = (l+h)/2;
    if (key <= A[mid]) return A[mid];
    if (key <A[mid]) h= mid-1;
    else l= mid+1;
}
}
```

Height = $\log n$



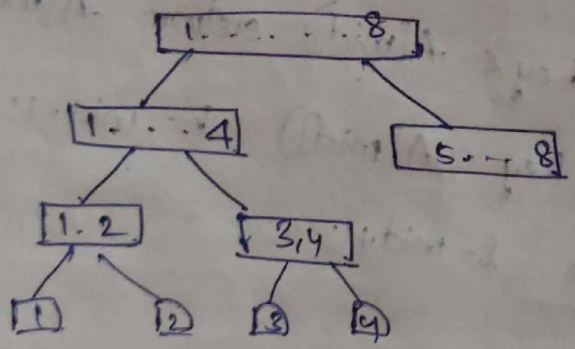

No. of comparisons = $\log_2(n)$

= 1 (min)

```
Binary Search (a[], l, h, n, key){
    mid = (l+h)/2;
    if (key == A[mid]) return A[mid];
    else if (key <A[mid]) BinarySearch (a, l, mid-1, n, key);
    else BinarySearch (a, mid+1, h, n, key);
}
```

## MergeSort:

```
Merge(A,B,m,n){
    int i=1,k=1;
    while(i<m && j<=m){
        if (A[i] < B[j])
            C[k++]=A[i++];
        else
            C[k++] = B[j++]);
    }
    for(; i<m; i++)  C[k++] = A[i]
    for(; j<n; j++)  C[k++] = B[j];
}
```




nlogn.

## QuickSort:

$T(n) = kn + T(n-i) + T(n-i-1)$

$= n-1 + 2T(n/2)$

$\theta(n\log n)$

## Median and order statistics:

To find min/max — (n-1) comparisions required.

using partitioning algorithm (divide and conquer) — n/2 steps.

49 10 59 92 6 70 9 31

```
DACMM (A, i, j, max, min) {
  if (i==j) return max=min=A[i];
  else if (i==j-1){
      if (A[i]<A[j]) {
          max = A[j];
          min = A[i];
      else
          max=A[i];
          min=A[j];
      }
  else{
      mid=(i+j)/2;
      DACMM (A, i, mid, max1, min1);
      DACMM (A, mid+1, j, max1, min1);
      if (max1 > max) max=max1;
      if (min1 < min) min=min1;
  }
```

$\frac{3n}{2}$ comparisions

( , ) ( , ) ( , )

find min and max. in every pair and

compare 'min' and 'max' with actual 'min' and 'max'.

Find $K^{th}$ smallest element.

Randomized Select Algorithm:

```
RandomizedSelect (A, p, q, i) {
    if (p == q) then return A[p];
    r = RandomizedPartition(A, p, q)
    k = r - p + 1
    if (i == k) then return A[r];
    if (i < k) then return
        RandomizedSelect (A, p, r-1, i)
    else
        return RandomizedSelect (A, r+1, q, i-k)
}
```

Partitioned element - element in correct position.

$O(n)$

worst case: to calculate partition element $\Rightarrow O(n^2)$

Selection in Worst case Algorithm:

Elements are divided into $(\frac{n}{5})$ groups, each group contains

① 5 elements and 1 group contains in mod5 groups.

n/5 medians are obtained (1 from each)

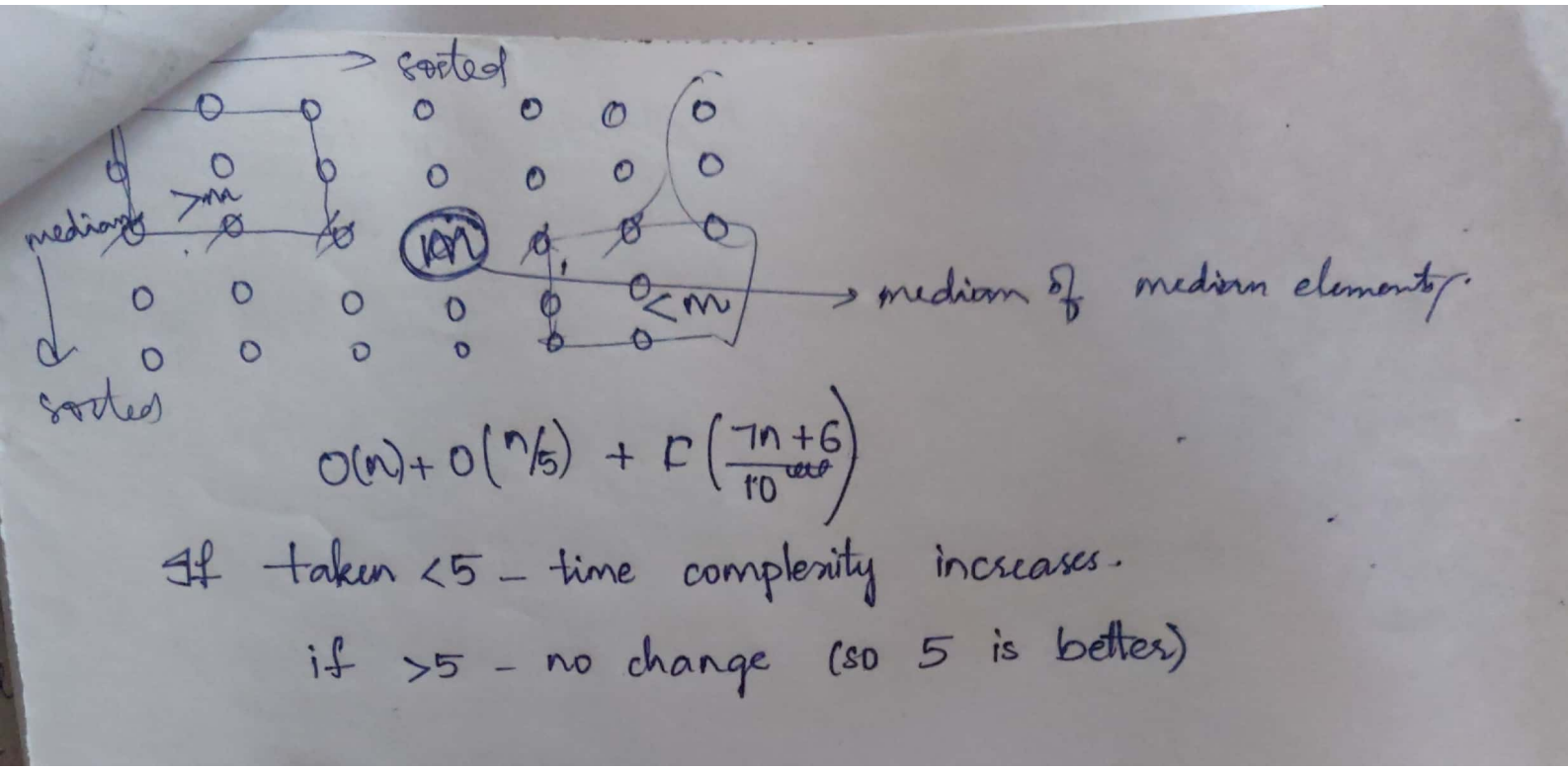
② Sorting for each group and find median.

$O(n)$

③ Median of $(\frac{n}{5})$ median elements.

$O(\frac{n}{5})$

④




$$O(n) + O(n/5) + T\left(\frac{7n+6}{10}\right)$$

If taken < 5 – time complexity increases.

if > 5 – no change (so 5 is better)

03/04/19

## Karatsuba Multiplication: (Divide and Conquer)

Divide the numbers into 2 parts. (n/2 each)

$X = \overset{a}{12}|\overset{b}{34} = 12 \times 10^{n/2} + 34 = a \times 10^{n/2} + b = 12 \times 10^2 + 34$

$Y = 43|21 = 43 \times 10^{n/2} + 21 = c \times 10^{n/2} + d$

$a = 12;\ b = 34$

$c = 43;\ d = 21$

$X * Y = (a \times 10^{n/2} + b)(c \times 10^{n/2} + d)$

$= ac\,10^n + ad\,10^{n/2} + bc\,10^{n/2} + bd$

$= (ad + bc)\,10^{n/2} + ac\,10^n + bd$

No. of multiplications = 4

It is recursive algorithm.

(further divisions are possible)

a - again into 2 parts and combining.

We can do as:

$X = ac;\ Y = bd$

$Z = (a+b)(c+d) - X - Y$

$\Rightarrow$ 3 multiplications.

Answer is $\boxed{X \times 10^n + Z \times 10^{n/2} + Y}$

Time complexity $= 3T(n/2) + O(n)$

$X = (12)(43) = 516$

$Y = (34)(21) = 714$

$Z = (a+b)(c+d) - X - Y$

$= (12+34)(43+21) - (12)(43) - (43)(21)$

Again call recursively.

X, Y is SP of 2

$a_1 = 1;\ b_1 = 2;\ c_1 = 4;\ d_1 = 3$

$x_1 = a_1 \times c_1 = 4$

$y_1 = 2 \times 3 = 6$

$z_1 = (3)(7) - (4) - 6 = 21 - 10 = 11$

Ans: $4 \times 10^2 + 11 \times 10^1 + 6 = 400 + 110 + 6 = 516$

Y = (12)

$a_1 = 3$; $b_1 = 4$; $c_1 = 1$; $d_1 = 2$

$x_1 = 3$; $y_1 = 4$

$z_1 = (3)(4) - 3 - 4 = 24 - 7 = 17$

Ans = $14 \times 10^2 + 4 \times 10^1 + 6 = 1446$

1400
450
6

$z = (55)(53) - 516 - 446$

↓ again rec!

$a_1 = 5$; $b_1 = 5$; $c_1 = 6$; $d_1 = 3$

$x_1 = 30$; $y_1 = 15$

$z = (10)(9) - 30 - 15 = 45$

$z = 30 \times 10^2 + 45 \times 10^1 + 15$

$= 3000 + 450 + 15$

$= 3465$

2000
450
15
3468

516
446
962

$z$ [Ans] $= 3465 - 962 = 2503$

2 14
3465
962
2503

Final Ans: $516 \times 10^4 + 2503 \times 10^2 + 446$

$= 5160000$
$250300$
$446$

Y: (34)(21)

$a_2 = 3$; $b_2 = 4$; $c_2 = 2$; $d_2 = 1$

$\Rightarrow x_2 = 6$; $y_2 = 4$

$z_2 = (7)(3) - 6 - 4 = 21 - 10 = 11$

Ans: $6 \times 10^2 + 11 \times 10^1 + 4 = 600 + 110 + 4 = 714$

$z = (46)(64) - 516 - 714$

2944

$a_3 = 4$; $b_3 = 6$; $c_3 = 6$; $d_2 = 4$

$\Rightarrow x_3 = 24$; $y_3 = 24$

$z_3 = (10)(10) - 24 - 24 = 100 - 48 = 52$

Ans: $2400 + 520 + 24 = 2944$

$2944 - 516 - 714$

$= 1714$

Ans: $516 \times 10^4 + 1714 \times 10^2 + 714$

$= 5160000$
$171400$
$+ \quad 1714$
$5332114$ ✓

Algorithm:

KaratSubu(x, y)

if n=1 use simple multiplication to find

T = x * y;

else split x and y into half $(X_H, X_L)$ $(Y_H, Y_L)$ such that

$X = X_H \times 10^{n/2} + X_L$ and

$Y = Y_H \times 10^{n/2} + Y_L$

// all $X_H, X_L, Y_H, Y_L$ having n/2 digits

A = karatsubu $(X_H, Y_H)$

B = karatsubu $(X_L, Y_L)$

E = karatsubu $(X_H + X_L, Y_H + Y_L)$

F = E - A - B

$T(n) = A \times 10^n + 10^{n/2} + B$

Selection worst case alg:

$T(n) = O(n) + O\left[\frac{n}{5}\right] + O\left(\frac{7n}{10} + 6\right)$

If 5 elements,

$T(n) = T\left(\frac{n}{5}\right) 3\left(\frac{1}{2}\left[\frac{n}{5}\right] - 2\right)$

If 3 elements taken,

T(n) becomes O(nlogn).

## ctivity Selection problem:

→ One of the greedy methods.

→ Greedy method doesn't always give optimal solution.

S - $\{a_1, a_2 \ldots a_n\}$ - set of 'n' activites

Each activity has starting and finishing time.

→ To accomodate more no.of activities, activity duration should be less.

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Activity: | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| time: Starting: | 1 | 3 | 0 | 5 | 3 | 5 | 6 | 8 | 8 | 2 | 12 |
| time: Finish | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 17 | 14 |

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| $S_i$ | 2 | 2 | 4 | 1 | 5 | 8 | 9 | 11 | 13 |
| $F_i$ | 3 | 5 | 7 | 8 | 9 | 10 | 11 | 14 | 16 |
| Duration | 1 | 3 | 3 | 7 | 4 | 2 | 2 | 3 | 3 |

→ Choose max. no.of activities without overlap. (mutually compatible)

Activities chosen - $\{a_1, a_3, a_6, a_8\}$ - max. size

→ It is not always successful.

(Here solved based on minimum duration).

With another method, more optimal may be obtained.

Minimum finishing time:

Consider $P_1$,

based on min. finishing time.

$\{a_1, a_4, a_8, a_{11}\}$

1-4 5-7 8-11 12-14

If Based on min. duration, min. starting point - overlapping may occur.

But based on min. finishing time, no problem.

greedy activity selection (s, f)

sort elements based on finishing time (asc. order)

n = s.length

A = {a}

K = 1

for m = 2 to n

if s[m] ≥ F[k]

A = A ∪ {$a_m$}

K = m

return A;

Time complexity = O(nlogn) + O(n)

→ Compare the greedy soln. with optimal soln.

If not same change optimal (∵ more optimals are possible)

Starting Time of 2nd acti. ⊜ Finishing of 1st act.