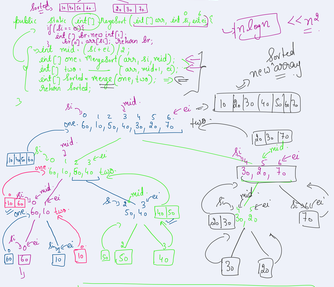
Divide & Conquer

Sorting Algo – Bubble Sort
→ Insertion Sort
→ Selection Sort

Mergesort
Quicksort

• Mergesort:
Step 1
↳ Sorted Array



$$T(N) = T(N/2) + T(N/2) + N.$$

$$T(N) = 2T(N/2) + N.$$

$$T(N/2) = 2T(N/4) + N$$

$$2T(N/4) = 2^2T(N/8) + N$$

$$2^kT(1) = N$$

N
N/2
N/4
N/8
1

adding both lhs & rhs

$$N/2^k = 1$$

$$\frac{N}{2^k} = 1$$

$$\frac{N}{2^k} = 1$$

$$N = 2^k.$$

$$\log_2 n = \log_2 2^k$$

$$\log_2 n = K$$

$$T(N) = K.N.$$

$$n \log_2 n$$

• Binary Search.

```
if (arr[mid] == key){
    return mid;
}
arr[mid] < key   ⇒ 25 < 25   ↳ si = mid+1;
arr[mid] > key   ⇒ 40 > 25   ↳ ei = mid-1;
return -1;
```

$$T(N) = T(N/2) + c$$
$$T(N/2) = T(N/4) + c$$
$$T(N/4) = T(N/8) + c$$
$$T(1) = c$$

N
N/2
N/2
N/4

$$T(N) = kc$$
$$= c \log_2 n$$
$$O(\log_2 n)$$

$$N, \frac{N}{2}, \frac{N}{2^2}, \frac{N}{2^3} \ldots \frac{N}{2^k} = 1$$

$$\frac{N}{2^k} = 1$$
$$N = 2^k$$
$$K = \log_2 N$$

$$si + (ei - si)/2$$

$$(10^8 + 10^9)/2$$

si = 0

ei = 10.

$$0 + (10 - 0)/2$$
↳ halfed

$$0 + 10^8 + (10^9 - 10^8)/2$$

Q. → { 10, 20, 30, 30, 30, 30, 30, 30, 40, 50, 60 }

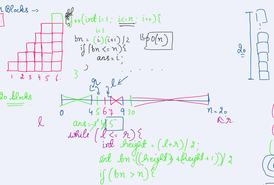
first index → 30

ans = -1 ✗ ?

↳ O(log n)

Monotonic function

Q. n Blocks → n = 20




```
for (int i = 1; i <= n; i++){
    bn = (i)(i+1)/2   // O(n)
    if (bn <= n){
        ans = i;
    }
}
```

n → 20 blocks

```
ans = -1;
while (l <= r){
    int height = (l+r)/2;
    int bn = ((height)(height+1))/2
    if (bn > n){
        r = height - 1;
    } else {
        ans = height;
        l = height + 1;
    }
}
return ans;
```

$(16 \times 17)/2 = \sqrt{55}$
$(4 \times 5)/2 = 10$
$(7 \times 8)/2 = 28$
$(5 \times 6)/2 = 15$
$(6 \times 7)/2 = 21$

↳ O(log n)

Resume at 8 Sopra

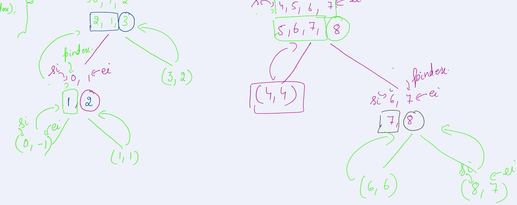
• Quickick

```
public static int partition(int[] arr, int si, int ei){
    int pivot = arr[ei];
    int pindex = si;
    for (int i = si; i <= ei-1; i++){
        if (arr[i] <= pivot){
            swap(arr, i, pindex);
            pindex++;
        }
    }
    swap(arr, pindex, ei);
    return pindex;
}
```

```
public static void qs (int[] arr, int si, int ei){
    if (si >= ei){
        return;
    }
    int pindex = partition(arr, si, ei);
    qs(arr, si, pindex-1);
    qs(arr, pindex+1, ei);
```

arr. { 8, 1, 3, 4, 8, 5, 7, 6 }

⇒ { 1, 2, 3, 4, 5, 6, 7, 8 }



↳ 0075 Ishdeep Singh Khurana ↳ Lecture 15

Recursion, subtend code, lecture notes.