K_3 (r_{23})

→ **Randomized Algorithms**

- Las Vegas → Randomized Quick Sort
- Monte Carlo → Min. count

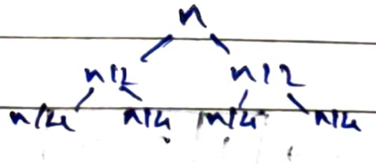
Amortized Analysis

- Aggregate Analysis
- Accounting Analysis
- Potential Analysis

→ **Quick Sort**

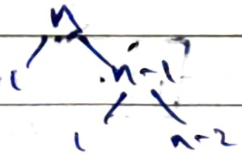
```
Quick Sort (int l, int r) {
    if (l < r) {
        p = partition (l, r);
        QuickSort (l, p-1);
        QuickSort (p+1, r);
    }
}
```

- **Best case**: The pivot is always in the middle,



depth = log n
Each level = n
⇓
$O(n \log n)$

- **Worst case**: If already sorted. (choosing A[0] in ascending)



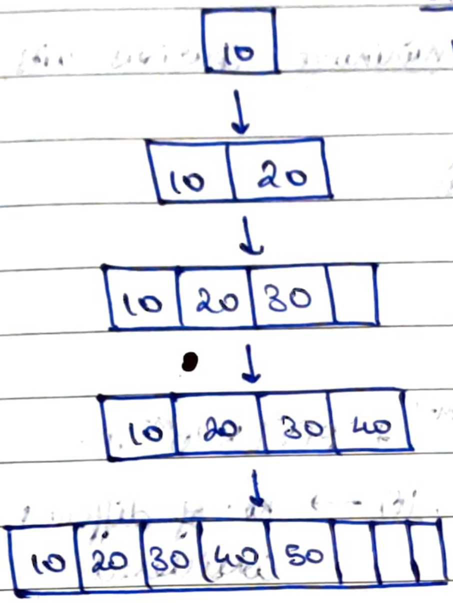
Depth = n
Each level = n
⇓
$O(n^2)$

- **Randomized Pivot** → Never worst case
  ⇒ Always $O(n \log n)$

```
Partition (int l, int r) {
    pivot = A[l];
    i = l; j = r;
    while (i < j) {
        do {
            i++;
        } while (A[i] <= pivot);
        do {
            j--;
        } while (A[i] > pivot);
        if (i < j)
            swap (A[i], A[j]);
    }
    swap (A[l], A[j]);
    return j;
}
```

→ Dynamic Arrays

- We keep doubling size of array to accomodate more elements,



Size	Cost
1	1
2	1+1
4	2+1
4	1
8	4+1

* Asymptotically, Insertion takes O(n) time $\overset{\text{for each element}}{\uparrow}$ as every few operations we need to copy the array.

However, realistically there are more elements taking O(1) time than taking O(n) time. Thus, we need to take average.

* For 10 insertions here, the cost is

$$\frac{1+2+3+1+5+1+1+1+9+1}{10} = \frac{25}{10} \rightarrow \underline{2.5} \quad (\text{asymptotic would be } \underline{10})$$

↓

Amortized Analysis (Aggregate)

□ Generally, we have

$$\frac{\overbrace{(1+1+1+\cdots)}^{n \text{ times}} + \overbrace{(1+2^1+2^2+\cdots)}^{\log n \text{ times}}}{n} = \frac{n+(2n-1)}{n} = \underline{O(1)}$$

→ Randomized Algos

- Uses random numbers (or) choices to decide the next step (or) logic
- Possible to reduce time and space complexity,

```
Las Vegas (A, n, x) {
    while (true) {
    randomly select an element out of 'n' elements
    if (x is found)
        return true
    }
}
```

* Iterations/runtim are varied and can be arbitraril large.

* Always gives correct answer.

```
Monte Carlo (A, n, x) {
    i=0, flag = false
    while (i <= n) {
        randomly select an element out of 'n' elements
        i++
        if (x is found)
            flag = true
    }
    return flag
}
```

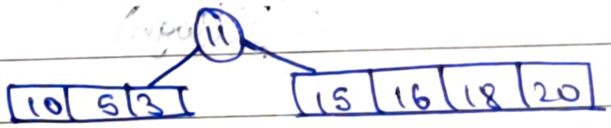
* Iterations / runtime is fixed.

* May or may not give correct answer.

→ **Randomized Quick Sort**

Ex: 10 15 5 3 11 16 18 20

If we choose 11 as pivot,



Here, ③ is only compared with ⑩ and ⑤ not everything else.

Time complexity Analysis

$X_{ij} = I$ → (random indicator variable)

↳ either 0 (or) 1

* If Z_i is compared to Z_j, then $X_{ij}=1$

If they are not compared, then $X_{ij}=0$

{Z_i, Z_j are simply two numbers}

* For example, in selection sort, we have $\sum_{i=1}^{n}\sum_{j=i+1}^{n} X_{ij}$ (as time complexity) {as, we have to find min. of (n), then (n-1) .....}

Here, for randomized quick sort, we need to find the expected time complexity

$$\Rightarrow E[x] = E\left[\sum_{i=1}^{n}\sum_{j=i+1}^{n} X_{ij}\right] = \sum_{i=1}^{n}\sum_{j=i+1}^{n} E[X_{ij}]$$

$$= \sum_{i=1}^{n}\sum_{j=i+1}^{n} P_{ij}$$

{P_{ij} = Probability of comparing i, j}

We have,

$$\underbrace{\cdots\cdots}_{\substack{(i-1)\text{ elements}\\ \text{I}}}\ \overbrace{Z_i \cdots\cdots Z_j}^{\substack{(j-i+1)\text{ elements}\\ \text{II}}}\ \underbrace{\cdots\cdots}_{\substack{(n-j)\text{ elements}\\ \text{III}}}$$

* If we choose pivot from section II, (z_i, z_j) will not be compared unless either (z_i) or (z_j) is chosen as pivot.

→ Probability = $\frac{2}{j-i+1}$ { not taking 2/n as we are taking Σ for all (i,j)

* From section I, III we will not have other suitable pivots

$$\Rightarrow \sum_{i=1}^{n} \sum_{j=i+1}^{n} (2/j-i+1)$$

If we let $k = (j-i)$,

$$\sum_{i=1}^{n} \sum_{k=1}^{n-i} (2/k+1)$$

But, we know

$$\sum_{i=1}^{n} \sum_{k=1}^{n-i} (2/k+1) < \sum_{i=1}^{n} \sum_{k=1}^{n} (2/k)$$

$$< \sum_{i=1}^{n} (\log n)$$

$$< \underline{O(n \log n)}$$