Searching :

Agenda :

- Sets
- DSA
- Linear search / Sequential / serial
- Binary search

Sets :

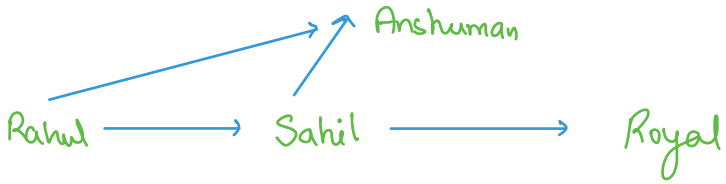
- Set is a kind of Data Structure
- Set is unordered in nature
- Set only have unique values
- Used when we want to avoid duplicacy.

---

* DSA : Data Structure & Algorithm

Data Structures

- List
- Set
- tuple
- Dict
- Str
- Trees
- Stack & Queue
- Linked list
- "Graph"



1st degree connectn : Sahil, Anshuman
2nd degree : Royal

Linkedin, facebook, Google maps

DS : Storing or maintaining the data

Algorithm : Set of instruction in specific order

DSA : To make our code efficiend & more optimised.

---

* **Searching:** Linear & Binary

Where ? Who ?

Where : Search space
Who : Target

Linear Search :

→ patient_id = [4, 8, 2, 1, 7, 9, 3]
(indices: 0 1 2 3 4 5 6)

→ target = 7

Summary :

```
def linear_search(ss, target):
    # iterate on ss:
        if target == iterator:
            return i

    # return "NA found"
```

Minimum : 1 step

Maximum : length of list (n)

---

* **Binary Search :**

Bi : means 2

Conditions : Works only on "Sorted / monotonic" space
Monotonic : Either Increasing or Decreasing order

Dictionary :

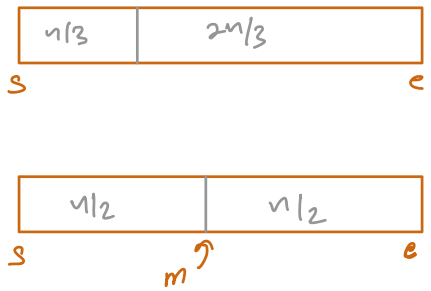
d = [A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, ... Y, Z]

s e

Target : "Dog"

We always look forward to discard a part of search space.

★ Best flip case :

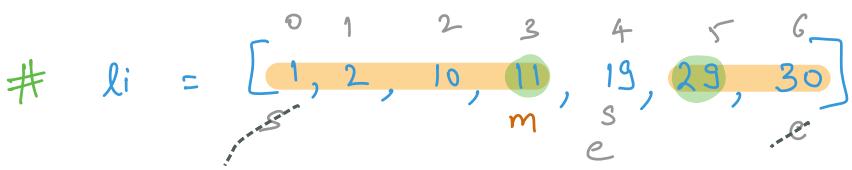


n = len of Search space

We always try to discard half of the search space.

Problem : You are given a sorted array find the index of target element.

target = 19



mid = (s+e) // 2

mid = 3 , 5 , 4

i) Check if mid == target.

ii) Compare target with mid.

target > mid : # discard left

start = mid + 1

iii) target < mid # discard right

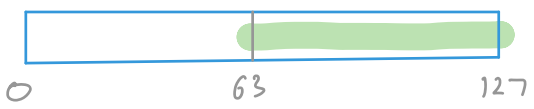
end = mid - 1

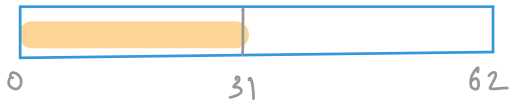
Summary :

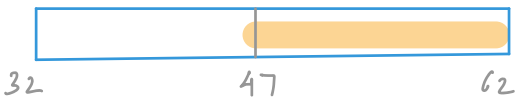
```
s = 0
e = len - 1

while s <= e :
    Find mid
    → if target == mid :
          return mid
    → target > mid      # discard left
                        # s = mid + 1
    → target < mid      # discard right
                        # e = mid - 1
return "Not found"
```

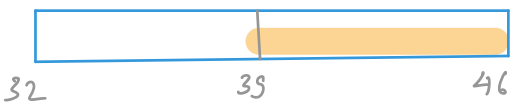
128 : 128 times → Linear Search

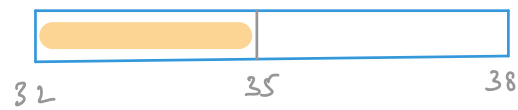
Binary Search

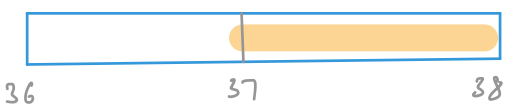


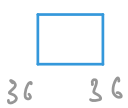

0
31
62


32
47
62


32
39
46


32
35
38


36
37
38


36
36