UNIT II – Brute Force and Divide and Conquer

- **Brute Force Design Technique**
  - Selection Sort
  - Bubble Sort
  - ***Sequential Search***
  - Closest pair and Convex hull problem
  - Travelling Salesman problem
  - Knapsack problem
  - Assignment problem

Sequential Search – Traditional method

- Worst case O(n) – element not found/ search element is in last position of list
- Best case O(1) – element found at 1st position
- Average case – element found at mid position of the list

```
#include<stdio.h>
void main()
{
    int a[100],n,i;
    printf("\n enter the array elements");
    scanf("%d",&n);
    for(i=0;i<n;i++)
    {
        scanf("%d",&a[i]);
    }
    printf("\n enter the element to search");
    scanf("%d",&n);
    printf("\n searching");
    for(i=0;i<n;i++)
    {
        if(a[i]==n)
        {
            printf("\n Element found %d at position %d",a[i],i+1);
            exit(0);
        }

    }
}
```

Sequential Search

- Extra trick in implementing sequential search – append the search element to the last position in the list

55	60	70	32	23	89	32
A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	Search key A[n]

ALGORITHM *SequentialSearch2*$(A[0..n], K)$

//Implements sequential search with a search key as a sentinel
//Input: An array A of n elements and a search key K
//Output: The index of the first element in $A[0..n-1]$ whose value is
// equal to K or -1 if no such element is found

$A[n] \leftarrow K$
$i \leftarrow 0$
while $A[i] \neq K$ **do**
  $i \leftarrow i + 1$
if $i < n$ **return** i
else return -1

UNIT II – Brute Force and Divide and Conquer

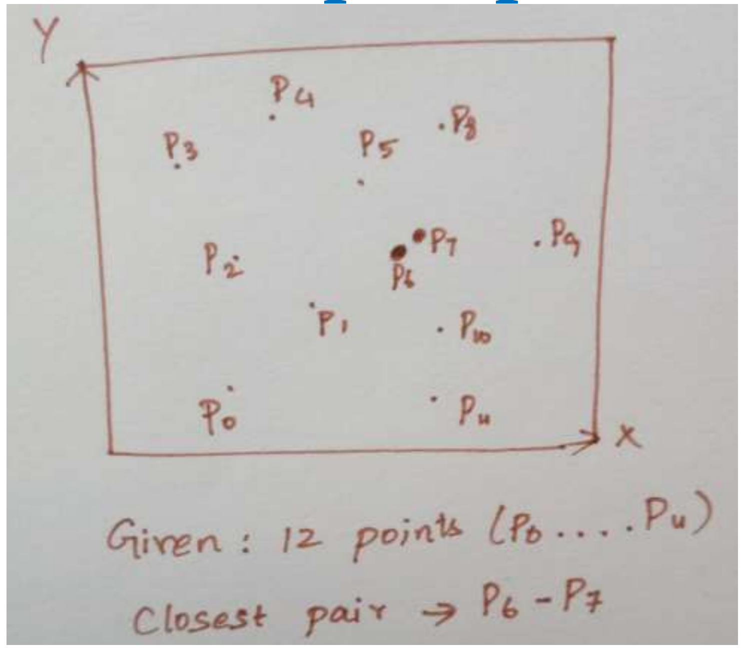
- **Brute Force Design Technique**
  - Selection Sort
  - Bubble Sort
  - Sequential Search
  - **Closest pair and Convex hull problem**
  - Travelling Salesman problem
  - Knapsack problem
  - Assignment problem

Closest pair problem

- Geometric problem
- Straight forward approach - Finite set of points in the plane
- Applications : computational geometry and operations research
- Google map- nearby restaurants
- *Problem statement: find the two closest points in a set of points*
- Solution:
- Assumption:
  - 2-dimensional space
  - (x,y) Cartesian coordinates
  - Distance between 2 points $P_i=(x_i,y_i)$, $P_j=(x_j,y_j)$ Euclidean distance

$$d(p_i, p_j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}.$$

Closest pair problem



ALGORITHM *BruteForceClosestPair(P)*

//Finds distance between two closest points in the plane by brute force
//Input: A list P of n ($n \geq 2$) points $p_1(x_1, y_1), \ldots, p_n(x_n, y_n)$
//Output: The distance between the closest pair of points

$d \leftarrow \infty$
for $i \leftarrow 1$ **to** $n - 1$ **do**
  for $j \leftarrow i + 1$ **to** n **do**
    $d \leftarrow \min(d, sqrt((x_i - x_j)^2 + (y_i - y_j)^2))$ //*sqrt* is square root
return d

Analysis of Closest-pair problem

1. Problem size : n
2. Basic operation : Euclidean Distance
3. Count of basic operation------------→
4. Efficiency – worst case

Closest pair problem – Count of basic operation

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

$$= 2 \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} 1$$

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

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

$$= 2\left[n\left(\sum_{i=1}^{n-1} 1\right) - \left(\sum_{i=1}^{n-1} i\right)\right]$$

S_1 ⇓, S_2 ⇓

$S_2 \rightarrow \frac{n(n+1)}{2}$

Here $n = n-1$

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

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

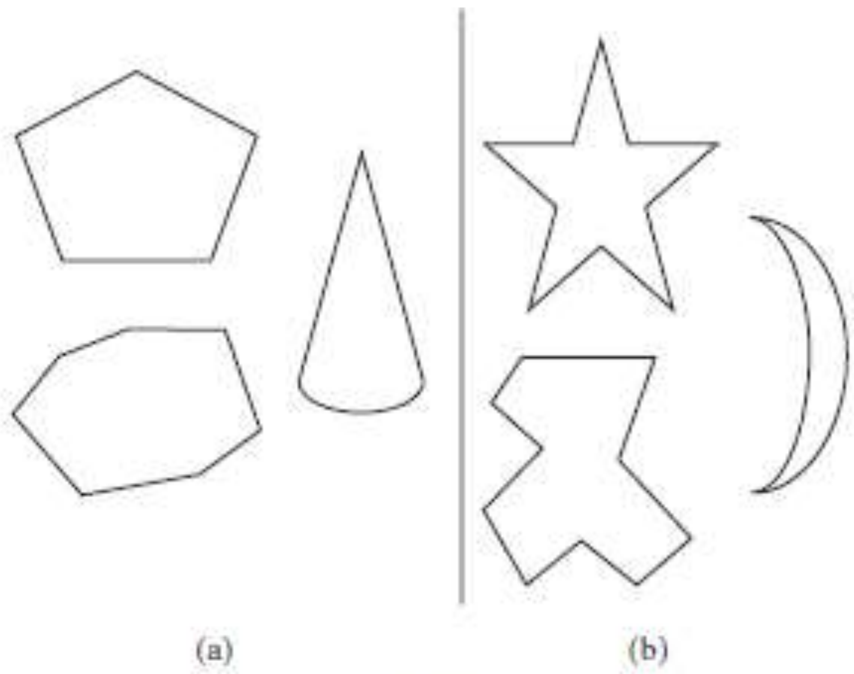
$$= 2\left[\frac{2[n(n-1)] - (n^2-n)}{2}\right]$$

$$= 2(n^2-n) - n^2 + n$$

$$= 2n^2 - 2n - n^2 + n$$

$$= n^2 - n = (n-1)n \in O(n^2)$$

Convex Hull

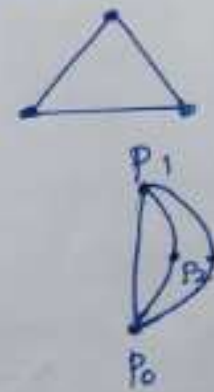


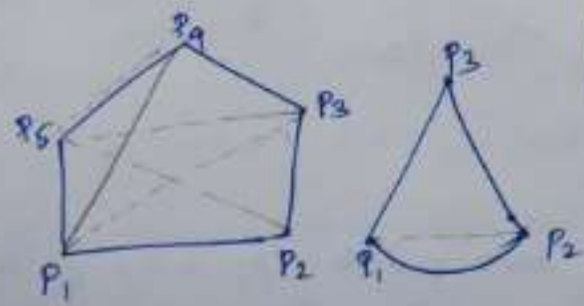
(a) Convex sets. (b) Sets that are not convex.

Convex Hull

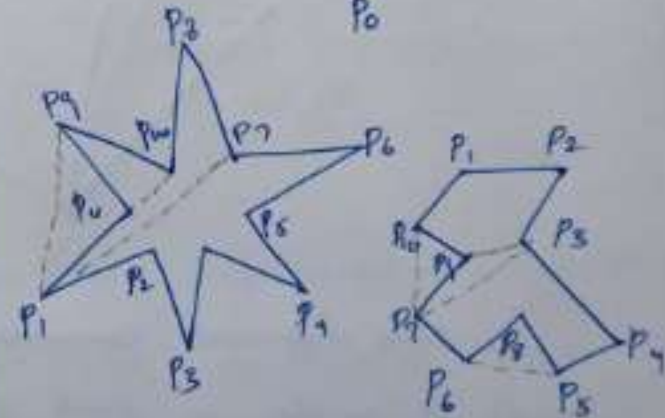
polygon (n>2)

- Geometric problem, Aircraft.
- Convex → shapes that curve outward
- Convex set
- Convex polygon





Convex sets



Not Convex (curve inward)

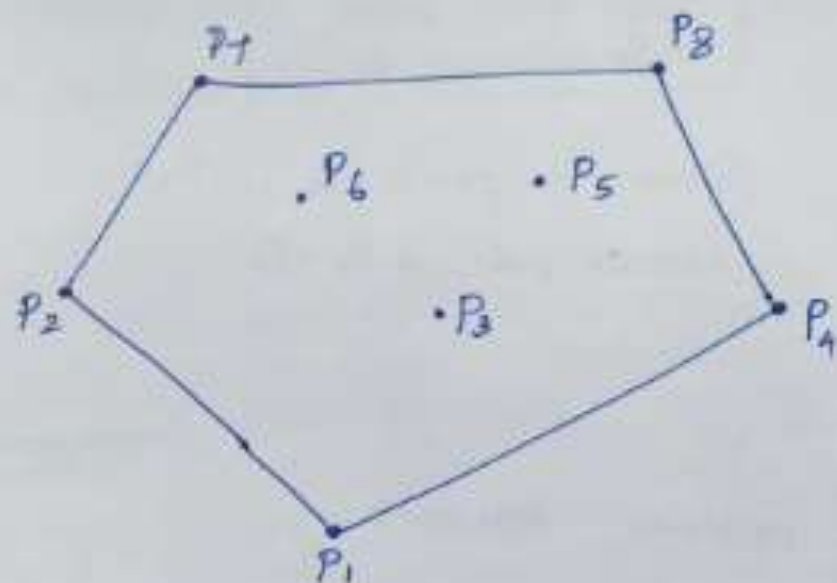
Convex Set

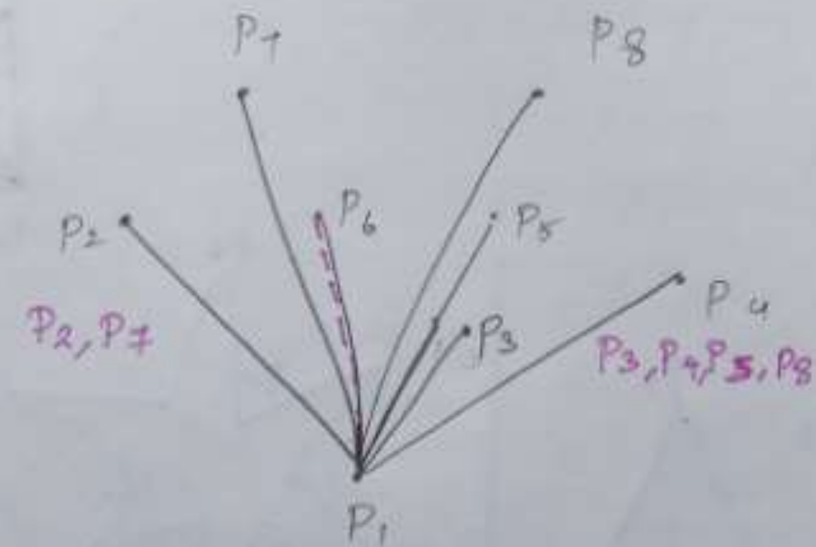
Set of points in the plane is called convex, if for any two points P & Q in set, the entire line segment with the endpoints at P & Q belongs to the set.

Convex hull of Set S of points is the Smallest Convex Set containing S.

* Convex polygon → Vertices. → extreme points

Should not be a middle point of any line segment





Which pair of points need to be connected to form the boundary of convex hull

a line segment connecting two points P_i & P_j of a set of n points is part of convex hull boundary, if and only if all other points of the set lie on the same side of the straight line through these points.

Straight line - 2 points (x_1, y_1) (x_2, y_2)

$$ax + by = c$$

Here $a = y_2 - y_1$

$b = x_1 - x_2$

$c = x_1 y_2 - y_1 x_2$

all points above the line → $ax + by > c$

all points below the line → $ax + by < c$

→ (P_1, P_2) forms boun

Algorithm

for each point P_i (P_1)

for each point P_j (P_2) where $P_j \neq P_i$

line segment (P_i, P_j)

for all other points P_k $(P_k \neq P_i \& P_j)$ $(P_3, P_4, P_5, \ldots P_8)$

if each P_k is on one side of line segment,

P_i, P_j ← Convex hull boundary.

P_1, P_2 (boundary of convex hull)

Convex Hull - Analysis

- Input size – n (set of points)
- Basic operation
- Count of basic operation – $O(n^3)$
- Worst case