WEEK 1 : GCD,GCD(NAIVE),GCD(EUCLID'S ALGO)

\*

WEEK 3:

Lecture 4 :

ARRAYS VS LISTS, BINARY SEARCH

1 . BINARY SEARCH :

Code :

def bs(li,x,f,l):

if (l-f == 0):

return False

else:

mid = ((l+f) // 2)

print("list : ",li)

print("mid : ",mid)

print("li[mid] : ",li[mid])

if li[mid] == x:

return True

elif(x < li[mid]):

return (bs(li,x,f,mid))

elif(x > li[mid]):

return (bs(li,x,mid+1,l))

print(bs(list(range(1,10)),5,0,10))

Time Complexity :

T(n) : T(0) = 1,

T(n) = 1 + T(n/2)= 1 + 1 + T(n/2^2) = ...

= 1 + 1 + ... + 1 + T(n/2^k) = 1+1+ ... + T(n/2^logn)

T(n) = O(logn)

Binary Search only works for arrays, not for list.

Lecture 5 : Efficiency

Different inputs take diff. time.

We will consider Worst case(value is not found).

O() notation : linear scan-O(n) for arrays and lists, Binary Scan-O(log n) for sorted arrays

T(n) = O(nk) is efficient.

T(n) = O(n2) has very limited range.

Lecture 6: Selection Sort

Code :

def ss(l):  
 for i in range(len(l)):  
  
 min = i  
 for j in range(i+1,len(l)):  
 if l[min] > l[j]:  
 min = j  
  
 l[i], l[min] = l[min], l[i]  
 print("Sorted array : ")  
 for i in range(len(l)):  
 print(l[i],end = " ")  
   
l = [74,32,89,55,21,64]  
print(ss(l))

T(n) = n + (n-1) + (n-2) + … + 1 = n(n+1)/2 = O(n2)

Lecture 7 : insertion sort