**PICT PRACTICAL EXAMINATION 2021**

**SUBJECT-DSL LAB**

**NAME-TANMAY KARMARKAR**

**ROLL NUMBER-21143**

**PERFORMANCE DATE-28/12/2021**

**CODE:**

*"""  
Write python program for linear search sentinel search binary search fibonacci search  
  
"""*class Searching:  
 def \_\_init\_\_(self):  
 self.array=[]  
 self.array\_sorted=[]  
  
  
 def create(self):  
 go=True  
 while go:  
 try:  
 n=int(input("Enter number of students present in class"))  
 go=False  
 except ValueError:  
 print("Enter integer only")  
 i=1  
 while i<=n:  
 try:  
 x=int(input("Enter Roll number "))  
 if x not in self.array:  
 self.array.append(x)  
 self.array\_sorted.append(x)  
 i+=1  
 else:  
 print("Roll number is already present")  
 except ValueError:  
 print("Enter integer values only")  
  
 """Creating second list for searching using binary and fibonacci search. They require a sorted list. SOrted using  
 insertion sort."""  
 for i in range(1,n):  
 curr=self.array\_sorted[i]  
 j=i-1  
 while j>=0 and self.array\_sorted[j]>curr:  
 self.array\_sorted[j+1]=self.array\_sorted[j]  
 j-=1  
 self.array\_sorted[j+1]=curr  
  
  
  
 def Linear\_Search(self):  
 *"""Best Case Time Complexity is o(1).Worst case Time complexity is O(n)"""* go = True  
 while go:  
 try:  
 x = int(input("Enter roll number which has to be searched"))  
 go = False  
 except ValueError:  
 print("Enter integer only")  
  
 for i in range(0,len(self.array)):  
 if self.array[i]==x:  
 return i  
 return -1  
  
  
 def Sentinel\_search(self):  
 *"""Time complexity of O(n).Difference between linear search and sentinel search is that the co  
 comparisons are less in sentinel search ."""* go = True  
 while go:  
 try:  
 x = int(input("Enter roll number which has to be searched"))  
 go = False  
 except ValueError:  
 print("Enter integer only")  
  
 last=self.array[len(self.array)-1]  
 self.array[len(self.array)-1]=x  
 i=0  
 while self.array[i]!=x:  
 i+=1  
  
 self.array[len(self.array)-1]=last  
  
 if (i<len(self.array)-1) or (self.array[len(self.array)-1]==x):  
 return i  
 else:  
 return -1  
  
 def Binary\_seach(self,low,high,search):  
 *"""Works on sorted array.Divide and Conquer Technique. Time complexity of O(logn)"""* if high>=low:  
 mid =(high+low)//2  
 if self.array\_sorted[mid]==search:  
 return mid  
 elif self.array\_sorted[mid]>search:  
 return s1.Binary\_seach(low,mid-1,search)  
 else:  
 return s1.Binary\_seach(mid+1,high,search)  
 else:  
 return -1  
  
 def Fibonacci\_search(self,search):  
 *"""Works on sorted array.Preferred over Binary search because it divides array in unequal parts.  
 Does not use division instead use +.Time Complexity of O(logn)"""* fib2=0  
 fib1=1  
 fibadd=fib1+fib2  
  
 while(fibadd<len(self.array\_sorted)):  
 fib2=fib1  
 fib1=fibadd  
 fibadd=fib1+fib2  
  
 offset=-1  
  
 while(fibadd>1):  
 if fib2+offset>len(self.array\_sorted)-1:  
 i=len(self.array\_sorted)-1  
 else:  
 i=fib2+offset  
 """If less than search eliminate left side"""  
 if self.array\_sorted[i]<search:  
 fibadd=fib1  
 fib1=fib2  
 fib2=fibadd-fib1  
 offset=i  
 """If greater than search eliminate right side"""  
 elif self.array\_sorted[i]>search:  
 fibadd=fib2  
 fib1=fib1-fib2  
 fib2=fibadd-fib1  
 else:  
 return i  
  
 if fib1 and self.array\_sorted[len(self.array\_sorted)-1]==search:  
 return len(self.array\_sorted)-1  
 return -1  
  
  
s1 = Searching()  
s1.create()  
menu=True  
while menu:  
 print("\*\*\*\*MENU\*\*\*\*\*")  
 print("1.Linear search")  
 print("2.Sentinel search")  
 print("3.Binary Search")  
 print("4.Fibonacci Search")  
 print("5.Exit")  
 ch = int(input("Enter operation to be performed"))  
  
 if ch==1:  
 print(s1.array)  
 a=s1.Linear\_Search()  
 if a==-1:  
 print("Roll number not found")  
 else:  
 print("Roll number found at ",a)  
  
 if ch==2:  
 print(s1.array)  
 a = s1.Sentinel\_search()  
 if a == -1:  
 print("Roll number not found")  
 else:  
 print("Roll number found at ", a)  
  
 if ch==3:  
 print(s1.array\_sorted)  
 x=int(input("Enter Roll number you want to search"))  
 a=s1.Binary\_seach(0,len(s1.array)-1,x)  
 if a == -1:  
 print("Roll number not found")  
 else:  
 print("Roll number found at ", a)  
 if ch == 4:  
 print(s1.array\_sorted)  
 x = int(input("Enter Roll number you want to search"))  
 a = s1.Fibonacci\_search(x)  
 if a == -1:  
 print("Roll number not found")  
 else:  
 print("Roll number found at ", a)  
 if ch==5:  
 print("THANK YOU")  
 menu=False

**OUTPUT:**

**1.LINEAR SEARCH**

A screenshot of a computer screen

Description automatically generated with medium confidence

**2.SENTINEL SEARCH:**

A screenshot of a computer

Description automatically generated with medium confidence

**3.BINARY SEARCH:**

A screenshot of a computer

Description automatically generated with medium confidence

**4.FIBONACCI SEARCH:**

A screenshot of a computer

Description automatically generated with medium confidence