1. Search

19 Computational thinking and problem-solving

19.1 Algorithms

Candidates should be able to:	Notes and guidance
Show understanding of linear and binary searching methods	Write an algorithm to implement a linear search Write an algorithm to implement a binary search The conditions necessary for the use of a binary search How the performance of a binary search varies according to the number of data items
Show understanding of insertion sort and bubble sort methods	Write an algorithm to implement an insertion sort Write an algorithm to implement a bubble sort Performance of a sorting routine may depend on the initial order of the data and the number of data items
Show understanding of and use Abstract Data Types (ADT)	Write algorithms to find an item in each of the following: linked list, binary tree Write algorithms to insert an item into each of the following: stack, queue, linked list, binary tree Write algorithms to delete an item from each of the following: stack, queue, linked list Show understanding that a graph is an example of an ADT. Describe the key features of a graph and justify its use for a given situation Candidates will not be required to write code for this structure.
Show how it is possible for ADTs to be implemented from another ADT	Describe the following ADTs and demonstrate how they can be implemented from appropriate built-in types or other ADTs: stack, queue, linked list, dictionary, binary tree
Show understanding that different algorithms which perform the same task can be compared by using criteria (e.g. time taken to complete the task and memory used)	Including use of Big O notation to specify time and space complexity

---

Search

Prerequisite:

```
static int[] mylist1= {12,45,21,22,7,39,88};
static int[] mylist2= {7,12,21,22,39,45,88};
```

Linear search

- [ ] 9618 s21 41 Q2

```
public static int linearSearch(int item){
    for(int i=0;i<mylist1.length;++i){//从头开始一个一个看数组里的`element`是否是正在找的`item`
        if(item==mylist1[i]){//如果相等
            return i;//返回当前index
        }
    }
    System.out.println("item not found");//如果找到最后了都没找到 则打印"item not found"
    return -1;//终止程序并返回一个不是index的数字
}
```

```
FUNCTION LinearSearch(item:INTEGER, arr: ARRAY[1:10] OF INTEGER) RETURNS INTEGER
        FOR i <- 1 TO 10
                IF arr[i] = item
                        THEN
                                OUTPUT "Item found"
                                RETURN i
                ENDIF
        NEXT i
        OUTPUT "Item not found"
        RETURN -1
ENDFUNCTION
```

Binary search

> 二分查找 可以用递归Recursion实现 也可以用循环iteration实现

```
public static int binarySearch(int item){//the item to be searched
    int lwb=0,upb=7;//lowerbound和upperbound的初始化
    int mid;
    Boolean found=false;//初始化
    while(!found && lwb<=upb){//还没找到且范围正常
        mid=(lwb+upb)/2;//算出区间中值 然后对比区间中间的element和寻找的element的大小
        if(mylist2[mid]==item){//如果刚好找到 则fouund=true, 终止循环
            found=true;
        }
        else if(mylist2[mid]<item){//如果要找的item比区间中值大 则lwb变成
```

```
mid+1
            lwb=mid+1;
        }
        else{
            upb=mid-1;
        }
    }
    if(found){
        return mid;
    }
    else{
        System.out.println("item not found");
        return -1;
    }
}

public static int recursiveBinarySearch(int[] bsarr, int lwb, int upb,
int snum){//snum: the number to be searched; bsarr: array for binary
search
    int mid=(upb-lwb)/2+lwb;
    if(lwb>upb){
        System.out.println("not found");
        return -1;
    }
    if(bsarr[mid]==snum){
        return mid;
    }
    else if(bsarr[mid]<snum){
        return recursiveBinarySearch(bsarr, mid+1, upb, snum);
    }
    else{
        return recursiveBinarySearch(bsarr, lwb, mid-1, snum);
    }
}
```

```
FUNCTION binarySearch(item:INTEGER, arr: ARRAY[1:10] OF INTEGER)
        DECLARE upb: INTEGER
        DECLARE lwb: INTEGER
        DECLARE mid: INTEGER
        upb <- 10
        lwb <- 1
        WHILE lwb<upb
                mid<-(upb+lwb)/2 // mid <- (upb-lwb)/2+lwb
```

```
                IF arr[mid]=item
                        THEN
                                OUTPUT "found"
                                RETURN mid
                        ELSE
                                IF arr[mid]<item
                                        THEN
                                                lwb=mid+1
                                        ELSE
                                                upb=mid-1
                                ENDIF
                ENDIF
        ENDWHILE
OUTPUT "not found"
RETURN -1
```

Running:

```
System.out.println("binary search:");
System.out.println("Position of 7 in mylist2: "+ binarySearch(7));
System.out.println("Position of -2 in mylist2: ");
binarySearch(-2);
System.out.println("linear search:");
System.out.println("Position of 22 in mylist1: "+linearSearch(22));
System.out.println("Position of -2 in mylist1: ");
linearSearch(-2);
```

Result:

```
binary search:
Position of 7 in mylist2: 0
Position of -2 in mylist2:
item not found
linear search:
Position of 22 in mylist1: 3
Position of -2 in mylist1:
item not found
```