Linear Search

LinearSearch.java

```java
package LinearSearch;

public class LinearSearch {

    public int linearSearch(int[] a, int x){

        int ans = -1;

        for (int i = 0; i < a.length; i++){  // Time complexity : O(n)
            if (a[i] == x){
                ans = i;
                return i;
            }
        }

        return -1;
    }
}
```

*. Question: "static" in Java function?

ex.

```
public class App {

    public static void main(...)

    public static int linearSearch(...)

}
```

↳ 加上 static 後
不需創 App obj 即可
使用此 function

? This is a class method rather than a instance method.

Binary Search — Recursion

Binary Search Recursion

*. 前提: Data must be sorted!

*. Procedure: public int binarySearch(int x, int[] A)

1° p=0, r= A.length-1

2° while (p ≤ r) do

a) set q= r+p/2 ← integer division 会自动舍去小数点

b) if A[q] = x, return q

c) if A[q] > x, r= q"-1" ← minus!

else A[q] < x, p= q"+1" ← plus!

3° return -1

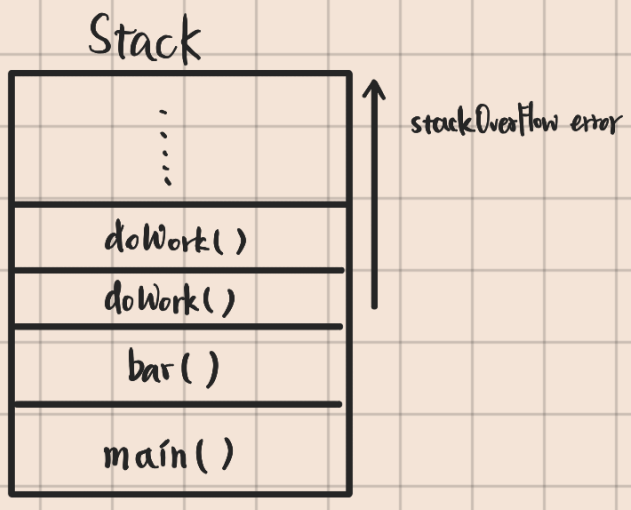
BinarySearch.java × SearchApp.java × Recursion.java ×

```java
package SearchAlgorithm;

public class BinarySearch {

    public static int binarySearch(int[] a, int x) {

        int p = 0;
        int r = a.length - 1;

        while (p <= r){

            int q = (p + r) / 2;

            if(x < a[q]) r = q-1;
            else if (x > a[q]) p = q+1;
            else return q;

        }
        return -1;
    }

}
```

Recursion

bar(){ doWork(); }

doWork(){ doWork(); }



*. Recursive.java

BinarySearch.java | SearchApp.java | Recursion.java | DoublyLinkedListApp.java

```java
public class Recursion {

    public static void reduceByOne(int n){

        if (n >= 0){  // Base case : condition for recursion to stop
            reduceByOne( n: n - 1);
        }

        System.out.println("Complete call: " + n); // executed after recursive function
    }                                                 從1 print到 n

    public static int recursiveLinearSearch(int[] A, int i, int x) {

        if (i > A.length - 1) return -1;  // x was not found in array
        else if (A[i] == x) return i;     // found x
        else return recursiveLinearSearch(A,  i: i + 1, x); // index+1 & search again
                    是"return" recursive function
    }

    public static int recursiveBinarySearch(int[] A, int p, int r, int x){

        System.out.println("[ "+ p + "..." + r + " ]"); → print 每輪 p,r

        if (p > r) return -1;
        else {
            int q = (p + r)/2;                          p  x  A[q]  r   A[q] > x → 小於"r"

            if (A[q] == x) return q;
            else if (A[q] > x) return recursiveBinarySearch(A, p, r: q-1 , x);
            else return recursiveBinarySearch(A, p: q+1, r , x);
        }
    }                                                   p  A[q]  r
}                                                           x   A[q] > x → 小於"p"
```

* Main

SearchApp.java | Recursion.java | BinarySearch.java | DoublyLinkedListApp.java

```java
package SearchAlgorithm;
```

```java
import ...

public class SearchApp {

    public static void main(String[] args) {

//        System.out.println(binarySearch(new int[] {1, 2, 3, 4, 5}, 1));
//        linearSearch(new int[] {1, 2, 3, 4, 5}, 1);
//        System.out.println(recursiveLinearSearch(new int[] {1, 2, 3, 4, 5}, 0, 5));
        System.out.println(recursiveBinarySearch(new int[] {1, 2, 3, 4, 7, 9, 12, 18},
                p: 0,  r: 7,  x: 18));
    }

}
```

first element index

last element index

*. Big O Notation of Binary Search

Binary Search → $O(\log n)$

原理：$\log_{10} 100 = 2$ ← 100 為 10 的 2 次方

$\log_2 4 = 2$ ← 4 為 2 的 2 次方

$\log_2(n)$ ← 表示 n 除以幾次 2 後會 = 1，n 代表 input size

Binary Search 執行次數：input array size 可以執行幾次 2 分法

∴ Time complexity = $\log_2(n)$，一般寫 $\log(n)$