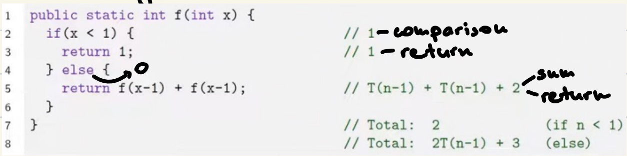
Video 1. Recursive algorithms



```
public static int f(int x) {
  if(x < 1) {                        // 1 – comparison
    return 1;                        // 1 – return
  } else {  0
    return f(x-1) + f(x-1);          // T(n-1) + T(n-1) + 2  sum
  }                                                          return
}                                    // Total:  2          (if n < 1)
                                     // Total:  2T(n-1) + 3  (else)
```

$T(n) = c_1 + 2T(n-1) \quad n>0$

$T(0) = c_0$

$T(n) = 3c_1 + 4T(n-2)$

$= 7c_1 + 8T(n-3)$

$T(n) = (2^k - 1)c_1 + 2^k T(n-k) \quad k=n \Rightarrow T(n) = (2^n - 1)c_1 + 2^n c_0 \quad O(2^n)$

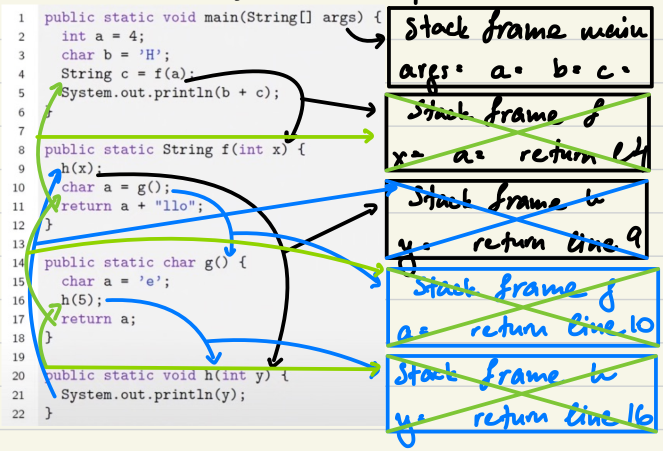
Video 2. Space complexity

call stack:
- method arguments
- local variables
- return address

heap:
- arrays
- objects

primitives + reference

stack - LIFO
Last in, First out
-> push new element
-> pop an element



```
1   public static void main(String[] args) {
2     int a = 4;
3     char b = 'H';
4     String c = f(a);
5     System.out.println(b + c);
6   }
7
8   public static String f(int x) {
9     h(x);
10    char a = g();
11    return a + "llo";
12  }
13
14  public static char g() {
15    char a = 'e';
16    h(5);
17    return a;
18  }
19
20  public static void h(int y) {
21    System.out.println(y);
22  }
```

Stack frame main
args: a= b= c=

~~Stack frame f~~
~~x= a= return l4~~

~~Stack frame h~~
~~y= return line 9~~

~~Stack frame g~~
~~a= return line 10~~

~~Stack frame h~~
~~y= return line 16~~