Lecture 3 - January 14

Recursion: Part 1
Asymptotic Analysis of Algorithms

splitArray: Implementation and Tracing

Announcements/Reminders

- Assignment 1 released
- Office Hours: 3pm to 4pm, Mon/Tue/Wed/Thu
- Contact Information of TAs on common eClass site

Problem on Recursion

https://codingbat.com/prob/p185204

Given an array of ints, is it possible to divide the ints into two groups, so that the sums of the two groups are the same. Every int must be in one group or the other. Write a recursive helper method that takes whatever arguments you like, and make the initial call to your recursive helper from splitArray(). (No loops needed.)

splitArray([2, 5, 3]) → time.

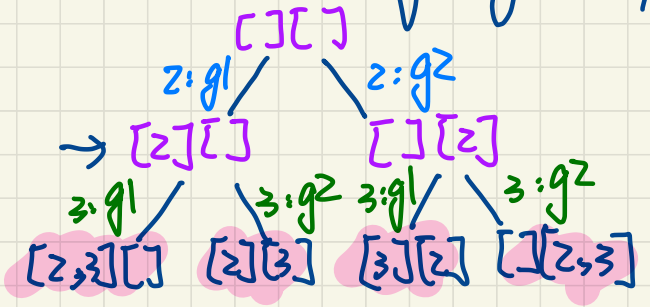
splitArray([2, 2]) → true
splitArray([2, 3]) → false
splitArray([5, 2, 3]) → true

Insight: The recursive and base cases are defined s.t. the tree of runtime recursive calls represents an exhaustive search on all the input scenarios.

worst case

Intuition

MATH 2030: How many ways to put n elements into m groups?



splitArray: Java Implementation

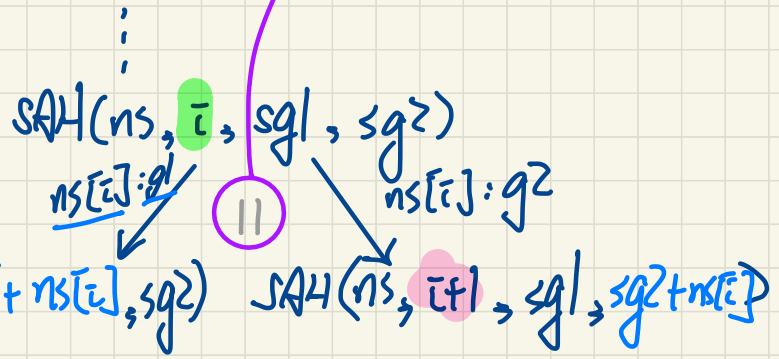
empty groups.

```java
public boolean splitArray(int[] ns) {
    return splitArrayHelper(ns, 0, 0, 0);
}
```

starting to group elements from index 0

```java
private boolean splitArrayHelper(int[] ns, int i, int sumOfGroup1, int sumOfGroup2) {
    if(i == ns.length) {
        return sumOfGroup1 == sumOfGroup2;
    }
    else {
        return
            splitArrayHelper(ns, i + 1, sumOfGroup1 + ns[i], sumOfGroup2)
            ||
            splitArrayHelper(ns, i + 1, sumOfGroup1, sumOfGroup2 + ns[i]);
    }
}
```

short-circuit eval.



Math

Commutativity:

$$p \wedge q \equiv q \wedge p$$
$$p \vee q \equiv q \vee p$$

Java

Evaluation orders matter

&& || do not commute

(2.1) $p \;\&\&\; q \neq q \;\&\&\; p$

(2.2) $p \;||\; q \neq q \;||\; p$

① Evaluation : left to right

(2.1) p eval. to T → still eval q
p eval. to F → skip eval of q → overall F

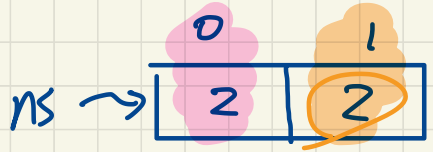
(2.2) p eval. to F → still eval q
p eval. to T → skip eval. of q → overall T

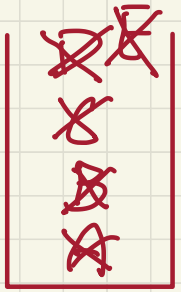
splitArray: Tracing (1)

```java
public boolean splitArray(int[] ns) {
    return splitArrayHelper(ns, 0, 0, 0);
}
```

```java
@Test
public void testSplitArray_01() {
    RecursiveMethods rm = new RecursiveMethods();
    int[] input = {2, 2};
    assertEquals(true, rm.splitArray(input));
}
```

```java
private boolean splitArrayHelper(int[] ns, int i, int sumOfGroup1, int sumOfGroup2) {
    if(i == ns.length) {
        return sumOfGroup1 == sumOfGroup2;
    }
    else {
        return
            splitArrayHelper(ns, i + 1, sumOfGroup1 + ns[i], sumOfGroup2)
            ||
            splitArrayHelper(ns, i + 1, sumOfGroup1, sumOfGroup2 + ns[i]);
    }
}
```





splitArray: Tracing (2)

```java
public boolean splitArray(int[] ns) {
    return splitArrayHelper(ns, 0, 0, 0);
}
```

```java
private boolean splitArrayHelper(int[] ns, int i, int sumOfGroup1, int sumOfGroup2) {
    if(i == ns.length) {
        return sumOfGroup1 == sumOfGroup2;
    }
    else {
        return
            splitArrayHelper(ns, i + 1, sumOfGroup1 + ns[i], sumOfGroup2)
            ||
            splitArrayHelper(ns, i + 1, sumOfGroup1, sumOfGroup2 + ns[i]);
    }
}
```

```java
@Test
public void testSplitArray_04() {
    RecursiveMethods rm = new RecursiveMethods();
    int[] input = {5, 2, 2};
    assertEquals(false, rm.splitArray(input));
}
```

Exercise

(1) Trace (recursion tree)

(2) Trace (debugger)

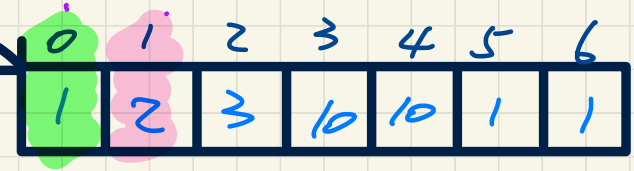
splitArray: Tracing (3)

```
@Test
public void testSplitArray_13() {
    RecursiveMethods rm = new RecursiveMethods();
    int[] input = {1, 2, 3, 10, 10, 1, 1};
    assertEquals(true, rm.splitArray(input));
}
```

```
public boolean splitArray(int[] ns) {
    return splitArrayHelper(ns, 0, 0, 0);
}
```

```
private boolean splitArrayHelper(int[] ns, int i, int sumOfGroup1, int sumOfGroup2) {
    if(i == ns.length) {
        return sumOfGroup1 == sumOfGroup2;
    }
    else {
        boolean possibility1 = splitArrayHelper(ns, i + 1, sumOfGroup1 + ns[i], sumOfGroup2);
        boolean possibility2 = splitArrayHelper(ns, i + 1, sumOfGroup1, sumOfGroup2 + ns[i]);
        return possibility1 || possibility2;
    }
}
```

splitArray: Tracing (3)



Level 0 (0, 0) $1 = 2^0$

Level 1 after insert. 1st elem. at index 0: 1:g1 (1, 0), 1:g2 (0, 1) $2 = 2^1$

Level 2 after insp. 2nd elem. at index 1: 2:g1, 2:g1, 2:g1, 2:g2 $4 = 2^2$

geometric seq. sum.

Level 7 after insp. 7th elem. at index 6: /\ /\ /\ . . . /\ /\ . . . /\ $= 2^7 = 128$