

# Recursive Backtracking

# The Backtracking Checklist

- **Find what choice(s) we have at each step.** What different options are there for the next step?
- For each valid choice:
  - **Make it and explore recursively.** Pass the information for a choice to the next recursive call(s).
  - **Undo if after exploring.** Restore everything to the way it was before making this choice.
- Find the base case(s). What should we do when we are out of decisions?



# groupSum problem


Given an array of ints, is it possible to choose a group of some of the ints, such that the group sums to the given target? This is a classic backtracking recursion problem. Once you understand the recursive backtracking strategy in this problem, you can use the same pattern for many problems to search a space of choices. Rather than looking at the whole array, our convention is to consider the part of the array starting at index **start** and continuing to the end of the array. The caller can specify the whole array simply by passing start as 0. No loops are needed -- the recursive calls progress down the array.

```
public boolean groupSum(int start, int[] nums, int target) {}
```

groupSum(0, [2, 4, 8], 10) → true

groupSum(0, [2, 4, 8], 14) → true

groupSum(0, [2, 4, 8], 9) → false



## groupSum - Hint

```
public static boolean groupSum(int start, int[] nums, int target) {}
```

`groupSum(0, [2, 4, 8], 10) → true`

`groupSum(0, [2, 4, 8], 14) → true`

`groupSum(0, [2, 4, 8], 9) → false`

The **base case** is when `start >= nums.length`. In that case, **return true** if `target==0`.

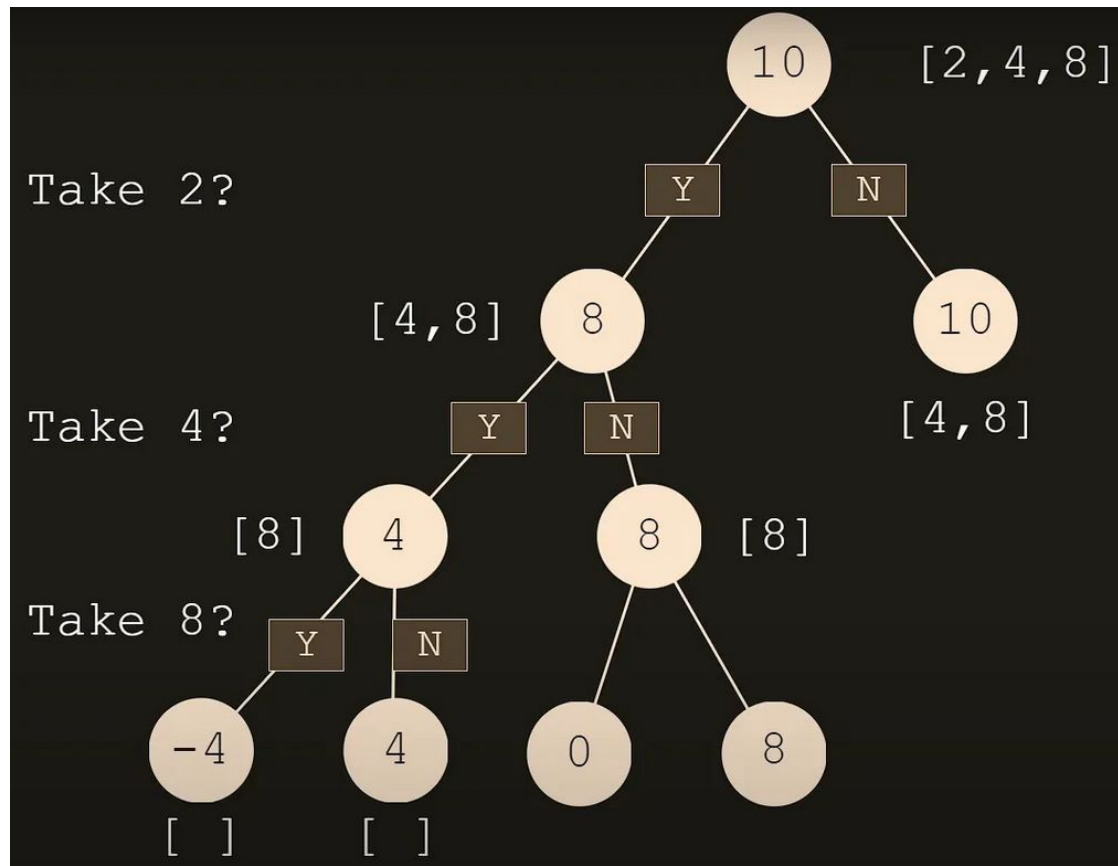
**Otherwise**, consider the element at `nums[start]`. The key idea is that there are only **2 possibilities** -- `nums[start]` **is chosen** or **it is not**.

**Make one recursive call** to see if a solution is possible **if `nums[start]` is chosen** (subtract `nums[start]` from `target` in that call).

**Make another recursive call** to see if a solution is possible **if `nums[start]` is not chosen**.

**Return true** if either of the two recursive calls returns true.





Source: Daniel Sutantyo

# Coding Time!!!

**Save your work here:**

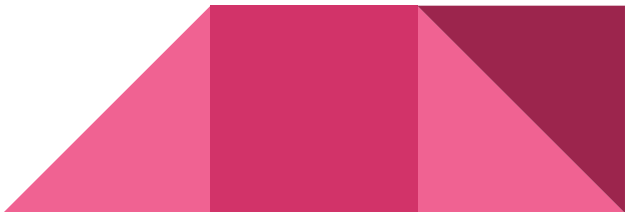
.../APCSA\_1/apcsa-assignments-YourUsername/classwork/38\_backtracking/Backtracking.java

**Method signature:**

```
public static boolean groupSum(int start, int[] nums, int target) {  
  
}
```

**Test Cases:**

```
groupSum(0, [2, 4, 8], 10) → true  
groupSum(0, [2, 4, 8], 14) → true  
groupSum(0, [2, 4, 8], 9)  → false
```



# CodingBat :o)

[https://codingbat.com/home/jnovillo@stuy.edu/apcsa\\_recursion\\_backtracking\\_v1](https://codingbat.com/home/jnovillo@stuy.edu/apcsa_recursion_backtracking_v1)

Log in to your CodingBat account.

- Go to your prefs page
- Teacher Share section: type my email jnovillo@stuy.edu
- Memo section: type YourPeriod\_YourLastName\_YourFirstName, like this  
01\_smith\_peter

