Programming Assignment 3: OrchardStroll

Dr. Sharma Thankachan

COP 3503 – Fall 2018

Due Monday, November 26th, at 11:59pm

1. Deliverable:

A **single source file** named *OrchardStroll.java* written in Java*.*

2. Objective:

The objective of this assignment is to test your understanding of dynamic programming and memoization. It includes some pre-written code that provides an example of clean, object-oriented Java.

3. Description:

You’re a giant mountain troll wandering the countryside, looking for food, work, and companionship. One day, you stumble upon an orchard with trees as far as your eyes can see. In the sea of green leaves and brown trunks before you, you spot thousands and thousands of splashes of color: brilliant yellows, vivacious reds, and a rainbow of other seductive hues. It’s a fruit orchard!

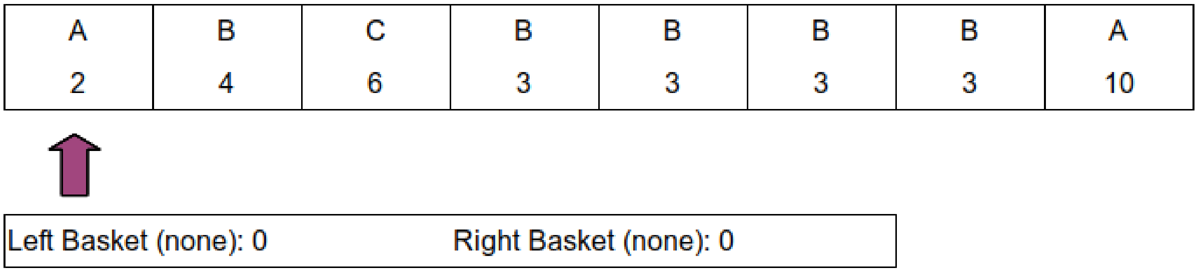
You lumber off the road and into a row of trees where you are quickly overwhelmed by the scents of the plump, succulent morsels all around. Apples! Bananas! Cherries! Jackfruit! This orchard has it all, and you begin to drool in anticipation. You crane your massive head to the side and spot two large wicker baskets sitting in the grass underneath the first tree. The earth screams and cracks under your humongous toes as you walk over to the baskets and pick them up. Both are empty and clean—left behind by a lazy worker the day before.

With baskets in both hands, you prepare for the feast of your life… but you’re a picky troll. You don’t like your foods to touch, and so you refuse to put fruits of different types in the same basket. Consequently, you can only carry two types of fruit at any given time. Your goal: collect as much fruit (by gross weight) as possible to feed your monstrous appetite.

You will be presented with an array of Tree objects—each Tree will have a fruit type and a gross weight associated it. Proceed through the array and collect as much fruit in your baskets as you can. Once you begin walking through the orchard row you may not turn around or idle, lest the orchard tender show up and run you off her land. You are free to dump out either basket at any time to make room for a different type of fruit, but only your final haul matters. Collect as much fruit as possible!

4. Example:

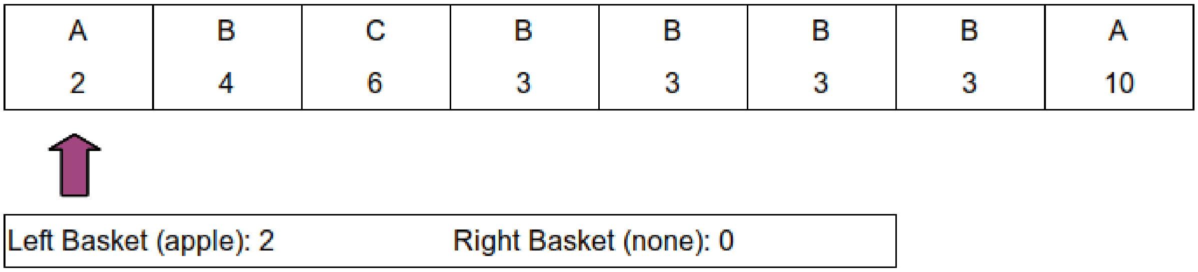
Here’s an example of a Tree array, with letters that represent fruit types and integers that represent gross weight. The arrow represents you, the troll. A’s are apples, B’s are bananas, and C’s are cherries.



A	B	C	B	B	B	B	A
2	4	6	3	3	3	3	10

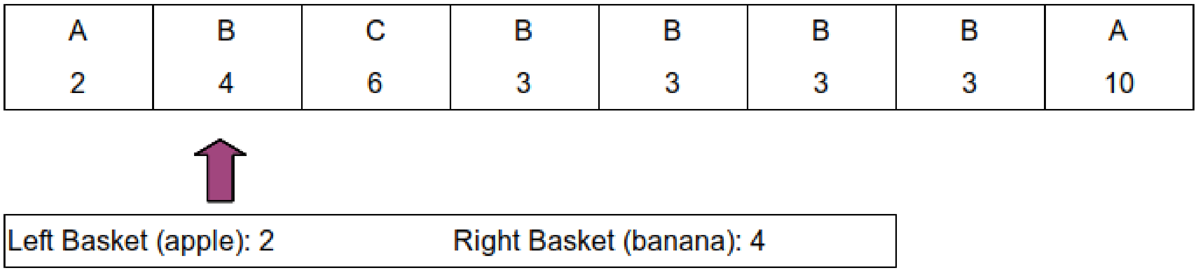
Left Basket (none): 0 Right Basket (none): 0

The first two trees can be collected without having to make a decision (your baskets are empty), so you collect both as you walk down the Tree array:



A	B	C	B	B	B	B	A
2	4	6	3	3	3	3	10

Left Basket (apple): 2 Right Basket (none): 0



A	B	C	B	B	B	B	A
2	4	6	3	3	3	3	10

Left Basket (apple): 2 Right Basket (banana): 4

The next tree contains a fruit that you don’t have in either of your baskets. You can either skip this fruit, or you can dump a basket and pick up this fruit. On the next page, decide to drop your apples (the lightest basket), and pick up the cherries.

A	B	C	B	B	B	B	A
2	4	6	3	3	3	3	10
		⇧					

Left Basket (cherry): 6 Right Basket (banana): 4

A	B	C	B	B	B	B	A
2	4	6	3	3	3	3	10
			⇧				

Left Basket (cherry): 6 Right Basket (banana): 7

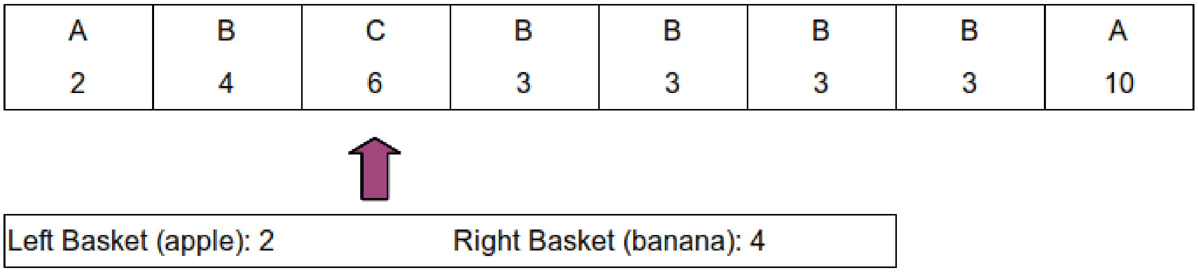
A	B	C	B	B	B	B	A
2	4	6	3	3	3	3	10
						⇧	

Left Basket (cherry): 6 Right Basket (banana): 16

A	B	C	B	B	B	B	A
2	4	6	3	3	3	3	10
							⇧

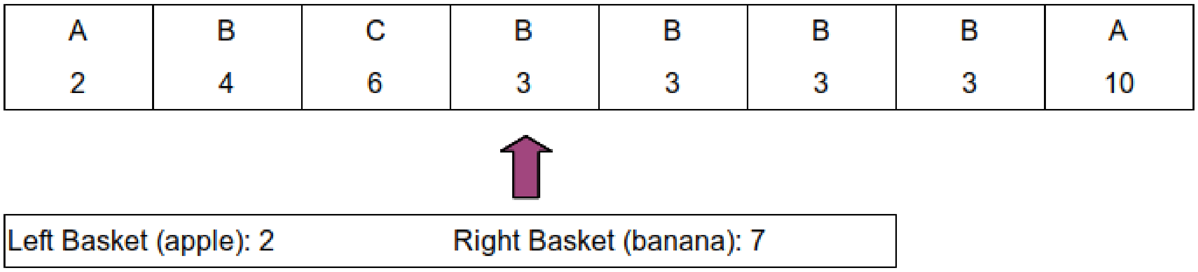
Left Basket (apple): 10 Right Basket (banana): 16

Once you reach the final tree, you decide to drop your cherries and pick up the final tree's apples (since this increases your haul's gross weight). At the end of this run, you have ten plus sixteen (twenty-six) fruits! On the next page, you decide instead to skip picking up the first cherries.



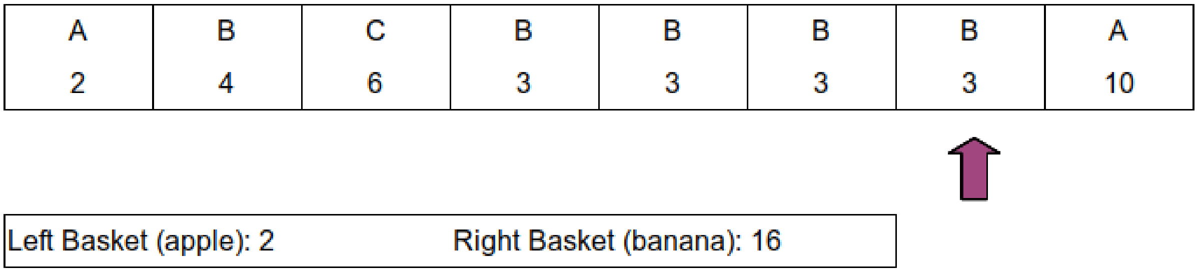
A	B	C	B	B	B	B	A
2	4	6	3	3	3	3	10

Left Basket (apple): 2 Right Basket (banana): 4



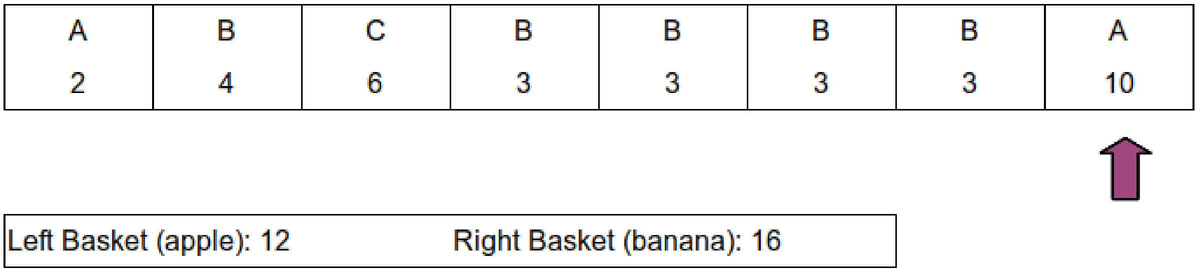
A	B	C	B	B	B	B	A
2	4	6	3	3	3	3	10

Left Basket (apple): 2 Right Basket (banana): 7



A	B	C	B	B	B	B	A
2	4	6	3	3	3	3	10

Left Basket (apple): 2 Right Basket (banana): 16



A	B	C	B	B	B	B	A
2	4	6	3	3	3	3	10

Left Basket (apple): 12 Right Basket (banana): 16

At the end of this run, you’ve collected twelve plus sixteen (twenty-eight) fruits! This is more than the last run, so skipping the first cherries was clearly the better call.

Your function must return the **MAXIMUM** value across all runs/possibilities. Use dynamic programming to avoid ridiculous runtimes that will fail the test cases.

5. Requirements:

You must write your assignment in the provided *OrchardStroll.java* source file. Do not edit the method signature that is provided. You may, however, write additional methods as you see fit. I suggest that any additional methods you add be static, and that you avoid using static class variables (unless you really, *really* know what you're doing with them).

```
public static int determineLargestHaul(Tree[] trees)
```

This method must determine the largest gross weight of fruit that can be collected from the provided array of trees using only two baskets. This method should then return that value as an integer. Your solution must employ iterative dynamic programming and it must run in linear (O(n)) time. See sections three and four for details on collecting fruit from this tree array.

NOTE: If you submit a recursive solution that does not employ dynamic programming, you will time out on all the grading test cases.

NOTE: If you submit a recursive dynamic programming solution, your stack will overflow on the grading test cases.

NOTE: If you submit an O(n) solution that does not use dynamic programming, you will not receive credit.

6. Input Specification:

The *determineLargestHaul()* function will receive an array of Tree objects with a length in the range [one, infinity). The array will **not** contain any null trees. Every tree will have a non-null type and will have a weight in the range [one, one million]. The different types of trees available can be found in the enumeration located within *Tree.java*. Finally, the return value for this function will **never** be so large that it overflows.

7. Testing:

To test, you will want to run the provided test script in a Linux environment. Your assignment will be graded on Eustis, and if your assignment does not work on Eustis then it **does not work.** For the script to function, you will need to set up your directory (folder) to look like this:

```
TestSuite/
        Test.sh
        Tree.java
        OrchardStroll.java
        OrchardStrollTester.java
```

To run the test script, use this command:

```
$ bash Test.sh
```

NOTE: The “$” character is not part of the command. It is a symbol frequently used to indicate the beginning of a terminal command.

8. Grading Criteria:

There are ten tests for this assignment and each test is weighed equally. The final assignment score will be out of one hundred. Here is how the tests are divided:

10/10 *determineLargestHaul()* unit tests

9. Final Notes:

Comments will not be graded generally, however I reserve the right to penalize assignments that include no comments, or whose comments are horrible. In the real world, comments are extremely useful and expected from developers, so get used to writing them. I also reserve the right to penalize assignments with atrocious/inconsistent style (or lack thereof). This does not mean that you will lose points for using one particular style over another—it only means that you need to apply your style consistently.

This assignment must be done individually. You may not review or use code from peers outside of your group, but you may have high-level discussions about this assignment with any of your peers.

Examples of **acceptable** peer questions about this assignment look like this:

- How exactly does memoization work?
- How do I reduce the runtime of a standard, recursive solution?
- What is the difference between a static method and a normal method?

Examples of **unacceptable** peer questions about this assignment look like this:

- Here's my code. How do I add memoization?
- Here's my code. How do I turn my recursive solution into a DP solution?
- Can you look at my code and tell me why I'm getting this error involving static variables and methods?

You MUST do all of the following:

- Include your name and NID in a comment at the top of your submission.
- Ensure that your submission compiles and runs on Eustis.
- Avoid including a package line in your source file.
- Submit the assignment before the deadline.
- Name your submission *OrchardStroll.java.*
- Write the assignment in Java.

If you do not follow the guidelines above you will receive a zero.