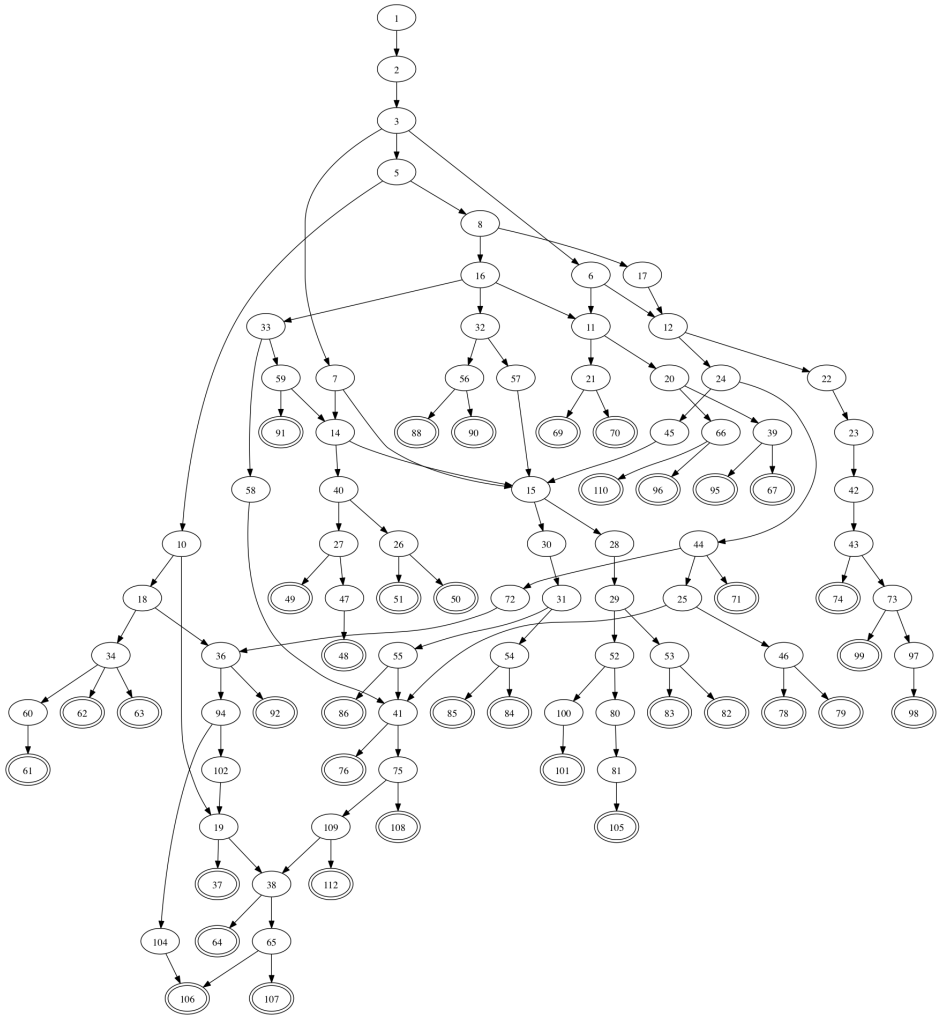
CS 145 Homework 6
Plot Plot

1 Overview

Choose Your Own Adventure books give the reader choices. For example, at the beginning of *War with the Evil Power Master*, by R.A. Montgomery, you receive an alert that two planets have been destroyed in a remote sector of space. The Evil Power Master claims responsibility. At the bottom of page 3, you are presented with a choice:

- If you decide to check out Sector 31-47-89 immediately, turn to page 6.
- If you decide to call in unit commanders for a meeting with Congress, turn to page 8.

Based on your choices, you end up with different endings. What happens if we plot all the possible paths through such a book? It'd probably look a bit like the following graph, which maps out *Deadwood City*, by Edward Packard:



You may find it helpful to read up a bit on these books:

- `http://mentalfloss.com/article/56160/brief-history-choose-your-own-adventure`
- `http://news.google.com/newspapers?id=_nUfAAAAIBAJ&sjid=XXUFAAAAIBAJ&dq=choose-your-own-adventure&pg=1663%2C2191360`
- `http://io9.com/remember-inside-ufo-54-40-the-unwinnable-choose-your-o-1552187271`
- `http://samizdat.cc/cyoa`

Your task in this homework is to develop an understanding of arrays. You will do so in the context of translating a `String` representation of a Choose Your Own Adventure book into a graph description. The graph description will be expressed in the DOT language. DOT graphs can be converted to images using the `dot` interpreter, available for download at `http://www.graphviz.org`. See `http://sandbox.kidstrythisathome.com/erdos` for a web-based converter. Producing images is not a required part of this assignment, but it's fun.

2 Requirements

Complete the classes described below. Place all code in package `hw6`.

2.1 Main

Write a class `Main`. It has a `main` method, which you are encouraged to use to test your code. Nothing in particular is required of it, but it must exist.

2.2 PlotPlot

Write a class `PlotPlot`, which contains several routines for composing a graph of a branching plot. It has the following public interface:

1. A method `getChoicesForOnePage` that accepts a `String` describing a single page and returns an `int` array of the next pages. If the `String` is just `"p"`, the page is a picture and `null` is returned. If the `String` is the empty `String`, return an array with no elements—which is different from `null`. Otherwise, the `String` looks something like `"23"` or `"15,40,99"`.

   The numbers represent the next possible pages that can be reached from this one. Multiple numbers indicate the reader has reached a fork in the plot, while a single number just directs the reader to the one next page in the current plot path. Whether there is one choice or several, return these numbers in an `int` array, but subtract one from each number to make future indexing easier. For example, `getChoicesForAllPages("15,40,99")` → `{14, 39, 98}`. `String.split` and `Integer.parseInt` may be helpful.

   Test this method before moving on. To print an array easily, use some thing like:

   ```
   System.out.println(Arrays.toString(pageChoices));
   ```

2. A method `getChoicesForAllPages` that accepts a `String` describing all pages of the book and returns a two-dimensional `int` array containing each page's possible next pages. The `String` looks something like `"2|3|5,6,7|p|||"`, with each page's choices separated by a "|". We interpret this example `String` as follows:

   (a) Page 1 leads to page 2.

   (b) Page 2 leads to page 3.

   (c) Page 3 leads to page 5, 6, or 7.

   (d) Page 4 is a picture.

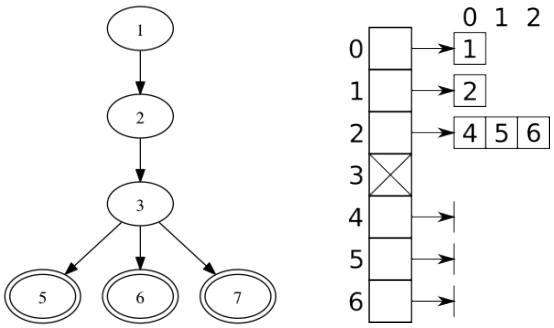
   (e) Page 5 is an end.

   (f) Page 6 is an end.

   (g) Page 7 is an end.

   Visually, this book has the plot structure shown in figure 1a.

   This method constructs and returns a two-dimensional array of choices for each page. The outer index of the array is the page number. The element at index i is an inner array of choices for page i. If page i is a picture, the element is `null`. If page i is an end, the element array has 0 elements. The corresponding array that is returned for the book `"2|3|5,6,7|p|||"`is shown in figure 1b.



(a) Plot plot (b) Choices array

Figure 1: Graph and array representations of `"2|3|5,6,7|p|||"`.

   Page numbers in the array are treated as one less than their actual value. For example, page 3 leads to pages 5, 6, and 7 in the printed book. In the array, an array containing 4, 5, and 6 is stored in element 2. Subtracting one makes using the page numbers as indices simpler.

   As you implement this method, don't redo the work of getting choices for one particular page. You have a method for that. Instead, take this approach:

```
split book into its pages
make 2D book choices array
for each page
  determine page's choices
  store page choices in the book choices array
```

`String.split` is helpful here, but it coalesces adjacent delimiters together, which will eliminate the ending pages, which have no choices. Additionally, the `|` character has a special meaning to `split`. To prevent the coalescing and take away the special meaning of `|`, use `book.split("\\|", -1)`.

Test this method before moving on. You can easily print a high-dimensional array with something like the following:

```
System.out.println(Arrays.deepToString(bookChoices));
```

3. A method `getForkPages` that accepts a 2D array as returned by `getChoicesForAllPages`. It returns an `ArrayList<Integer>` of all the pages that have two or more choices. For `"2|3|5,6,7|p|||"`, a list containing 2 is returned.

4. A method `getEndPages` that accepts a 2D array as returned by `getChoicesForAllPages`. It returns an `ArrayList<Integer>` of all the pages that end a path through the book. A page ends a path if it has no next choices and is not a picture. For `"2|3|5,6,7|p|||"`, a list containing 4, 5, and 6 is returned.

5. A method `getOrphanPages` that accepts a 2D array as returned by `getChoicesForAllPages`. It returns an `ArrayList<Integer>` of all the non-picture pages that are unreachable from other pages. Whether these orphans are interpreted as mistakes or hidden treasures largely depends on how secure you felt as a child.

   You might find it advantageous to create a little helper array that tracks which pages do have predecessors. Such an array can be constructed with the following pseudocode:

   ```
   create a hasPrevious array with an element for each page
   for each page p
     for each choice c from page p
       set c's hasPrevious to true
   ```

   Any pages that don't have predecessors and are not pictures are orphans.

6. A method `dotEndPages` that accepts a 2D array as returned by `getChoicesForAllPages`. It prints to `System.out` DOT commands to draw all the end page nodes with a double outline. For `"2|3|5,6,7|p|||"`, the following output is produced, with spaces marked as ␣:

   ```
   ␣␣5␣[peripheries=2];
   ␣␣6␣[peripheries=2];
   ␣␣7␣[peripheries=2];
   ```

   Note that this method produces output to be consumed by the user, so our little trick of subtracting one off the page numbers must be undone. The output page numbers are restored to their original values.

7. A method `dotOrphanPages` that accepts a 2D array as returned by `getChoicesForAllPages`. It prints to `System.out` DOT commands to draw all the orphan page nodes with a dashed outline. For `"2|3|5,6,7|p||||9,10||"`, the following output is produced:

   ```
   ␣␣8␣[style=dashed];
   ```

Each line is indented two spaces. Note that this method produces output to be consumed by the user, so our little trick of subtracting one off the page numbers must be undone. The output page numbers are restored to their original values.

8. A method `dotPlot` that accepts a 2D array as returned by `getChoicesForAllPages`. It prints to `System.out` DOT commands to draw edges between each page and its choices. For `"2|3|5,6,7|p||||9,10||"`, the following output is produced:

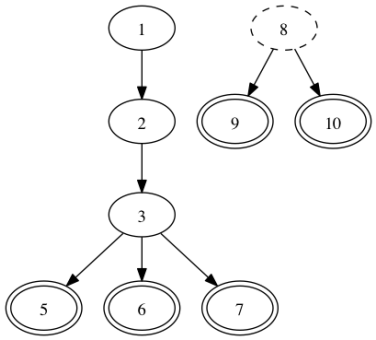
```
␣␣1␣->␣2;
␣␣2␣->␣3;
␣␣3␣->␣5;
␣␣3␣->␣6;
␣␣3␣->␣7;
␣␣8␣->␣9;
␣␣8␣->␣10;
```

Each line is indented two spaces. Note that this method produces output to be consumed by the user, so our little trick of subtracting one off the page numbers must be undone. The output page numbers are restored to their original values.

9. A method `dot` that accepts a 2D array as returned by `getChoicesForAllPages`. It prints to `System.out` DOT commands to draw a graph of the book, with end pages marked with a double outline, orphan pages marked with a dashed outline, and edges connecting each page to its choices. For `"2|3|5,6,7|p||||9,10||"`, the following output is produced:

```
digraph␣G␣{
␣␣5␣[peripheries=2];
␣␣6␣[peripheries=2];
␣␣7␣[peripheries=2];
␣␣9␣[peripheries=2];
␣␣10␣[peripheries=2];
␣␣8␣[style=dashed];
␣␣1␣->␣2;
␣␣2␣->␣3;
␣␣3␣->␣5;
␣␣3␣->␣6;
␣␣3␣->␣7;
␣␣8␣->␣9;
␣␣8␣->␣10;
}
```

After applying the `dot` interpreter to this output, the following graph is produced:



10. A method `getPathOfFirsts` that accepts a 2D array as returned by `getChoicesForAllPages`. It returns an `ArrayList<Integer>` of the sequence of pages that the reader follows when always taking the first choice. That is, the reader uses the following algorithm:

    ```
    start at page 0
    while I'm not yet at an end
      pick choice 0 of current page
    ```

    For `"2|3|5,6,7|p|||"`, the list returned is 0, 1, 2, 4.

3 Extra

Anyone who shares on Piazza a choices `String` as described in `getChoicesForAllPages` and its resulting graph for some Choose Your Own Adventure book not already posted will receive two extra credit participation points.

4 Submission

To submit your work for grading:

1. Put the SpecChecker for this homework in your Build Path. Run the SpecChecker as a Java Application and fix problems until all tests pass.
2. Commit and push your work to your repository. Verify that your solution is on Bitbucket.

A passing SpecChecker does not guarantee you credit. Your grade is conditioned on a few things:

- You must meet the requirements described above. The SpecChecker checks some of them, but not all.
- You must not plagiarize. Write your own code. Talk about code with your classmates. Ask questions of your instructor or TA. Do not look at others' code. Do not ask questions specific to your homework anywhere online but Piazza. Your instructor employs a vast repertoire of tools to sniff out academic dishonesty, including: drones, CS 145 moles, and a piece of software called MOSS that rigorously compares your code to every other submission. You don't want to live in a world serviced by those who squeaked by through questionable means. For your future self, career, and family, do your own work.
- Your code must be submitted correctly and on time. Most excuses devolve into, "I started too late." The fix for this problem is not an extension.