Branching Story to Graph ADT

Suppose we wanted to generalize the **BranchingStory** class from the interactive story example to be an abstract data type (ADT) that supported representing a directed graph with connections between any object.

1. How would you go about making this change? What operations would your ADT support?

One way to do this is to create a class (eg, called `Options`) that keeps track of all of the available options. This class will have:

- a String field for storing any introductory text
- an ArrayList that stores a list of available options. Another class can be created, called `Option`, and the ArrayList will store objects of that type.

The `Option` class stores an individual option, and will have (at minimum):

- a String field for storing the option text
- the method to execute when the option is selected
- a reference to the next node (if there are multiple options to choose from, then this could be another ArrayList of type `Option`)

The `Options`/`Option` classes can be renamed depending on the context.

Alternatively, a Map could also be used instead of an ArrayList. This would be of the type <String, Runnable>, where `String` is the node text, and `Runnable` refers to the method to run when the option is selected. Map<String, Option>, would also work (for storing additional information per option).

The operations this would support are:

- Add options to the list (either to the end, or at a specific index)
- Remove specific option from the list
- Get/set text for a specific option
- Get size of the options list
- Clear list
- Search by value, and return the index
- Check if list contains a specific string or Option object

2. What is an advantage of using an ADT in this context?

- ADT can be modified without needing to make changes outside of the class
- Higher level code makes it easier to understand
- Reusability for other contexts that require using a graph (eg, branching story example, Facebook friends (nodes), and every connection representing an edge, etc)