MODULE 6 PROJECT: HEAPS

Learning Objectives:

CLO 1. Identify fundamental data structures in computer science and their use in real-life applications.

CLO 3. Develop problem solving skills by implementing data structures.

CLO 4. Compare advantages and disadvantages of different data structure implementations.

Data Structure Implementation

1. Implement `BinaryHeap`

   Files to modify: BinaryHeap.py

   Directions: Finish the implementation of the following functions and methods:

   Functions:

   - `left(i)` - returns the index of the left child of the element at index `i`.
   - `right(i)` - returns the index of the right child of the element at index `i`.
   - `parent(i)` - returns the index of the parent of the element at index `i`.

   Methods:

   - `add(x)` - adds the element `x` to the heap. The resulting binary heap should still abide by the heap order.
   - `remove()` - removes the smallest element, i.e., the root. The resulting binary heap should still abide by the heap order. If the binary heap is empty, then an `IndexError` is raised.
   - `depth(u)` - returns the integer depth of element `u`, if found in the binary heap; otherwise raises a `ValueError` with message "`<insert u> is not found in the binary tree.`". The worst-case run-time of `depth(u)` should be $O(n)$.
   - `height()` - returns the height of this binary heap. The worst-case run-time should be $O(1)$. HINT: Use the built-in function `math.log2()` from the Python `math` module.
   - `bf_order()` - returns the `list` of values in this binary heap as they are traversed using the breadth-first traversal order.
   - `in_order()` - returns the `list` of values in this binary heap as they are traversed using the in-order traversal. HINT: You may need to create a helper method `_in_order(i)` that takes the index `i` of the root of the tree that you are performing the traversal on.
   - `post_order()` - returns the `list` of values in this binary heap as they are traversed using the in-order traversal. HINT: You may need to create a helper method `_post_order(i)` that takes the index `i` of the root of the tree that you are performing the traversal on.
   - `pre_order()` - returns the `list` of values in this binary heap as they are traversed using the in-order traversal. HINT: You may need to create a helper method `_pre_order(i)` that takes the index `i` of the root of the tree that you are performing the traversal on.

Helper Methods:

- `_bubble_up_last()` - helper method to `add(x)`; moves the element at index `self.n - 1` "higher" in the binary heap so that the heap order is maintained after an add operation.
- `_trickle_down_root()` - helper method to `remove()`; moves the element at the root "lower" in the binary heap so that the heap order is maintained after a remove operation.
- `resize()` - resizes the backing array a to maintain the invariant $n \leq len(a) \leq 3n$.

2. **(Optional) Test your data structure:**

   - Remove one element from an empty `BinaryHeap`, should result in an `IndexError` with message `"Can not remove from an empty heap."`
   - Add 3 elements: `add(2)`, `add(1)`, `add(5)`
   - Check that `size()` returns 3
   - Remove one element: `remove()` and check that it returns 1.
   - Add 3 elements: `add(4)`, `add(1)`, `add(3)`
   - Check that `size()` returns 5
   - Remove all the elements and check that they return in order 1, 2, 3, 4, 5.

Changes to the Bookstore System

Files to modify: BookStore.py

Directions:

Add a new method called `bestsellers_with(infix, structure, n = 0)` that searches the book catalog for the first `n` books containing the given string `infix`, stores them in a data structure determined by `structure`, and prints them in order of highest ranking to lowest ranking (i.e., highest num. copies sold to lowest num. copies sold).

Parameter `infix`:

- If `infix` is empty, no search is performed; instead the message `"Invalid infix."` is printed.

Parameter `structure`:

- If `structure` is integer 1, the books with the given infix will be added to an instance of `BinarySearchTree`. The rank of the book (i.e., the number of copies sold) will serve as the node key and the `Book` object will be the value. To print the books in order of highest rank to lowest rank, keep in mind that every left node will contain a rank smaller than the root, and in turn, smaller than the right node. Is there a traversal that allows you print the books in the order highest rank to lowest rank? (Hint: Yes.)
- If `structure` is integer 2, the books with the given infix will be added to an instance of `BinaryHeap`. The rank of the book (i.e., the number of copies sold) should determine the order of the binary heap. In particular, the root should contain the book with the largest number of copies sold out of all the books in the heap (Hint: Multiply all the ranks by -1 before adding to the binary heap so that the maximum becomes the minimum). Use the `remove()` method in `BinaryHeap` to print the books in order of highest rank to lowest rank.
- Any other value of `structure` should result in the printed message `"Invalid data structure."`

Parameter `n`:

- If `n` < 0, no search is performed; instead the message `"Invalid number of titles."` is printed.
- If `n` is 0, then the method adds **all** the books in the catalog with given infix.

Like other methods in `BookStore`, the method `bestsellers_with(...)` should display the amount of time it took to execute its main responsibilities. Use the template below to ensure this time is displayed:

```
def bestsellers_with(self, infix, structure, n = 0) :
        best_sellers = None
        if structure == 1:
            best_sellers = BinarySearchTree.BinarySearchTree()
        elif structure == 2:
            best_sellers = BinaryHeap.BinaryHeap()
        else:
            print("Invalid data structure.")

        if best_sellers is not None:
            if infix == "":
                print("Invalid infix.")
            else:
                start_time = time.time()
                # FIXME: Insert the rest of the implementation here
                elapsed_time = time.time() - start_time

                print(f"Displayed bestsellers_with({infix}, {structure}, {n}) in {elapsed_time} seconds")
```

Changes to Main

Files to modify: main.py

Directions:

Modify `menu_bookstore_system()` so that it includes a new option to search best-sellers containing a given infix. Your new list of options should read as follows:

```
s FIFO shopping cart
r Random shopping cart
1 Load book catalog
2 Remove a book by index from catalog
3 Add a book by index to shopping cart
4 Remove from the shopping cart
5 Search book by infix
6 Get cart best-seller
7 Add a book by key to shopping cart
8 Add a book by title prefix to shopping cart
9 Search best-sellers with infix
0 Return to main menu
```

If the user chooses option 9, the system should prompt for an infix, a structure integer, and for the max number of titles to consider. The prompt should read as follows,

```
Enter infix: <Insert infix>
Enter structure (1 or 2):  <Insert integer>
Enter max number of titles: <Insert integer>
```

(Optional) Test your program:

1. Load the catalog with `books.txt`.
2. Search for books by:
   (a) Empty prefix. Error message `"Invalid infix"` should be printed.
   (b) `bestsellers_with("World of Po", 1)`
   (c) `bestsellers_with("World of Po", 2)`
   Both non-empty prefixes should return the following books (the only potential difference being in the time it takes for each call to terminate):

```
Book: 076580915X
Title: Making Development Work: Development Learning in a World of Poverty and Wealth (World Bank Series on Evaluation and Development, 4)
Group: Book
Rank: 2174510

Book: 0521894433
Title: World of Possibilities : Flexibility and Mass Production in Western Industrialization (Studies in Modern Capitalism)
Group: Book
Rank: 1284397

Book: 0226900177
Title: The Politics of Difference : Ethnic Premises in a World of Power
Group: Book
Rank: 1155407

Book: 0525444521
Title: The World of Winnie-the-Pooh (Two Volume Slipcased Set: The World of Pooh and The World of Christopher Robin)
Group: Book
Rank: 797901

Book: 1578062500
Title: Dubose Heyward: A Charleston Gentleman and the World of Porgy and Bess
Group: Book
Rank: 794685

Book: 0966636104
Title: The Puzzling World of Polyhedral Dissections
Group: Book
Rank: 547071

Book: 0395924871
Title: Shtetl: The Life and Death of a Small Town and the World of Polish Jews
Group: Book
Rank: 293490

Book: B000006ODM
Title: World of Pop Hits of 70's
Group: Music
Rank: 234485

Book: 0892366877
Title: The Lost World of Pompeii
Group: Book
Rank: 219707

Book: 0525444475
Title: The World of Pooh : The Complete Winnie-the-Pooh and The House at Pooh Corner (Pooh Original Edition)
Group: Book
Rank: 28615

Book: B00005YUPP
Title: Baby Einstein - Baby Shakespeare - World of Poetry
Group: DVD
Rank: 1794
```

   (d) `bestsellers_with("World of Po", 2, 5)` should display the **first five** books listed above.

SUBMISSION PROCESS

Submit to CodePost

- `BinaryHeap.py`
- `main.py`
- `BookStore.py`