CS 261P Project #2: Fibonacci Heap

In this project you are to implement the Fibonacci Heap data structure.

Project Files:

You will need to download three files for this project. Download the lab2-files-2024.zip file using this **canvas** link.

The zip file contains 3 files: fib.py, project2_tests.py, and requirements.py.

Go ahead and make a copy of fib.py and name it fib_lazy.py. This will be for a lazy delete-min implementation.

Project Specification:

The specification for the project is given in the file **requirements.py**. Read it carefully and completely before starting the project.

The information for fib_lazy.py is here:

In addition to implementing the fibonacci heap as described in lecture for fib.py, you will implement a variant of fibonacci heap called ***F-heaps with vacant nodes***. We suggest you finish implementing the regular fibonacci heap in fib.py first, then make a copy of that file to use the template for fib_lazy.py

This is described in section 3 “Variants of Fibonacci Heaps” of the original paper:
https://www.cs.princeton.edu/courses/archive/fall03/cs528/handouts/fibonacci%20heaps.pdf

Here is the relevant text:

Tarjan [4]. This idea applied to F-heaps gives our first variant, *F-heaps with vacant nodes*, which we shall now describe.

We perform a *delete min* or *delete* operation merely by removing the item to be deleted from the node containing it, leaving a vacant node in the data structure (which if necessary we can mark vacant). Now deletions take only $O(1)$ time, but we must modify the implementations of *meld* and *find min* since in general the minimum node in a heap may be vacant. When performing *meld*, if one of the heaps to be melded has a vacant minimum node, this node becomes the minimum node of the new heap. To perform *find min* (h) if the minimum node is vacant, we traverse the trees representing the heap top-down, destroying all vacant nodes reached during the traversal and not continuing the traversal below any nonvacant node. This produces a set of trees all of whose roots are nonvacant, which we then link as in the original implementation of *delete min*. (See Figure 8.) The following

Figure 8 from the paper illustrates this design:

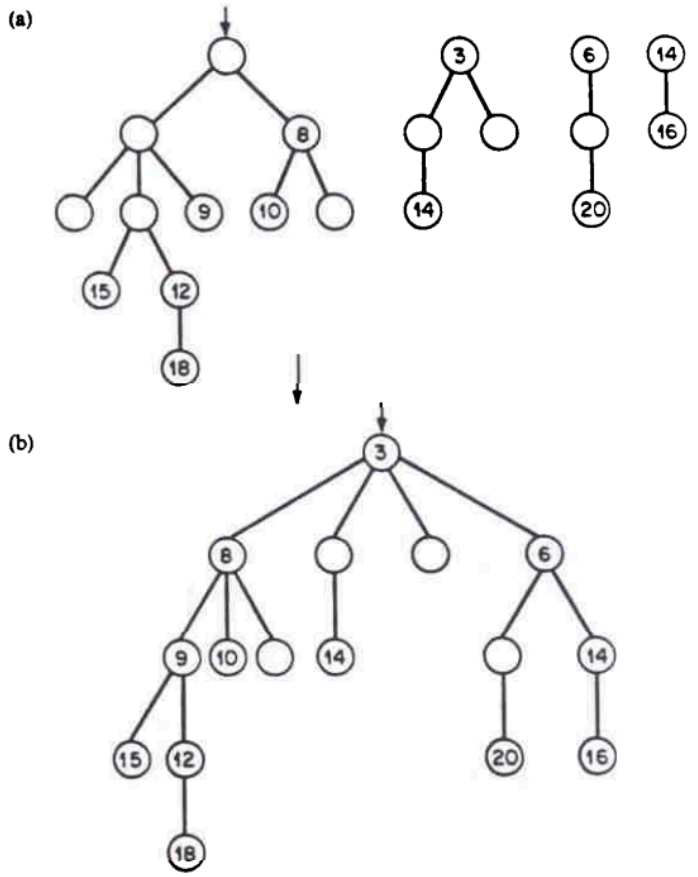

FIG. 8. A *find min* operation on a heap with vacant nodes. (a) The original heap. (b) After *find min*. Subtrees with roots 15, 12, 9, and 8 become trees, and then linking takes place.

The project deliverables are the file **fib.py** and **fib_lazy.py**, modified by replacing the code stubs of the required methods with working code.

Project Testing:

The file **project2_tests.py** contains a few tests that you can use to test your code. As stated there, these tests are just to get you started, and passing these tests does not mean that your code is correct. We strongly recommend that you create more tests, with varying data structure sizes and operation sequences, to test your code more completely.

Project submission

The project is to be submitted to GradeScope. Instructions for what to submit are in the file **requirements.py**.

Project due date/time

The project is due on **Tuesday, February 27, at 11:59 PM PST.**
Late projects will be penalized 2.5% for every hour the project is late, rounded up to the nearest hour.

Miscellaneous notes

The projects are to be done **individually** or in **pairs** (**no groups of more than 2 students**) and there should be one submission per group. Make sure to indicate all members of the group when submitting on Gradescope.

If you have questions about the project of general interest, the best way to ask your question is on the Ed Discussion message board.