CS 234 Fall 2019
Naomi Nishimura

Assignment 4

Coverage: Modules 6-9.

This assignment consists of a written component and a programming component. Please read the course website carefully to ensure that you submit each component correctly.

All the questions in the written component should be submitted as a single pdf file with the name `a4written.pdf`. Although scanned documents are acceptable, please keep in mind that only legible writing will be marked (subject to the markers' discretion) and that MarkUs forces a limit of 5000 KB per file, which might be exceeded by high resolution scans.

Written component

For full marks, you are expected to provide a brief justification of any answer you provide.

W1. *[2 marks]* Describe how you would modify the depth-first search pseudocode given in lecture to colour each edge of the tree formed black. (You can assume that the vertices are all initially white, and that it does not matter how or if these colours are changed by your pseudocode algorithm.) You can assume that the ADT has been augmented with the operation SET_EDGE_COLOUR*(G, One, Two, New)* that assigns the colour *New* to the edge with endpoints *One* and *Two*. Refer to line numbers in the pseudocode and explain where you would add, delete, or modify lines in the pseudocode.

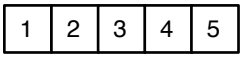
W2. *[5 marks]* Suppose you are using a (2,3) tree to implement the ADT Priority Queue. For each of the following ADT operations, briefly describe the algorithm used to implement the operation and state and justify the worst-case running time, assuming a linked implementation of the (2,3) tree.

(a) *[3 marks]* DELETE_MIN

(b) *[2 marks]* ADD

W3. *[3 marks]* In this question we consider the **alternating heuristic**. In this heuristic, the first step and every subsequent odd step uses move-to-front, and the second step and every subsequent even step uses transpose.

(a) *[2 marks]* Starting with the ordering of values illustrated in the figure below, use the alternating heuristic to search for items in the order 4, 5, 2, 1, and then 3. Show the order of the values after each step, including the resulting final order.



(b) *[1 mark]* Give the worst-case cost of a LOOK_UP operation using the alternating heuristic. Briefly justify your answer.

W4. *[4 marks]* In this question, we implement an ADT Dictionary using hashing by separate chaining, where each bucket is implemented as an AVL tree. Give answers in Θ notation as a function of n, the total number of data items stored, N, the total number of buckets.

(a) *[2 marks]* What is the worst-case cost of a LOOK_UP operation? Briefly justify your answer.

(b) *[2 marks]* What is the **best-case cost** of a LOOK_UP operation? Briefly justify your answer.

W5. *[6 marks]* Suppose that for a hash function $f(k)$ and a hash table of size N, for key k we use the probe sequence $f(k) \bmod N$, $(f(k)+3) \bmod N$, $(f(k)+6) \bmod N$, and so on (that is, position i in the probe sequence is $(f(k)+3i) \bmod N$).

(a) *[2 marks]* Discuss the advantages and disadvantages of this choice.

(b) *[1 mark]* Is there a choice of $f(k)$ that would be particularly good or particularly bad for this hashing scheme, but not necessarily for other hashing schemes? (Limit your discussion to hashing schemes covered in lecture.)

(c) *[1 mark]* Is there a choice of N that would be particularly good or particularly bad for this hashing scheme, but not necessarily for other hashing schemes? (Limit your discussion to hashing schemes covered in lecture.)

(d) *[1 mark]* What is the worst-case cost of searching for an item in a hash table storing n items?

(e) *[1 mark]* Describe an algorithm for finding the smallest value stored in the hash table when using the scheme described in this question.

Programming component

Please read the course website carefully to ensure that you are using the correct version of Python and the correct style. For full marks, you are required not only to have a correct solution, but also to adhere to the requirements of the assignment question and the style guide, including aspects of the design recipe.

Although submitting tests is not required, it is highly recommended that you test your code. For each assignment question, create a testing file that imports your submission and tests the code. Do not submit your testing file.

In this question, you will be writing functions that consume arrays (as defined in `contiguous.py`) and treating them as heaps. You will not be using operations of the ADT Priority Queue, but instead you will be directly handling the contiguous data structure.

For your convenience, we have provided an extra module `makecontiguous.py` that can be used to quickly fill arrays for use in testing your work.

For any of the programming questions in this assignment, you may import any of the following files: `check.py`, `contiguous.py`, and `makecontiguous.py`. The file `helpersmodel.py` may be used only for the second and third questions.

P1. *[5 marks]* In this question you will be writing four helper functions for use in the next two questions. A template for the functions can be found in the file `asst4helperstemplate.py`.

Submit your work in a file with the name `helpers.py`.

P2. *[7 marks]* Write a function `heap_order` that consumes an array of numbers and produces `True` if the array can be interpreted as a heap stored in a contiguous manner. You can assume that each slot in the input array contains a distinct number. You should import `contiguous.py` using the line `from contiguous import *`.

You may wish to use the solution to Programming Question P1 in your solution, and may do so even if you are not able to complete it. To use the model solution, import the file `helpersmodel.py` to use our solution to P1 by including the line `from helpersmodel import *`. You are not required to use the helper functions created in Question P1 but doing so is likely to make your code easier to read.

Submit your work in a file with the name `heaporder.py`.

P3. *[8 marks]* Write a function `heapify` that consumes an array of numbers in which each slot contains a distinct number and as a side effect rearranges the values in the array to form a heap, using the heapify algorithm discussed in class. Your function does not need to produce anything.

You may wish to use the solution to Programming Question P1 in your solution, and may do so even if you are not able to complete it. To use the model solution, import the file `helpersmodel.py` to use our solution to P1 by including the line `from helpersmodel import *`.

Submit your work in a file with the name `heapify.py`.