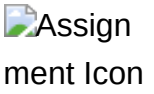
Assignment 5: Hash Map and Min Heap Implementation (Portfolio Assignment)

Portfolio Assignment



The purpose of the Portfolio Assignment is to showcase your programming skills to potential employers and colleagues. This is the only assignment in this course that is allowed to be publicly posted online (e.g. GitHub, personal website, etc. …). While this is a great opportunity to publicize your work, it is not required that you post the assignment online.

This assignment is comprised of 3 parts. In the first part, you will complete the implementation of a **hash map (https://canvas.oregonstate.edu/courses/1776054/pages/exploration-introduction-to-maps-and-hash-tables)** . In the second part, you will complete the implementation of **heap. (https://canvas.oregonstate.edu/courses/1776054/pages/exploration-heap-implementation)** In the third and final part, in addition to the programming portion of the assignment, you will also be answering some questions about hash tables.

Part 1: Hash Map

First, complete the hash map implementation by using the skeleton code provided below. This hash map uses a hash table of buckets, each containing a linked list of hash links. Each hash link stores the key-value pair (string and object in this case) and a pointer to the next link in the list. Use the skeleton code below for your hash map implementation. For detailed instructions and method descriptions, please read this document:
CS261_Programming_Assignment_5_Instructions.pdf (https://canvas.oregonstate.edu/courses/1776054/files/80782698/download?wrap=1) **(https://canvas.oregonstate.edu/courses/1776054/files/80782698/download?download_frd=1)** very carefully. Use the provided tests in the skeleton file to help debug any failing test cases on GradeScope. You are encouraged to write additional unit tests, however, **you are not permitted to share any additional tests with other students**.

Skeleton code files: **hash_map.py (https://canvas.oregonstate.edu/courses/1776054/files/80783248/download?wrap=1)** **(https://canvas.oregonstate.edu/courses/1776054/files/80783248/download?download_frd=1)**

a5_include.py
ps://canvas.oregonstate.edu/courses/1776054/files/80818698/download?wrap=1)

(https://canvas.oregonstate.edu/courses/1776054/files/80818698/download?download_frd=1)
(updated)

Scoring (62 pts)

- clear() (5pts)
- get() (5pts)
- resize_table() (10 pts)
- put() (10 pts)
- remove() (10pts)
- contains_key() (6 pts)
- get_keys() (5 pts)
- empty_buckets() (6 pts)
- table_load() (5pts)

Part 2: Min Heap

In this part, complete the min heap implementation by using the skeleton code provided below. Here you will use a dynamic array to implement the complete binary tree heap in which the value in each internal node is smaller than or equal to the values in the children of that node. Use the skeleton code below for your min heap implementation. For detailed instructions and method descriptions, please read this document:
CS261_Programming_Assignment_5_Instructions.pdf
(https://canvas.oregonstate.edu/courses/1776054/files/80782698/download?wrap=1)
(https://canvas.oregonstate.edu/courses/1776054/files/80782698/download?download_frd=1) very carefully. Use the provided tests in the skeleton file to help debug any failing test cases on GradeScope. You are encouraged to write additional unit tests, however, **you are not permitted to share any additional tests with other students**.

Skeleton code files: **min_heap.py**
(https://canvas.oregonstate.edu/courses/1776054/files/80782820/download?wrap=1)
(https://canvas.oregonstate.edu/courses/1776054/files/80782820/download?download_frd=1)
(https://canvas.oregonstate.edu/courses/1776054/files/80783248/download?wrap=1)

Scoring (33 pts)

- add() (8 pts)
- get_min() (3 pts)
- remove_min() (10 pts)

- build_heap() (12 pts)

Part 3: Written Questions

In addition to the programming portion of the assignment, you will also be answering some questions about hash tables and hash functions. Refer to the hash functions in hash_map.py for the questions below. You will submit these questions in Canvas.

Question 1

Give an example of two words that would hash to the same value using hash_function_1 but would not be using hash_function_2. Explain why this is the case.

Question 2

Why does the above observation make hash_function_2 superior to hash_function_1?

Question 3

When you run your program on the same input file once with hash_function_1 and once with hash_function_2, is it possible for your empty_bucket() and table_load() functions to return different values? Why or why not?

Scoring (15 pts)

- Question 1 (5 pts)
- Question 2 (5 pts)
- Question 3 (5 pts)

What to Turn in

Submit your hash_map.py and min_heap.py (together with provided a5_include.py) files to Gradescope and submit your answers to the written questions on Canvas. You will get automated feedback from Gradescope which you can use to make changes and then resubmit your code as needed.

- Submit your hash_map.py, min_heap.py and a5_include.py files here: **Assignment 5: GradeScope Submission (Portfolio Assignment) (https://canvas.oregonstate.edu/courses/1776054/assignments/7961730)**
- Submit your written answers here: **Assignment 5: Canvas Submission (Portfolio Assignment) (https://canvas.oregonstate.edu/courses/1776054/assignments/7961729)**