College of Engineering
CS261

Programming Assignment 5

Heap Implementation of a ToDo List

CS261 Home
Schedule
Assignments
Resources
Policies
Grades

admin |
edit SideBar

In this assignment, you implement a **To-Do List application** with a **Heap-based Priority Queue**.

The application allows its users to manage their prioritized to-do lists using console commands for operations such as adding new tasks, retrieving, or removing the highest priority task. Additionally, users can save the list to a file or load the list from a file.

To complete the assignment, you must first complete the heap implementation of the priority queue interface. We will approach this new interface as we have approached previous interfaces such as stack, bag, and queue. In particular, we will simply add the priority queue operations to the dynamic array implementation. In addition, you'll use the 'compare' function for comparisons, as we did in the previous assignment (BST).

We have provided the functions to save and load the todo list to and from a file. These functions are in the `toDoList.[h,c]` files.

You must complete the `_buildHeap()` and `sortHeap()` functions to implement the heap sort algorithm in the `dynArray.c` file.

Refer to Worksheets 33 and 34 for more details regarding the heap implementation of the priority queue interface and the heapsort algorithm.

The provided files are described as below:

- dynArray.c: Implementation of dynamic array and heap-based priority queue. You will finish the functions for a heap-based priority queue in this file.
- dynArray.h: Header for `dynArray.c`. This file should not be changed.
- toDoList.c: Implementation of functions specialized for a to-do list application. NOTE: You'll need to use the strcpy(...) function to copy the description into your task structure.
- toDoList.h: Header for `toDoList.c`. This file should not be changed.
- type.h: Header file for `Task` structure. This file should not be changed.
- main.c: Implementation of the `main()` function, which controls the interactions between the user and the program. This file should not be changed. You can use it to test your functions.
- Makefile: The program's makefile.
- todo.txt: An example of a file storing a to-do list saved by function `saveList()`. Your `saveList()` function should generate a to-do list file with the same format. You can also use this file to test your `loadList()` function.
- program_demo.txt: Examples of command lines showing how a user can interact with the program. This file is provided for your reference.
- main2.c: Test file for the heap sort algorithm. You will need to change the `Makefile` and function names in `main.c` and `main2.c` accordingly to switch between testing the to-do list application and testing the heap sort algorithm.

CHALLENGE YOURSELF (AND EARN A FEW EXTRA POINTS!)

Challenge 1 - Generalize

The `dynArray` is restrictive in that it holds `TYPE` which is defined as a description and task. If we want to change what we store in the dynArr, we must modify TYPE and recompile. An alternative would be to make the dynArray store `*void` as we did with the BST in the last assignment.

For this challenge, generalize the dynArray implementation such that it can store any type (ie. *void) . You'll want to create a `structs.h` file where, as the user , you define what you want to put in the dynArray (in this case a Task).

Challenge 2 - Duplicate Priorities

For this challenge, imagine adding 10 tasks to the priority queue, all having the same priority value, say 1. Simulate this by hand. What order are they removed when you call `removeMinHeap` 10 times? Is this what you would expect? If not, modify your implementation to produce a more appropriate result when duplicate tasks are added.

RUBRIC

1. _adjustHeap 10
2. addHeap 10
3. getMinHeap 5
4. removeMinHeap 5
5. createTask 5
6. buildHeap 3
7. heapSort 3
8. smallerIndex 5
9. Compile/Style 20

WHAT TO SUBMIT

Note: All files with XC in the name are only necessary if you choose to address the posed Challenges.

1. dynArray.c
2. toDoList.c
3. dynArrayXC.c
4. dynArrayXC.h
5. typeXC.h