COP 3530

25/07/2022

# Project 3: Report

## I. Administrative

Team Name: Team Rex

Team Members: Gage Zahn, Marcus Lugrand, Benjamin Weiss

GitHub URL: https://github.com/marcuslugrand/Ante-Up-by-Team-Rex

Video: https://youtube.com/video/0YqKMsy0gD4/edit

## II. Refined Proposal

1) Problem: What problem are we trying to solve?
   a) Provide a tool that allows for Poker hand Research.
2) Motivation: Why is this a problem?
   a) When playing a game of poker, it is often advantageous to know what hands are possible and how good each hand is.
3) Features implemented
   a) A working GUI menu to select from:
      i) Find all hands of quality X
      ii) Find the quality of hand X
   b) Correctly executes menu options
   c) Displays a history of run times
   d) Displays card images of hands
   e) Runs on two different data structures
4) Description of data
   a) https://archive.ics.uci.edu/ml/datasets/Poker+Hand
   b) 1) S1 "Suit of card #1"
   c) 2) C1 "Rank of card #1"
   d) 3) S2 "Suit of card #2"
   e) 4) C2 "Rank of card #2"
   f) 5) S3 "Suit of card #3"
   g) 6) C3 "Rank of card #3"
   h) 7) S4 "Suit of card #4"
   i) 8) C4 "Rank of card #4"
   j) 9) S5 "Suit of card #5"
   k) 10) C5 "Rank of card 5"
   l) 11) CLASS "Poker Hand"
5) Tools/Languages/APIs/Libraries used
   a) C++, SFML
6) Algorithms implemented
   a) None
7) Additional Data Structures/Algorithms used

  a) Hash Map and Hash Tree

8) Distribution of Responsibility and Roles: Who did what?
  a) Ben: Hash Map
  b) Gage: SFML, Interaction and Display, Testing and Debugging
  c) Marcus: Tree Map

## III. Analysis

1) Change Log
  a) Data Structures changed from Hash Map and Hash Set to Hash Map and Tree Map
  b) Subtle changes to interface
    i) Added Run Button
    ii) Removed Data Structure Slider
  c) Feature Reduction
2) Complexity Analysis
  a) <u>map:</u>
    i) Function: Constructor
      (1) O(1) in all cases.
    ii) Function: insert
      (1) O(1) average and best case when defined by key, O(logn) is the worst case when key is not inset, where n is the amount of nodes
    iii) Function: Search by Hand
      (1) O(logn) best and average case, O(logn) is the worst case
    iv) Function: Search by Quality
      (1) O(logn) best, average, and worst case, where n is amount of node stored in the balanced BST.
    v) Function: Access
      (1) O(1) for all cases.
  b) <u>unordered_map:</u>
    i) Function: Constructor
      (1) O(1) in all cases.
    ii) Function: insert
      (1) O(1) average and best case, O(n) worst case if a rehash is triggered.
    iii) Function: Search by Hand
      (1) O(1) best and average case, O(n) worst case since open addressing is used.
    iv) Function: Search by Quality
      (1) O(1) all cases.
    v) Function: Access
      (1) O(1) best and average case, O(n) worst case.

## IV. Reflection

Although not everything went according to plan our group worked hard. Things became stressful near the end, and we got worried it may not be functioning by the due date. Fortunately, through many hours of labor, collaboration, and learning we were able to get a functioning project in time.

We ran into many issues with the maps. Getting the proper functionality to work and work efficiently enough to load in a reasonable time was often a challenge. There were times where our program took hours to load and open.

We would have liked to pick a project that had more real-world value. Unfortunately, it was difficult to find and obtain a data set that was both the required size, and practical.

Gage: I had never really understood how GitHub works before this project. I now feel like I understand enough to start keeping my projects on GitHub. I also learned how to store pairs in a hash map, and how to accept text input via SFML. I was able to get hands on experience debugging as a team. Since I oversaw main.cpp I was often the one that found the issues and had to report back with what is going wrong and why. This made it important that I learned proper communication so that the other members of my group could best do their jobs.

Ben: I think we worked well as group, sometimes the progress was stop-and-go but overall, the group dynamic was positive and productive. The group aspect of the project was great as I learned how to communicate and interact with team members in order to achieve a non-trivial goal. My responsibility was the implementation of the hash table data structure. I used a std::vector as the underlying representation of the table that used a power hash function and quadratic probing for the collision resolution policy. The client interface was a bespoke implementation of the hash map ADT that was optimized for our particular use case. While the course materials left me with a high-level understanding of the core concepts and implementing the data structure in the project allowed me to gain a deeper understanding of the core concepts of hashing, reducing, probing, etc. My initial implementation was rather slow, so if I could start again from scratch, I would spend more time in the planning phase of implementing the data structure. If we had more time, I would have liked to implement a full hash table API rather than just the functionality needed for the project, but I guess that will make for a good project over the coming semester break.

**Marcus:** My experience with this project was a roller coaster, but a good one. It had a lot of difficulties. For the tree map, I used an AVL tree that stored the vector of cards, the quality of those cards, and an individualistic calculation of the vector has a key. I referenced my Gator AVL project for inspiration. I was not as experienced as my teammates when it comes to programming but to make up for that, I buckled down and reinforced my coding knowledge. When it comes to teammates, I could not have asked for anyone better because Gage's implementation for the GUI was spot on and Benjamin knew what to do from the beginning for his hash map. Before this project I did not know how to use the debugger on Virtual Studio and my teammates helped me understand how to code and understand If a container has the correct values at a breakpoint. Implementing the tree map was tumultuous at best. It came down to the wire, but everything came together inevitably with everyone's hard work. I feel like with the help of my teammates, I really went beyond for my part and further enhanced my coding knowledge.

## V. References

1) https://archive.ics.uci.edu/ml/datasets/Poker+Hand

2) https://www.geeksforgeeks.org/how-to-create-an-unordered_map-of-pairs-in-c/
3) http://www.milefoot.com/math/discrete/counting/cardfreq.htm
4)