Advantages and disadvantages

Vector advantages

* Fast insertions at end of array
* Fast deletions at end of array

Vector disadvantages

* Slow searching

Hash table advantages

* Fast insertions
* Fast deletions
* Fast searching

Hash table disadvantages

* Not ordered

Binary tree advantages

* Useful for comparing elements in a less than / greater fashion

Binary tree disadvantages

* Slow insertion
* Slow deletion
* Slow searching

Data structure selection

Based on the pros and cons of each one, I think it’s fitting to use a double cache for this assignment. I’m making this decision for speed reasons specifically. I will be using a Hash Table and a Vector. Hash table for the fast selections, and a vector for the printing. I had considered using a binary tree for printing; however, the vector is much faster for loading. The vector can also be just as fast at printing if I cache the sorted array, I opted for not doing this under the assumption the list might not be fixed and might need to be updatable. Removing and sorting per update would just be insane.

A double cache has multiple tradeoffs though. Space and data consistency. Every time I update 1 cache, I must update the other. An additional draw back here is the 0(n)-removal process of the vector in the scenario the removal is in the middle of the vector. This tradeoff is worth it though for everything else.

This decision was strictly based on the choices at hand, in a perfect world I would use a Linked HashMap for this.

Time complexities for each function with the double cache system

* initCoursesFromPath() -> 0(n^2) -> insertions will be 0(1)
* loadCourse() -> 0(n)
* printCourse() -> 0(n) -> selection will be 0(1) and printing will be 0(n)
* printCourses() -> 0(n^2) unless everything is fixed, in which case it would be 0(n)