Vector sorting and printing would be good if we kept the classes and prereqs of the same size and amount we have now. It will be faster for the smaller course structures and amounts like this. It will be better for memory as well. The bad sides are since it's a vector if it's not sorted initially only on the print it will need resorted which could be tedious, or if adding new classes at the end of the vector it will need fully resorted again. And printing won't be alphanumeric unless its sorted.

The hash table would be good when adding new classes to the table. The search function would be fast and still posing only an O(n log n) when it goes about printing. But it would add extra pieces of data like temps to get it into a sortable option. It would use more memory. It also won't be “in order” for sorted data since hash can't just go through the table or walk it.

Binary search tree should already order. It has a default ordering with in-order traversal. We can add new information to it while keeping order in order. Inserting new information would be slightly slower as it traverses the tree, and due to the pointers, it would need more memory for the tree. The worst case will go to O(n) unless its balances.

Based on what I see this used for I would assume to use the HASH table unless we plan on loading and printing the list frequently. If it's a simple print, I would do it with HASH. If we must print frequently or sort it frequently, we would want to use TREE. The hash table will allow for very fast lookups where we need to lookup individual courses, average is O(1+k). Since the program initially will load the courses one, maybe add individual eventually, need to lookup frequently, and very infrequently print all of them it would offer the best overall balance to use HASH.

Vector

|  |  |  |  |
| --- | --- | --- | --- |
| Code | Line Cost | # Times Executes | Total cost |
| For All courses | 1 | n | n |
| If the course is the same as the courseNumber | 1 | n | n |
| For each prerequisite of the course | 1 | 1 | 1 |
| For each prerequisite of the course | 1 | n | n |
| Print the prerequisite course information | 1 | n | n |
| Total cost | | | 4n+1 |
| runtime | | | O(n) |

Hash

|  |  |  |  |
| --- | --- | --- | --- |
| Code | Line Cost | # Times Executes | Total cost |
| Search hash table | 1 | 1 | 1 |
| If the course is the same as the courseNumber | 1 | 1 | 1 |
| For each prerequisite of the course | 1 | k | k |
| For each prerequisite in table | 1 | k | k |
| Print the prerequisite course information | 1 | k | k |
| Total cost | | | 3k+2 |
| runtime | | | O(k) average |

BST

|  |  |  |  |
| --- | --- | --- | --- |
| Code | Line Cost | # Times Executes | Total cost |
| Search course | Log n | 1 | Log n |
| If the course is the same as the courseNumber | 1 | 1 | 1 |
| For each prerequisite of the course | 1 | k | k |
| For each prerequisite of the course | Log n | k | K\*log n |
| Print the prerequisite course information | 1 | k | k |
| Total cost | | | K\*log n+ k + log n+ 1 |
| runtime | | | O(k log n) |