Table of CPU time for different ADT

Haozhe Gu

999200555

ADT	FILE	TIME#1	TIME#2	TIME#3
Linked List	Fiel1	0.065	0.067	0.068
	File2	34.124	30.556	33.837
	File3	0.048	0.039	0.044
	File4	18.297	15.326	19.233
Cursor List	File1	0.054	0.061	0.059
	File2	182.497	181.236	180.688
	File3	0.063	0.059	0.061
	File4	91.324	90.236	91.758
Stack Ar	File1	0.059	0.057	0.058
	File2	0.048	0.049	0.050
	File3	0.051	0.050	0.052
	File4	0.057	0.056	0.058
Stack Li	File1	0.058	0.057	0.058
	File2	0.046	0.047	0.047
	File3	0.047	0.046	0.048
	File4	0.048	0.046	0.047
Queue Ar	File1	0.043	0.044	0.044
	File2	0.045	0.046	0.045
	File3	0.043	0.042	0.044
	File4	0.044	0.045	0.044
Skip List	File1	0.202	0.213	0.245

	File2	0.139	0.134	0.137
	File3	0.145	0..142	0.143
	File4	0.269	0.258	0.264
Individual insertion	Individual deletion	Entire series of insertions	Entire series of deletions	Entire file
Linked List				
File1:1 File2:1 File3:1 File4:1	Na N 1 N/2	N N N N	Na N^2 N N^2	N N^2 N N^2
Cursor List				
File1:1 File2:1 File3:1 File4:1	Na N 1 N/2	N N N N	Na N^2 N N^2	N N^2 N N^2
Stack Ar				
File1:1 File2:1 File3:1 File4:1	Na 1 1 1	N N N N	Na N N N	N N N N
Stack Li				
File1:1 File2:1 File3:1 File4:1	Na 1 1 1	N N N N	Na N N N	N N N N
Queue Ar				
File1:1 File2:1 File3:1 File4:1	Na 1 1 1	N N N N	Na N N N	N N N N
Skip List				
File1:logN File2:logN File3:logN File4:logN	Na logN logN logN	NlogN NlogN NlogN NlogN	Na NlogN NlogN NlogN	NlogN NlogN NlogN NlogN

Analysis of the CPU time of different File

Linked list:

The cost of the insertion and deletion of linked list are all O(1), which means it takes constant time to complete the insertion and deletion. However, this situation will change depend on whether we need to move through the list while do the insertion and deletion. So this is also the reason why the big o notation for the four data files are different. In the implement, I choose to insert at the head of the list, which make file1 takes constant time. So the big o notation of file1 should be also be constant time O(1).then if we need to insert 250000 at the head, the overall would be O(N)

File2 has 125000 head insertions in order 1 to 125000 but the deletion of file 2 follow the same order 1 to 125000. Which means every time you delete a value in the list, you need to go through the whole list. This make a simple constant time deletion become O(N) and thus O(N^2) for the overall program. Because of this, the cpu time of File2 become much larger for that of File1.

File3 has 12500 insertion in order 1 to 125000 and deletion in the reverse order 125000 to 1. This means this time we don't need to go through the whole list to complete one single deletion. These are based on that we use the head insertion, if we change to tail insertion, the situation would be change completely. So our overall big o notation for File3 is O(N)

File4 has 12500 insertion in random order and 12500 deletions also in random order. This means whatever we use head insertion or tail insertion, we need to move in single deletion. But different from the case in File2, the position we need to move is random. Thus the average position we need to move is N/2 in one single deletion. So the overall big o notation would be O(N^2)again, however, we have half of the time in case File 2 because we move less this time, actually half the time. The CPU timer also agree with our conclusion. For the three test, File4 time always closed to half of that for file2. Also, in this case, we have the same time whether we use head or tail insertion.

Also notice that the time needed for file1 and file3 is much less that need for file2 and file4. This is the difference come from the O(N) and O(N^2)

Cursor List:

We know that CursorList is very similar to the linked list. Actually just the array form. It still keep the property that have O(1) for the single insertion and deletion. Because we still use the head insertion, so file 1 and file3 is still much faster than file2 and file4 because the way they are deleted. Also the cpu time for file2 is roughly twice of that file4 required because of the random insertion and deletion.

Stack Ar:

Because of the deletion in stack (which is pop) ignores the actual value for the deletion command. So, this time, O(1) is the complexity of the single deletion and insertion in all four files. Also, the overall complexity would be O(N). Our CPU time also agree with our conclusion. All four file time are roughly the same.

Stack Li:

This one is very similar to the array implement of stack. The only difference is that we notice that file1 have slightly more CPU time than File2 3and File4. That is because the insertion take slightly longer time than deletion in this case.

Queue Ar:

Similar to the stack

SkipList:

This one is pretty straight forward. Because of the binary search to insert and delete. Both the insertion and deletion are no longer O(1) but actually O(logN). Also because of the log, the file1 will take more time than other three. Because It does 250000 insertion in a row, this will take more time than 125000 insertion and 125000 deletion.

Analysis between ADT

The three ADT stacklist stackarray and queue array are pretty much the same concept and they are all very fast at insertion and deletion. But we notice Cursor List take much longer time for file2 3 4 than the Linked list except for file1.This might because the cursor list have better ability to insert, or in other word create, allocate new nodes. But the linked list surely good at deletion. Also, the reason why cursor list and linked list delete and insert slower than the normal list is because list just create a whole array of int and

this is easy to change without allocate lots of memory. However, linked list and cursor list allocate extra memory to change the int. Also, notice the Skip List. It might be true that other ADT like linked list and cursor list may take less CPU time to do things in file1 and file4. But file2 and file4 represent two bad situation that illustrate that the order to insert and delete in this two ADT can affect the CPU time dramatically. On the other side, SkipList don't have such problem, in other words SkipList have better average CPU time and the order of deletions and insertion will not affect much.