Question Statement:

Study memory reallocation algorithm efficiency for a dynamic table, based on different values of increasing and decreasing factors.
Find the statistics for different ratios of inserts and deletes and find best statistical measure.
Increasing factors: 1.25x, 1.5x, 1.75x, 2x, 3x.
Decreasing factors: 0.25x, 0.5x, 0.75x.

Approach:

1) C Structures as basic layout for dynamic table.
2) Variable length array implemented using dynamic allocation to a integer array pointer.
3) Variable length array member of dynamic table structure.
4) Sequence of operations generated using Rand library of C.
5) Insertions are assured before deletions by compensating a present delete with a future insert.

Findings:

1)For 1:1 insert to delete ratio, 0.5x decrease factor with 2x increase factor is optimal.
2)For 3:2 insert to delete ratio, 0.75x decrease factor with 1.75x increase factor is optimal.
3)For 4:2 insert to delete ratio, 0.25x decrease factor with 1.5x increase factor is optimal.
4)On average 1.75x and 1.5x increasing factors, gave the best average efficiency.
5)On average 0.5x decreasing factor, gave the best average efficiency.
6)If there are more insertions, its optimal to have lower decreasing factor and higher increasing factor.
7)If there are more deletions, its optimal to have higher decreasing factor and lower increasing factor.
8) Optimal ratios of reallocation factors usually follow the ratio of insertions to deletions.

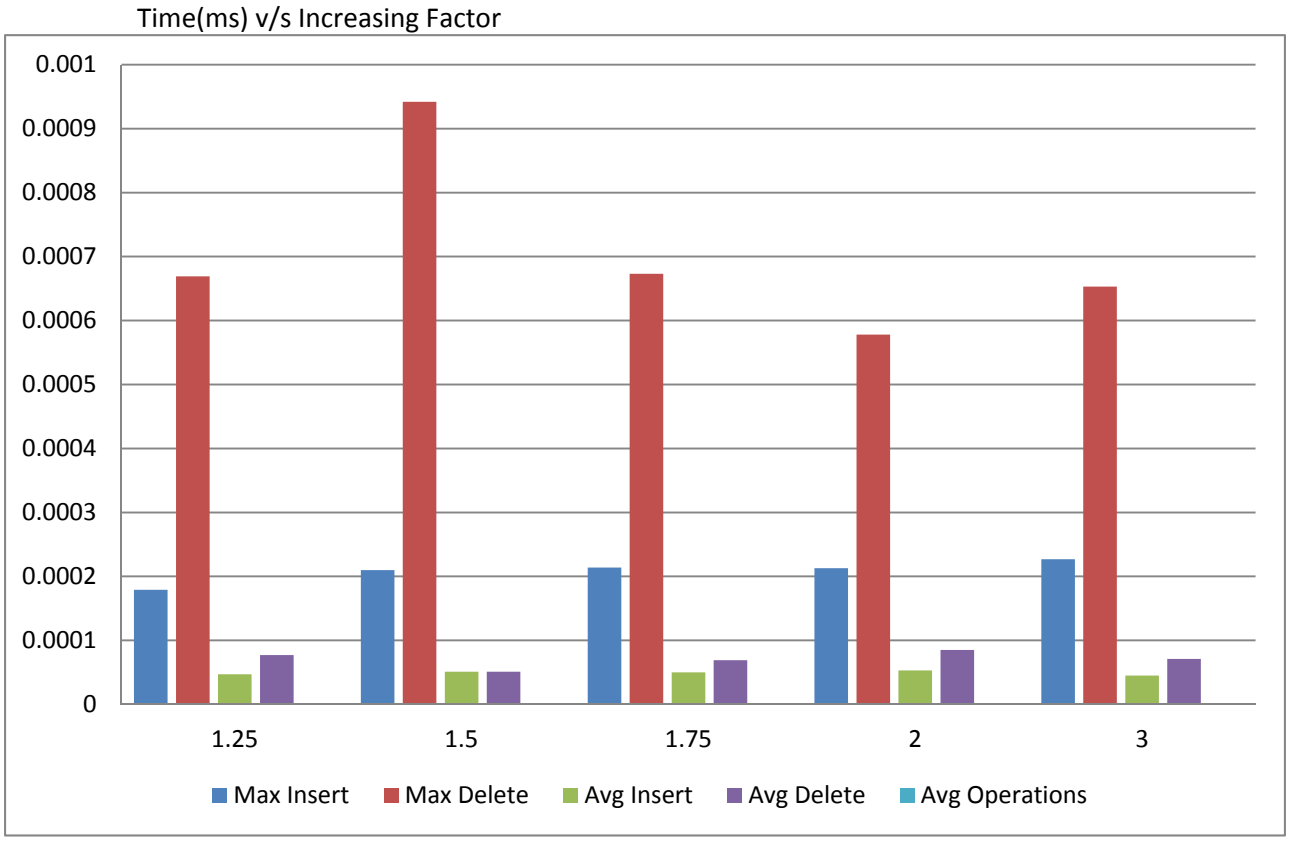
Points understood:

1)On resizing, pointers, references and iterators change. Therefore they must be reassigned, and reallocated memory.
2) Realloc() in C dynamic memory allocation does not promise contiguous memory,and return of same pointer and thus separate copying of older data must be done after reallocation to the dynamic table.
3) Since C does not have implicit garbage collection, we cannot surely know if a pointer is valid or not.
4) Memory allocation in C is non-determinitic, and may be fragmented or contiguous.
5)To find out if a pointer points to garbage, we can use another pointer to point to it, and dereference it to see if it's value is NULL. If malloc NULL, that implies dynamic allocation has failed and heap is filled.
5)Efficient application memory management can help us track if a pointer points to garbage.
6)It is good practice to assign NULL to a pointer after it is free'd. Free() does not return any value to assure it has free'd the memory correctly.
7)Memory fragmentation does not occur implicitly in C. Malloc,realloc and calloc return NULL if a contiguous block of memory cannot be assigned for the given size asked.
8)Memory pools can be created beforehand by the user (blocks of memory of sizes that can be potentially required by the user later on).Malloc can then be programmed to choose from this pool.
9)The allocated pointers point to the first element of the block of memory. There is also another pointer initialized implicitly for book-keeping and tracking the memory block that is allocated on the heap.

Insert : Delete = 1:1

Average Insertion time and Average deletion time are almost equal for **1.5 increase factor with 0.25 decrease factor**. Compared to other decrease factors, this gives better statistical measure for all operations since,the size of the dynamic table is sufficient for most of the operations and reallocation for enlargement is very rare. This is at the cost of **large memory retention**.

1:1 ratio	Increase Factor				
Decrease Factor					
0.25	1.25	1.5	1.75	2	3
Max Insert	0.000179	0.00021	0.000214	0.000213	0.000227
Max Delete	0.000669	0.000942	0.000673	0.000578	0.000653
Avg Insert	0.000047	0.000051	0.00005	0.000053	0.000045
Avg Delete	0.000077	0.000051	0.000069	0.000085	0.000071
Avg Operations	0.000001	0.000001	0.000001	0.000001	0.000001

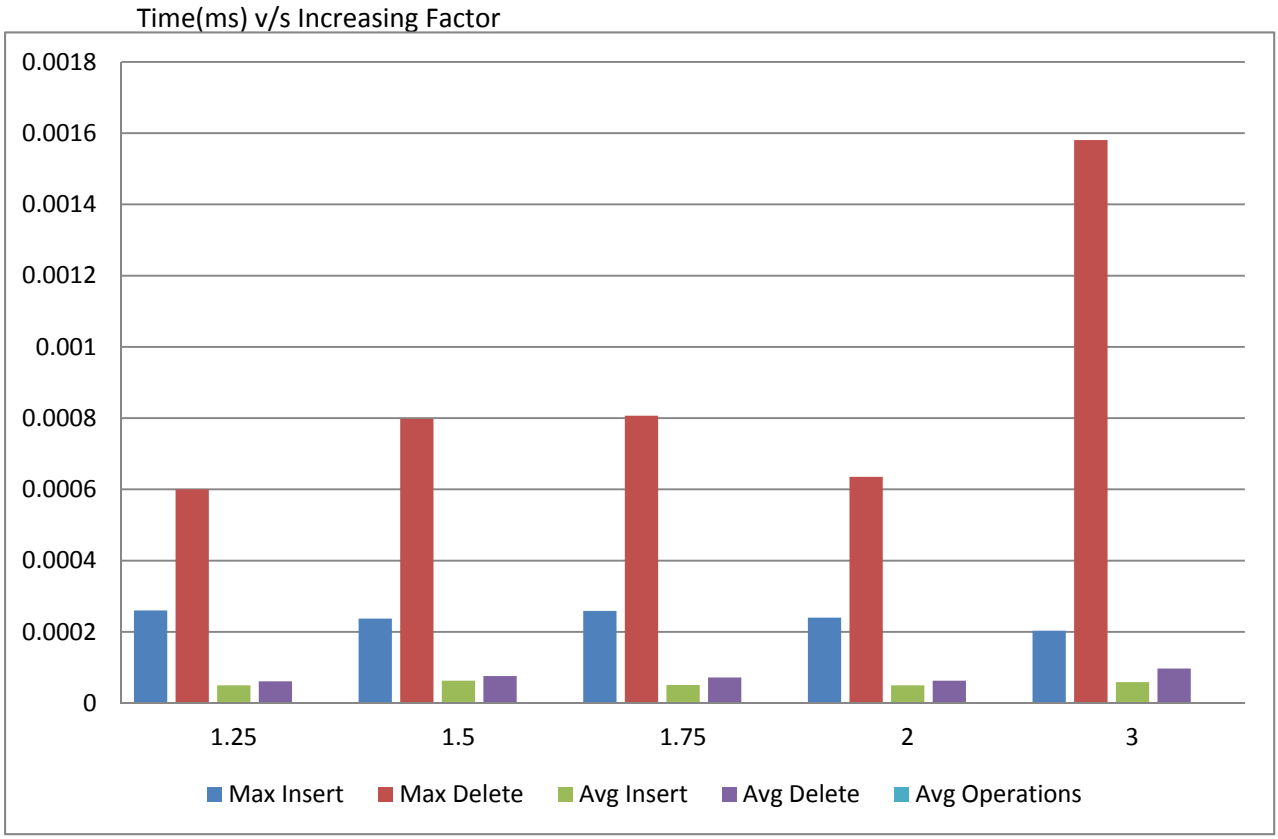


Assignment 1
UE15CS311
Rutha Prasad
01FB15ECS367

Insert : Delete = 1:1 (optimum)

Average Insertion time and Average deletion time are almost equal for all ratios of increase factor with decrease factors.Therefore this ratio might be best for 1:1 insertion and deletion probability. Compared to other decrease factors, this gives **best average statistics** since large memory is also not retained.**2 increase factor with 0.5 decrease** is best.

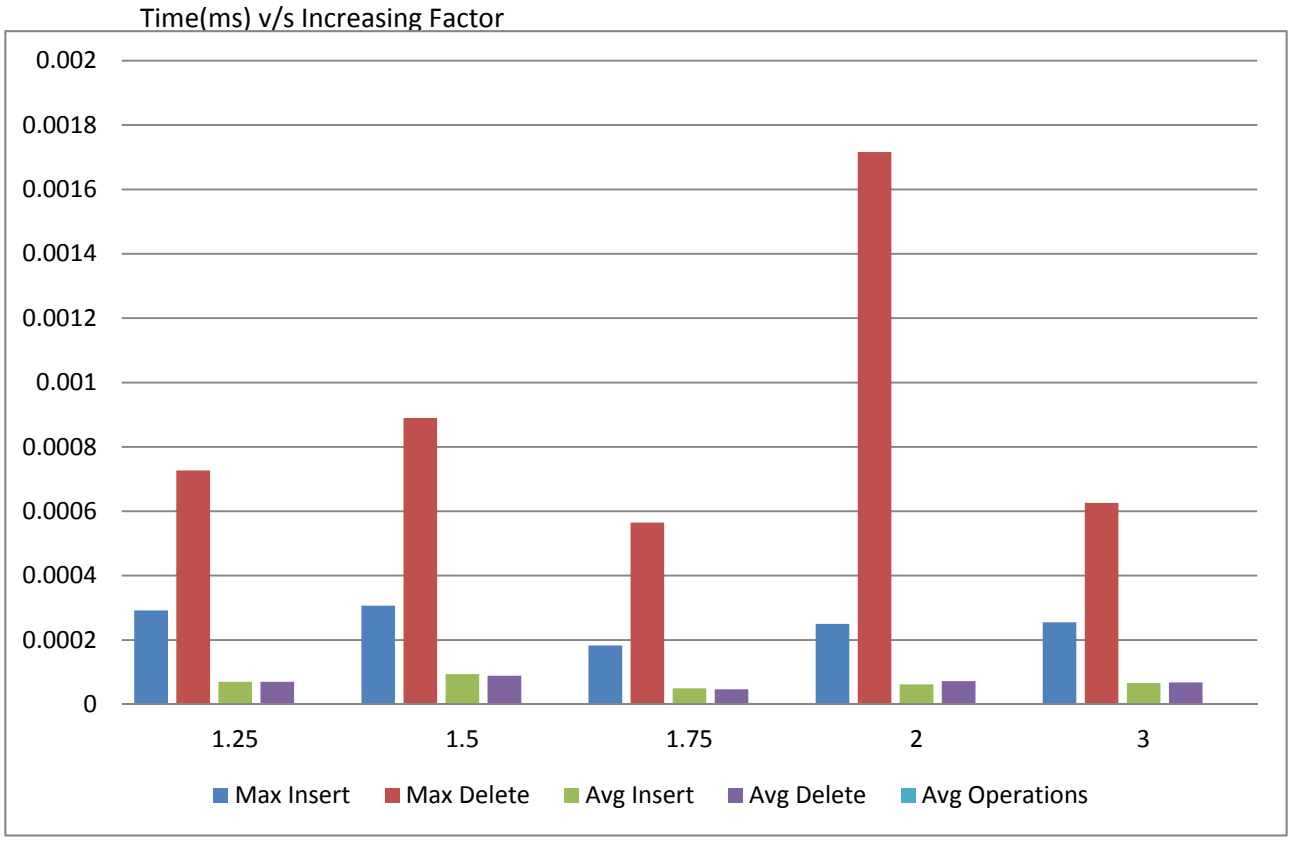
1:1 ratio	Increase Factor				
Decrease Factor					
0.5	1.25	1.5	1.75	2	3
Max Insert	0.00026	0.000237	0.000259	0.00024	0.000203
Max Delete	0.0006	0.000798	0.000807	0.000635	0.001581
Avg Insert	0.00005	0.000063	0.000051	0.00005	0.000059
Avg Delete	0.000061	0.000076	0.000072	0.000063	0.000097
Avg Operations	0.000001	0.000001	0.000001	0.000001	0.000002



Insert : Delete = 1:1

Average Insertion time and Average deletion time are almost equal for all ratios of increase factor with decrease factors.Compared to other decrease factors, this initiates **more memory reallocations** during inserts, this **slowing down insertion**. But **1.75 increase factor with 0.75 decrease factor** gives the least memory strain.

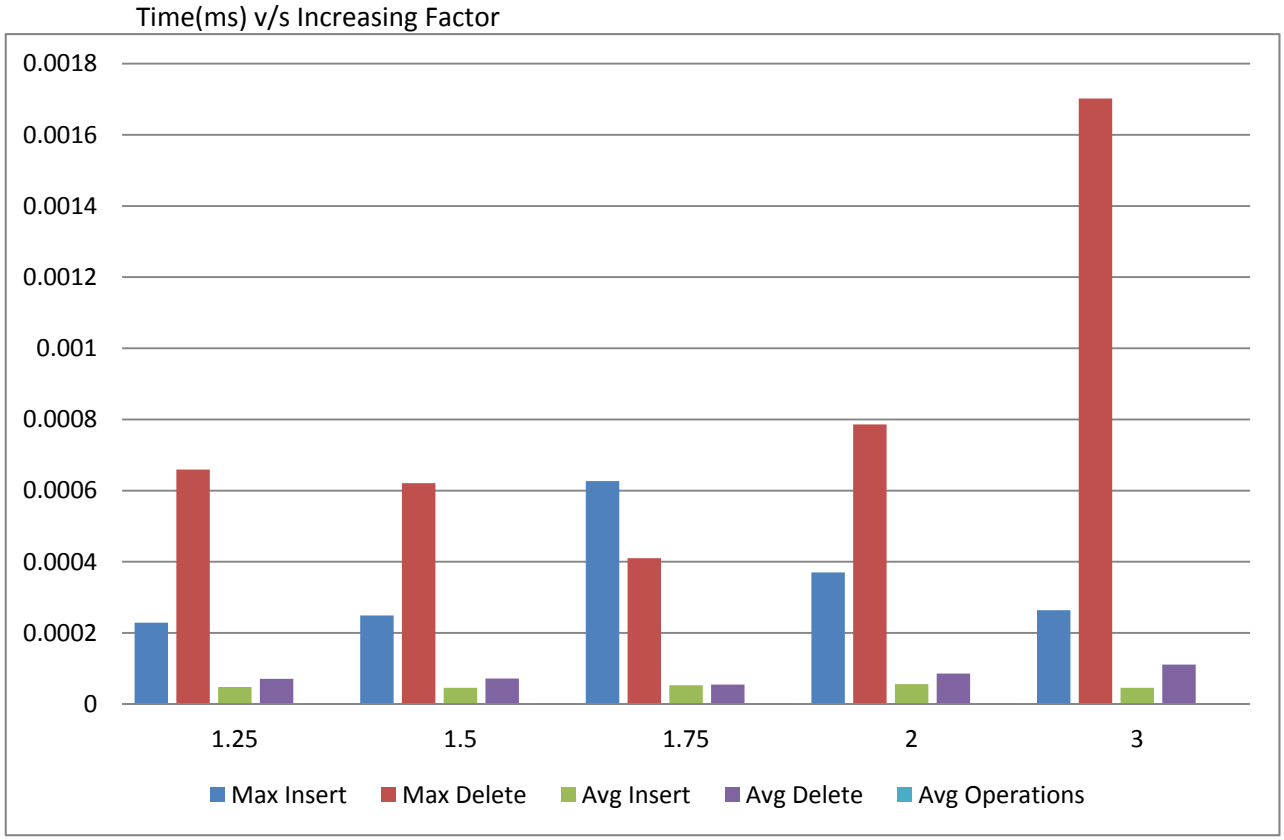
1:1 ratio	Increase Factor				
Decrease Factor					
0.75	1.25	1.5	1.75	2	3
Max Insert	0.000292	0.000307	0.000183	0.00025	0.000255
Max Delete	0.000727	0.00089	0.000565	0.001716	0.000626
Avg Insert	0.00007	0.000094	0.00005	0.000062	0.000066
Avg Delete	0.00007	0.000089	0.000047	0.000072	0.000068
Avg Operations	0.000001	0.000002	0.000001	0.000001	0.000001



Insert : Delete = 3:2

Average Insertion time and Average deletion time are almost equal for **1.75 increase factor with 0.25 decrease factor**. Compared to other decrease factors, this gives better statistical measure for all operations since,the size of the dynamic table is **always sufficient for the insertions** and reallocation for enlargement is very rare, but **memory is wasted** since deletes are also frequent.

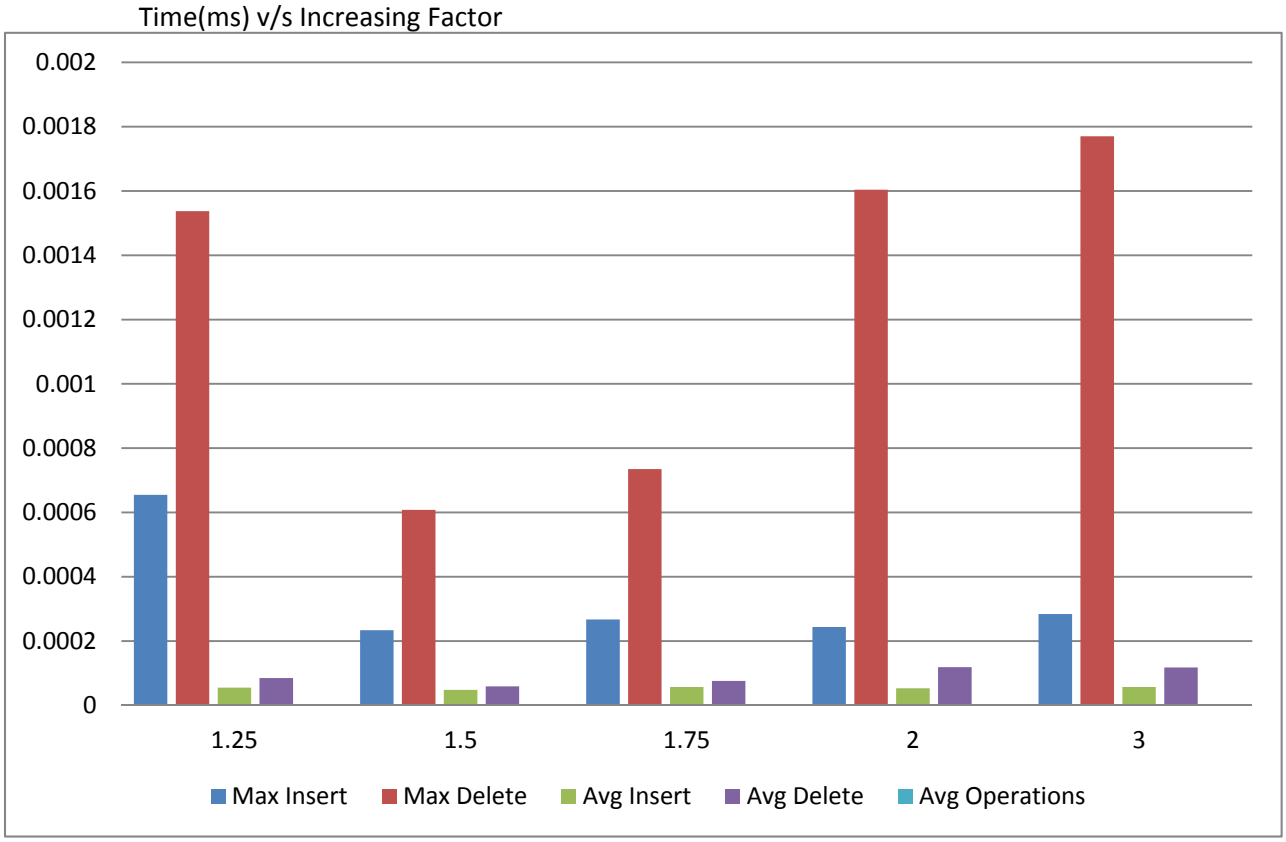
3:2 ratio	Increase Factor				
Decrease Factor					
0.25	1.25	1.5	1.75	2	3
Max Insert	0.000229	0.000249	0.000627	0.00037	0.000264
Max Delete	0.000659	0.000621	0.00041	0.000786	0.001702
Avg Insert	0.000048	0.000046	0.000053	0.000056	0.000046
Avg Delete	0.000071	0.000072	0.000055	0.000086	0.000111
Avg Operations	0.000001	0.000001	0.000001	0.000001	0.000002



Insert : Delete = 3:2

Average Insertion time and Average deletion time are almost equal for all ratios of increase factor with decrease factors.**1.5 increase factor with 0.5 decrease** is best.

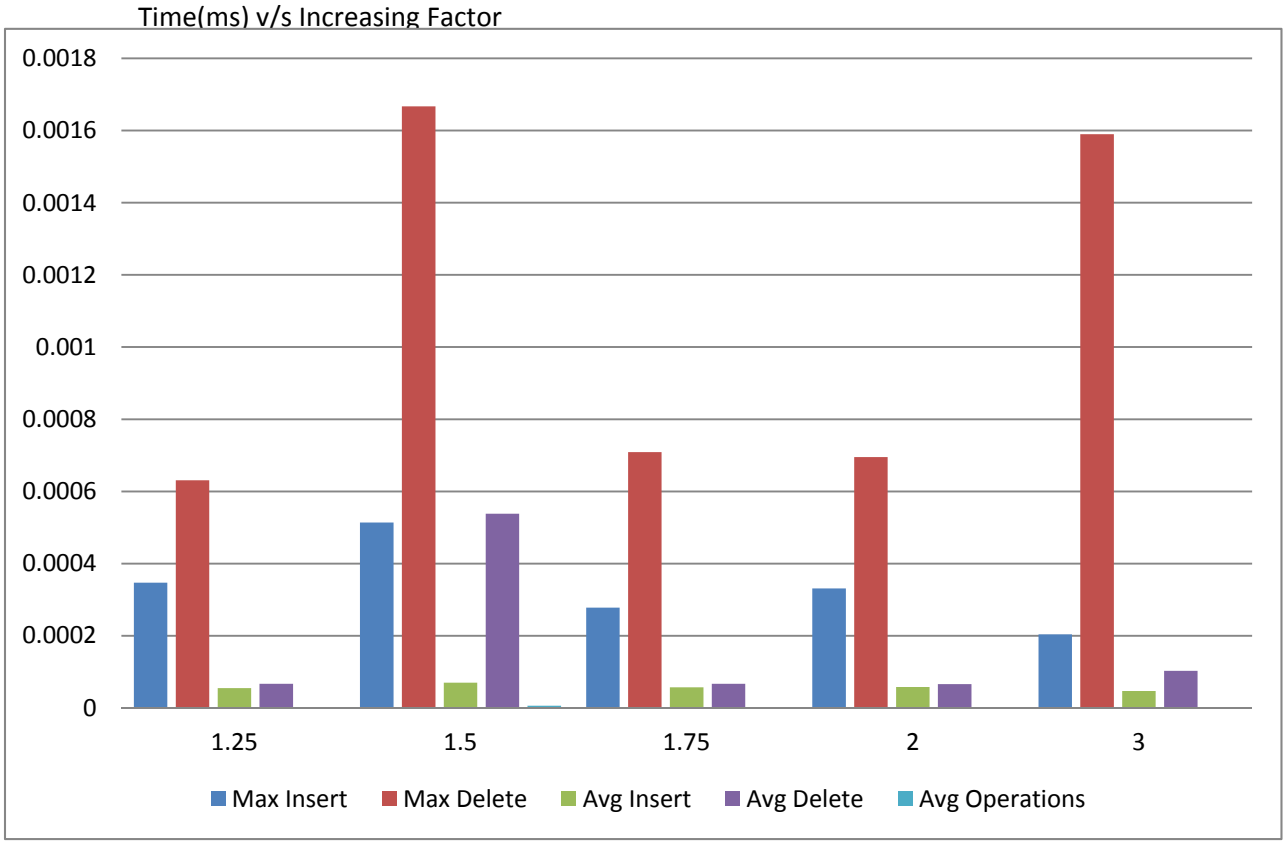
3:2 ratio	Increase Factor				
Decrease Factor					
0.5	1.25	1.5	1.75	2	3
Max Insert	0.000655	0.000234	0.000267	0.000244	0.000284
Max Delete	0.001537	0.000608	0.000735	0.001604	0.00177
Avg Insert	0.000055	0.000048	0.000057	0.000053	0.000057
Avg Delete	0.000085	0.000059	0.000076	0.000119	0.000118
Avg Operations	0.000001	0.000001	0.000001	0.000002	0.000002



Insert : Delete = 3:2 (optimum)

Average Insertion time and Average deletion time are almost equal for **1.75 increase factor with 0.75 decrease factors**.Compared to other decrease factors, this initiates **more memory reallocations** during inserts, but **memory is not wasted** as there are **almost equal number of deletes**.

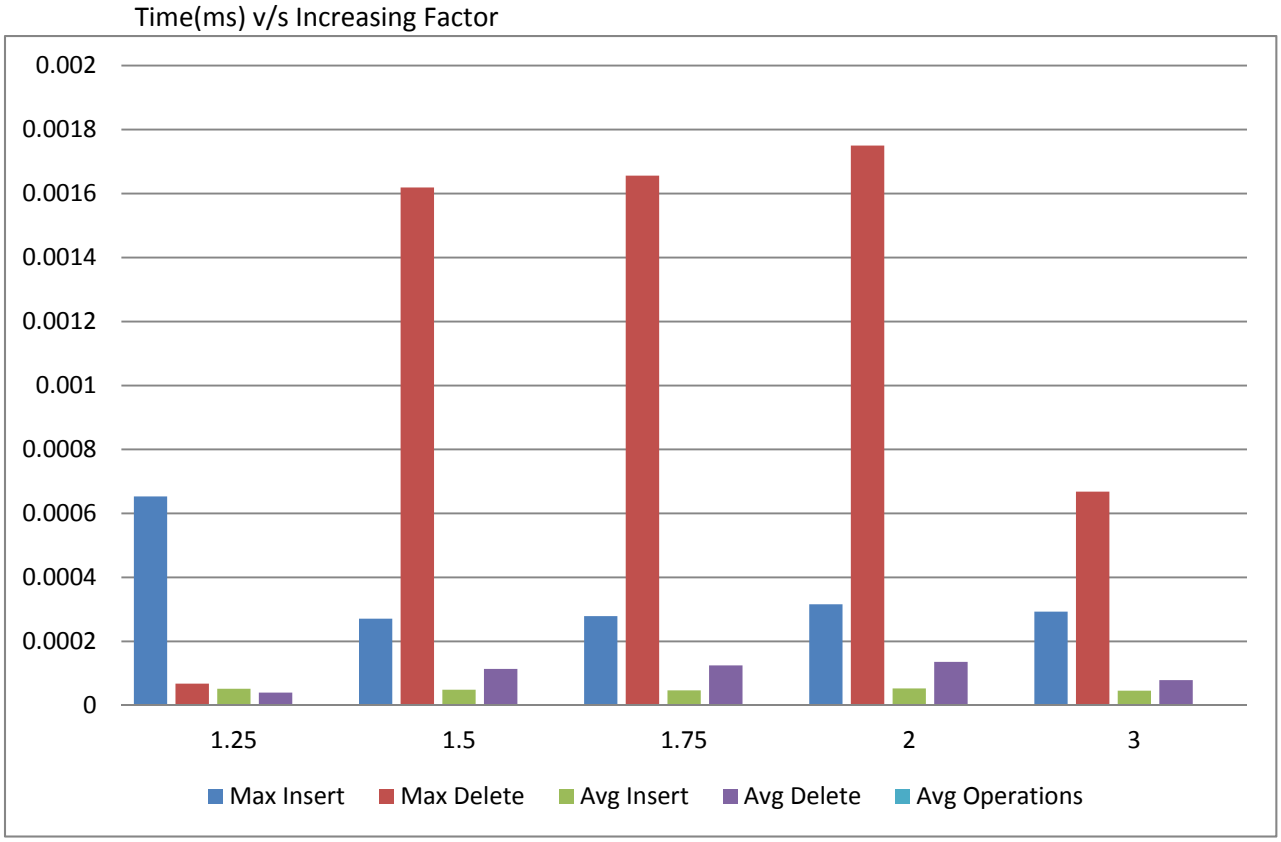
3:2 ratio	Increase Factor				
Decrease Factor					
0.75	1.25	1.5	1.75	2	3
Max Insert	0.000347	0.000514	0.000278	0.000331	0.000204
Max Delete	0.000631	0.001667	0.000709	0.000695	0.00159
Avg Insert	0.000055	0.00007	0.000057	0.000058	0.000047
Avg Delete	0.000067	0.000538	0.000067	0.000066	0.000103
Avg Operations	0.000001	0.000006	0.000001	0.000001	0.000001



Insert : Delete = 4:2 (optimum)

Average Insertion time and Average deletion time are almost equal for **1.5 increase factor with 0.25 decrease factor**. Compared to other decrease factors, this gives better statistical measure for all operations since,the size of the dynamic table is **sufficient for the excess insertions** and reallocation for enlargement is very rare.Even **max insertions time for all increase factors are almost equal**.The max delete is larger than for other decreasing factors but since **deletions are less frequent**, it is **amortized**.

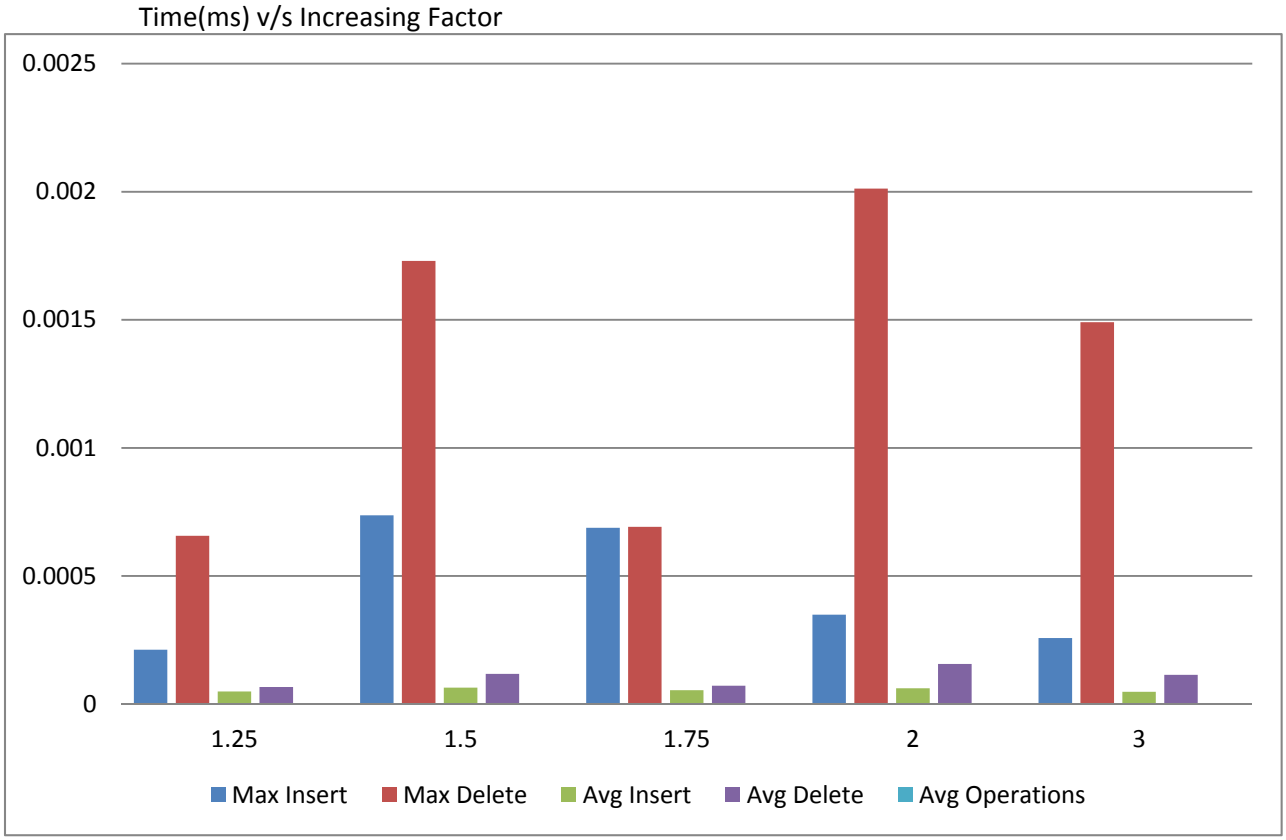
4:2 ratio	Increase Factor				
Decrease Factor					
0.25	1.25	1.5	1.75	2	3
Max Insert	0.000653	0.000271	0.000279	0.000316	0.000293
Max Delete	0.000068	0.001619	0.001656	0.00175	0.000668
Avg Insert	0.000052	0.000049	0.000047	0.000053	0.000046
Avg Delete	0.00004	0.000114	0.000125	0.000136	0.000079
Avg Operations	0.000001	0.000002	0.000002	0.000002	0.000001



Insert : Delete = 4:2

Average Insertion time and Average deletion time are almost equal for all ratios of increase factor with decrease factors. Compared to other decrease factors, this gives **best average statistics** since **large memory is also not retained**.But **average insertion take more time**. 1.75 **increase factor with 0.5 decrease** is best.

4:2 ratio	Increase Factor				
Decrease Factor					
0.5	1.25	1.5	1.75	2	3
Max Insert	0.000212	0.000737	0.000688	0.000349	0.000258
Max Delete	0.000657	0.00173	0.000692	0.002012	0.001491
Avg Insert	0.000049	0.000064	0.000054	0.000062	0.000048
Avg Delete	0.000067	0.000118	0.000072	0.000157	0.000114
Avg Operations	0.000001	0.000002	0.000001	0.000002	0.000002



Insert : Delete = 4:2

Average Insertion times are more than Average deletion times.Compared to other decrease factors, this initiates **more memory reallocations** during inserts, this **slowing down insertion**. But **2 increase factor with 0.75 decrease factor** gives the least memory strain.

4:2 ratio	Increase Factor				
Decrease Factor					
0.75	1.25	1.5	1.75	2	3
Max Insert	0.000321	0.000695	0.000225	0.000267	0.000782
Max Delete	0.000701	0.000087	0.001507	0.000596	0.001779
Avg Insert	0.000059	0.000053	0.000051	0.00005	0.000072
Avg Delete	0.000032	0.000037	0.000071	0.000066	0.00014
Avg Operations	0.000001	0.000001	0.000001	0.000001	0.000002