HW2
Name: Xiang Gu
EID: xg2847

Part A:

8.4.1

Action	No Index	Start Index	Movie Index	Both Index
Q1	100	4	100	4
Q2	100	100	4	4
I	2	4	4	6
Average	2+98p1+98p2	4+96p2	4+96p1	6-2p1-2p2

8.4.2

Action	No Index	Name Index	Class Index	Launched Index	Name, Class Index	Name, Launched Index	Class, Launched Index	All Index
Q1	25	2	25	25	2	2	25	2
Q2	25	25	2	25	2	25	2	2
Q3	50	50	50	26	50	26	26	26
I	2	4	4	4	6	6	6	8
Average	2+23p1+23p2+48p3	4-2p1+21p2+46p3	4+21p1-2p2+46p3	4+21p1+21p2+22p3	6-4p1-4p2+44p3	6-4p1+19p2+20p3	6+19p1-4p2+20p3	8-6p1-6p2+18p3

14.1.1

Dense Index: n/3 + n/10

Sparse Index: n/3 + n/30

14.2.1

(b) We need 100,000 data blocks. Starting from bottom to top, if each leaf node in the B+ tree structure is assumed to have 70 pointers then we need 1,000,000/70=14,286 leaf nodes. Then we need 14,286/70=204 blocks in the second layer, and 204/70=3 blocks in the third layer, and finally 1 block for the root.

Hence, we need 100,000 + 14,286 + 204 + 3 + 1 = **114,494** blocks.

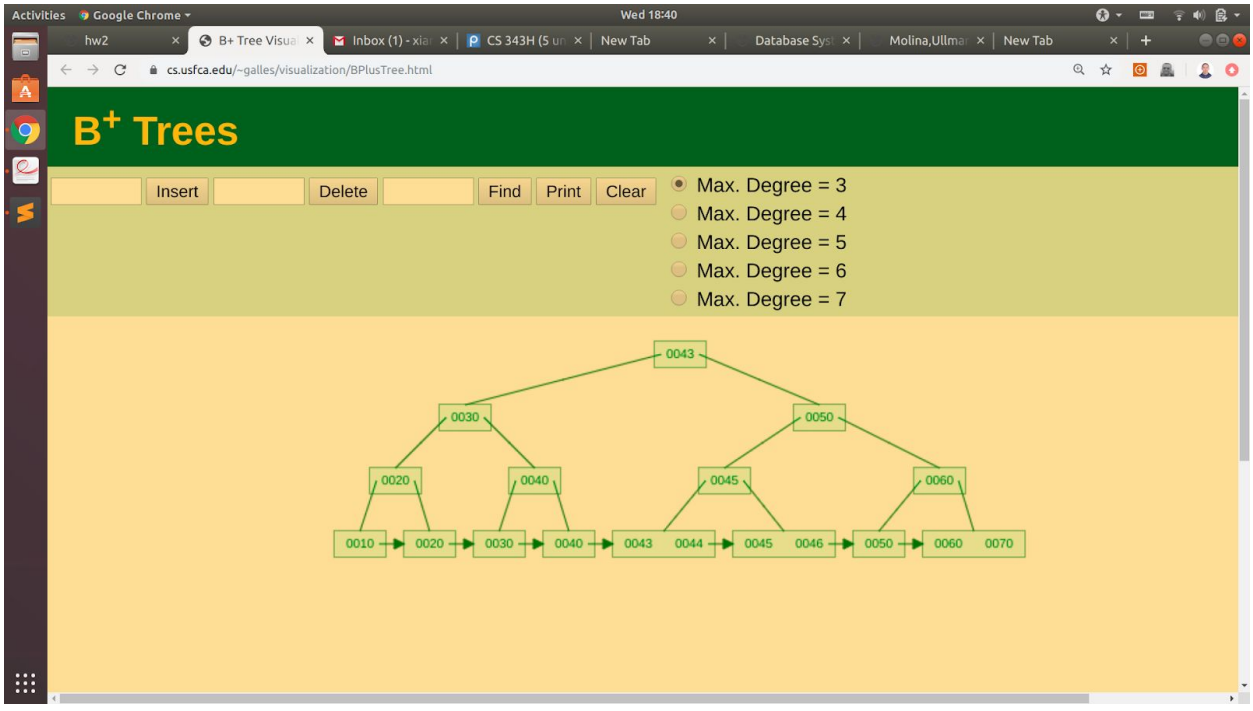
Since this B+ tree has 4 layers, we need 4 + 1 = **5** disk I/O's to get the record.

(c) We again need 100,000 data blocks. But now since our B+ tree is a sparse index, the number of leaf nodes we need decreases to 100,000/70=1429. Then in the second layer we need 1429/70=20 blocks and 1 block in the third layer as the root.
Hence, we need 100,000 + 1429 + 20 + 1 = **101,450** blocks.
Now the B+ tree has only 3 layers so we need 3 + 1 = **4** disk I/O's to get the record.

14.2.2
(b) Same as above, we need **114,494** blocks for the data file and the B+ tree index.
But now after we go down to the leaf node with the lower bound of the range query, we need to sequentially read 1000 records for all the matching records. Since the data is not sorted we need to read in one block by following the 1st pointer in the leaf node for the 1st record, moves to the next pointer and read in another whole block for the second record. After we finished reading all 70 blocks by following those 70 pointers in this leaf, we move to the next sibling leaf and continue. Thus, the total number of disk I/O's is 3 + (1+70) * 14 + (1+20) = **1018** disk I/O's.

(c) Same as above, we need **101,450** blocks.
But now, our B+ tree is a sparse index which means everytime we bring in a block from disk by following a pointer in the leaf, we can get 10 records and we move to the next pointer and get another 10 records. Hence, the total number of disk I/O's is 3 + (1+70) + (1+30) = **105** disk I/O's.

Part B:
My sequence of insertion, after 10, 20,..., 50, is **60, 70, 45, 46, 44, 43**.
Final Visualization:



Visualization Before Inserting the Last Element (43):