Homework 1 (20 pts)

General Instructions

- Due on 10/18 (Tue), 11:59PM
  - Late submission within one week: penalty = 7pts
  - Late submission within two weeks: penalty = 14pts
  - After two weeks: no credits will be given
  - Late submissions will not get any bonus credit
- If you cannot finish all, submit solutions for some problems to get partial credits.
- Only submit source files. Do NOT include any executables. All files should be packed into a single zip file and be submitted via blackboard.
- Ensure your code can be compiled and executed in command line (not a java IDE). Otherwise, you will NOT get any credits.
- In the zip file, include a text file: README.txt. Write down which problems have you finished. Also write down on which platform (mac, linux, window) is the code compiled and executed.
- You are NOT allowed to use any native implementation of lists (ArrayList, LinkedList, etc.). Consult the instructor if you want to use any native class that is not mentioned here.
- This is NOT something you can finish in three days. To understand the problem itself takes quite some time. You have to start as early as possible.

Problem 1: Sparse Vector Using Linked List (7 pts)

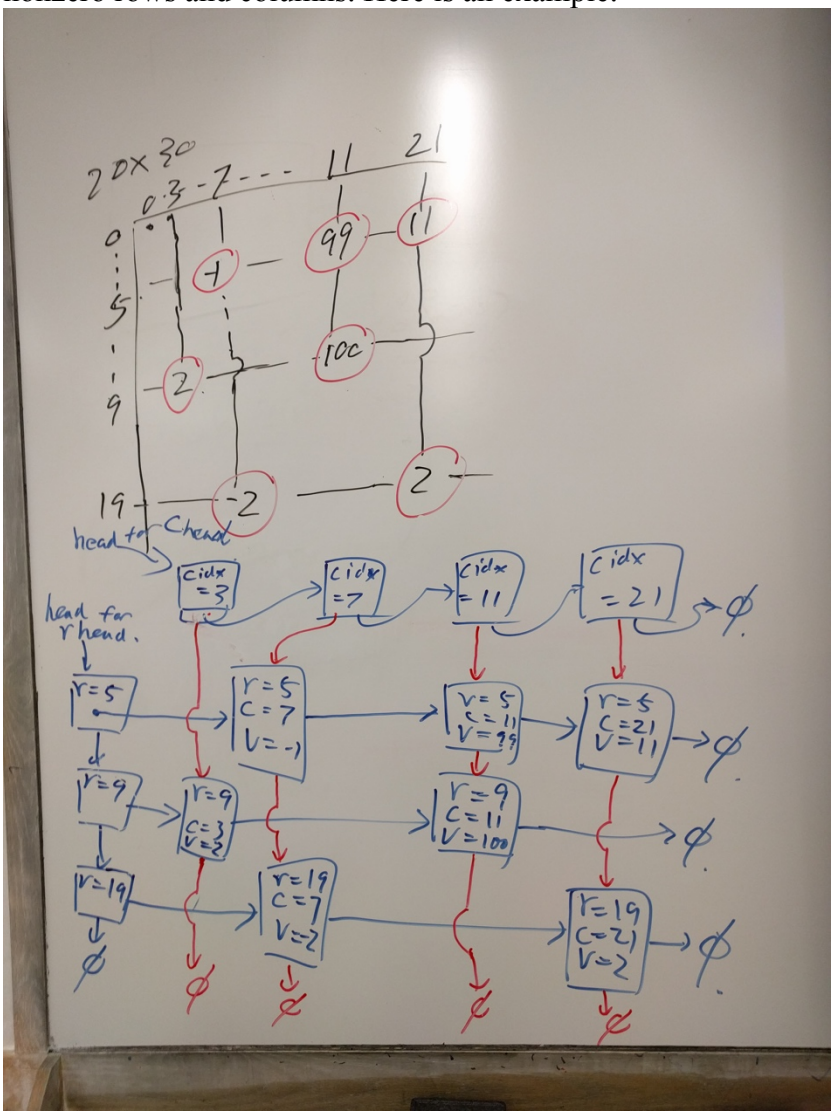
- Implement the linked list sparse vector class (LLSparseVec.java) so that LLMainClass can be executed.
- Nodes in the linked list are nonzero elements of the vector, sorted according to their index (refer to the slide in Lecture 2 – part 2).
- Implement the constructor, access methods, getElement, setElement, clearElement, getAllIndices, getAllValues. In otherwords, when LLMainClass is called using VEC argument and with a single input file, the program should be able to run correctly and give the same output as ArrayMainClass.

Problem 2: Sparse Vector Operation Linked List (7 pts)

- Implement the addition, substraction, and multiplication methods in LLSparseVec.
- The algorithm has to be O(m), in which m is the maximum number of nonzero elements in a vector. To achieve this, you cannot simply use get and set in Problem 1. Only algorithmes with O(m) complexity will get credits.
- All operations return a new sparse vector object, storing the result. The method substraction(otherV) means the current vector minus otherV.
- If the two vectors' length do not match, return a null object.
- When LLMainClass is called using VEC argument and with multiple input files, the program should be able to run correctly and give the same output as ArrayMainClass.

Problem 3: Sparse Matrix Using Linked List (6 pts)

- Implement the linked list sparse matrix class (LLSparseMat.java) so that LLMainClass can be executed with MAT argument.
- Element nodes are nonzero elements of the matrix. RowHead and ColHead nodes are nonzero rows and columns. Here is an example:



- Recommended different types of nodes (not mandatory, you can define your own nodes as you like):
  - An element node stores the following information: row index, column index, value, next element in the same row, next element in the same column;
  - A row head node stores the following information: row index, next nonzero row, the first element in this row;
  - A column head node stores the following information: column index, next nonzero column, the first element in this column.
- Implement the constructor, access methods, getElement, setElement, clearElement. Also implement the methods to get the indices of all nonzero rows and all nonzero columns: getRowIndices, getColIndices. Implement methods to access a single row or a single

column: getOneRowColIndices, getOneRowValues, getOneColRowIndices, getOneColValues. NOTE that these methods should all be linear to the number of nonzero elements in the row/column of interest (simply find the corresponding row/col head, follow the pointers through all elements of this row/column).

- When LLMainClass is called using MAT argument and with a single input file, the program should be able to run correctly and give the same output as ArrayMainClass.

Problem 4: Sparse Matrix Operation Linked List (6 bonus points)

- Implement the addition, substraction, and multiplication methods in LLSparseM.
- The algorithm has to be O(m), in which m is the maximum number of nonzero elements in a matrix. To achieve this, you cannot simply use get and set in Problem 3. Only algorithmes with O(m) complexity will get credits.
- All operations return a new sparse matrix object, storing the result. The method substraction(otherM) means the current vector minus otherM.
- If the two matrices' dimensions (nrows, ncols) do not match, return a null object.
- When LLMainClass is called using MAT argument and with multiple input files, the program should be able to run correctly and give the same output as ArrayMainClass.