COMP9024 (20T0)

- **Assignment : How to Implement?**

Notes:

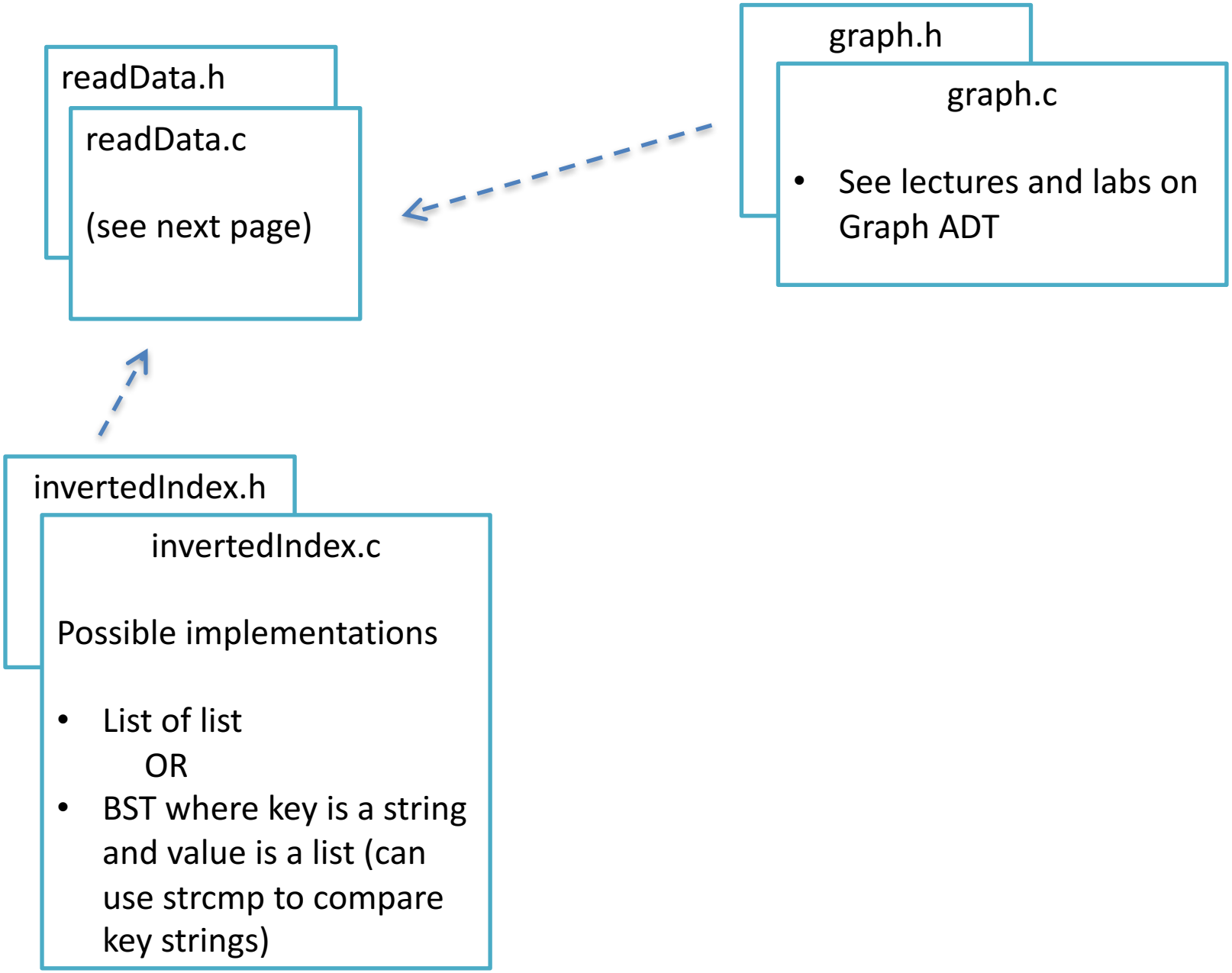
- All three parts are **independent**.
- **Suggested sequence:**
  - Part-C (doesn't need Graph or BST ADTs)
  - Part-A (Graph ADT)
  - Part-B (List ADT and/or BST ADT for indexing)

COMP9024 (20T0)

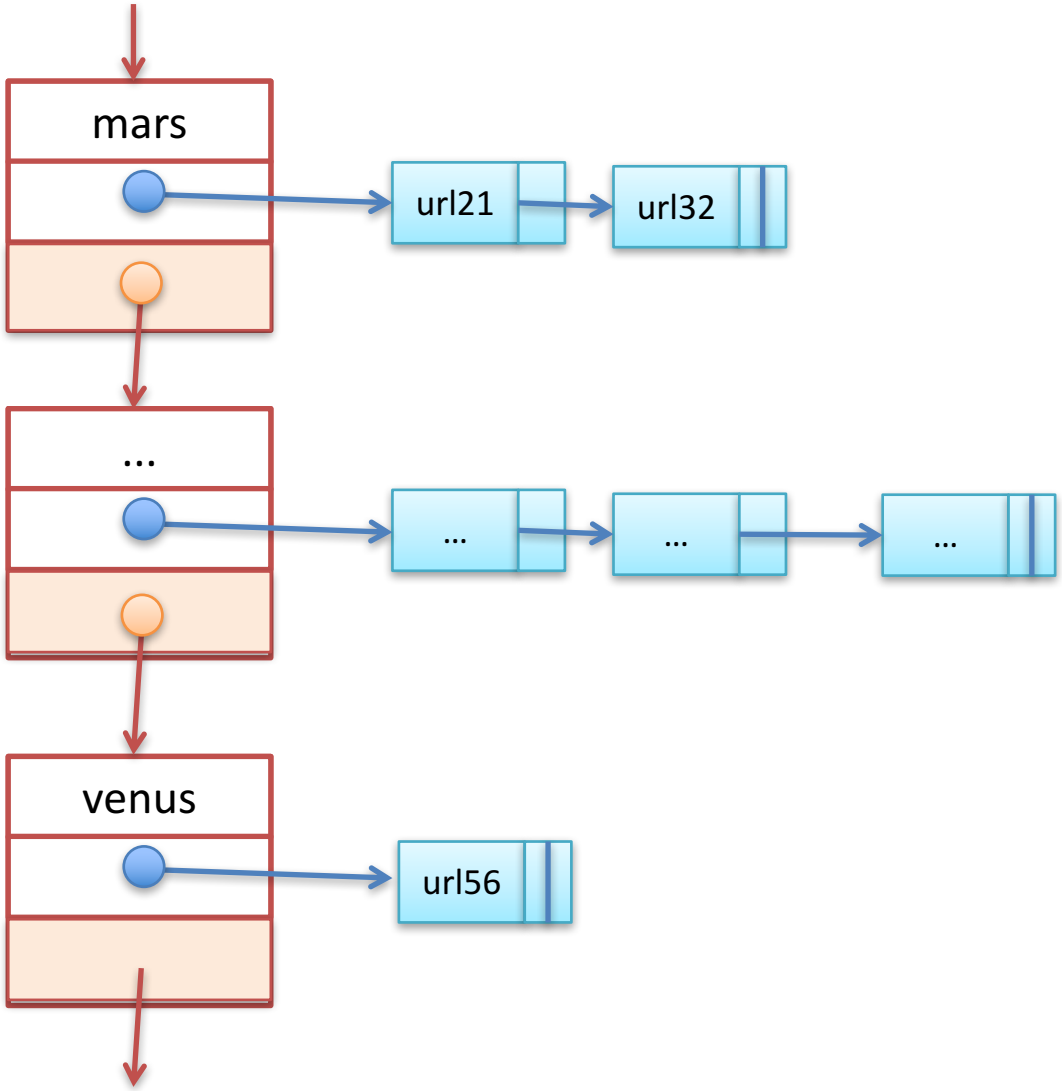
- **Assignment : How to Implement?**

Notes:

- The document offers some **suggestions only**, with incomplete pseudo code
- The pseudo code is easy to read, but may not be efficient. You need to improve it!
- You **can use** code from lecture material, however, must acknowledge it and provide a reference. For example you can use List, Graph and BST ADT implementations from the lectures, and adapt them for this assignment.
- You can build a graph structure using Adjacency Matrix or List Representation.


readData.h
readData.c
(see next page)
graph.h
graph.c
• See lectures and labs on Graph ADT
invertedIndex.h
invertedIndex.c
Possible implementations
• List of list
OR
• BST where key is a string and value is a list (can use strcmp to compare key strings)

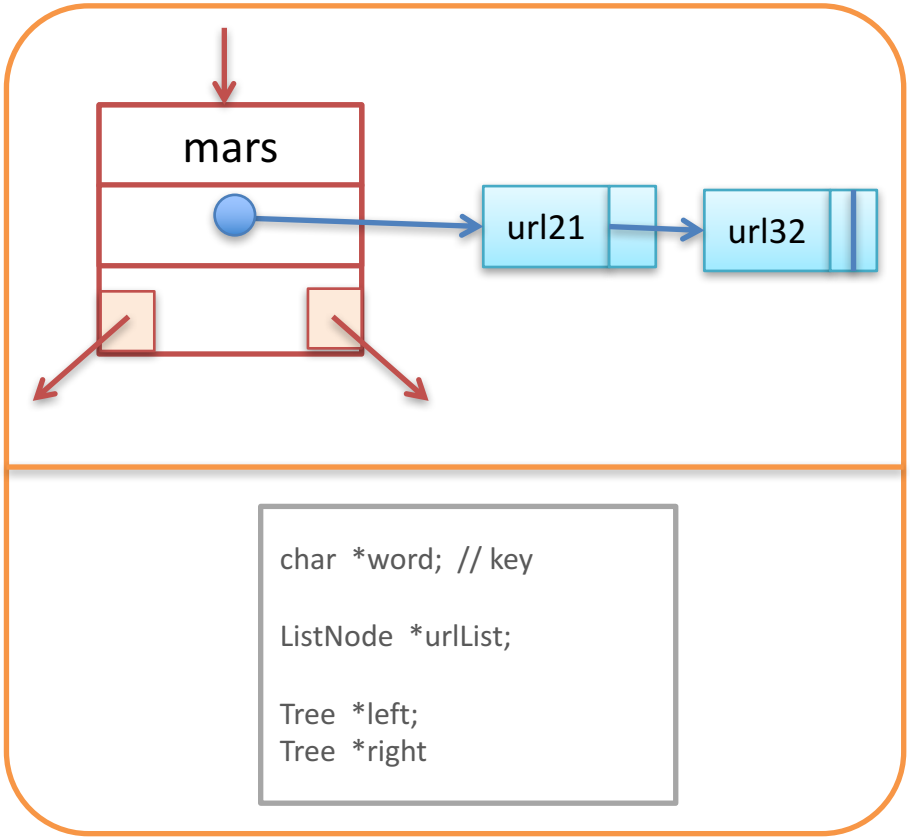
List of List



Binary Search Tree (BST)

Binary Search Tree where key is a string and value is a list
(can use `strcmp` to compare key strings).

We will discuss BST in Week 03/04. Please note that all three parts are independent, so you can work on Part-A and Part-C before Part-B (indexing).



readData.c

List_of_Urls ← GetCollection( )

Create a set (list) of urls to process by reading data from file "collection.txt"

Graph g ← GetGraph(List_of_Urls)

Create empty graph (use graph ADT in say graph.h and graph.c)

For each url in the above list

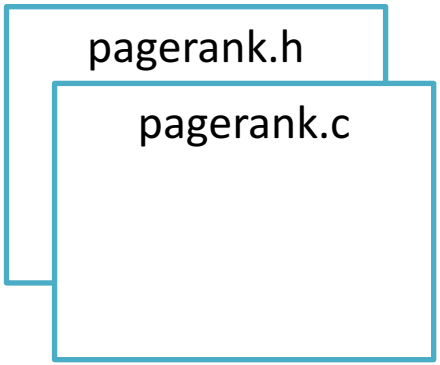
- read <url>.txt file, and update graph by adding a node and outgoing links

InvertedList ← GetInvertedList(List_of_Urls)

Create empty inverted list (use say List of lists, BST where values are lists, etc)

For each url in List_of_Urls

- read <url>.txt file, and update inverted index



pagerank.c

Get args : d, diffPR, maxIterations

List_of_Urls ← GetCollection( )
Graph g ← GetGraph(List_of_Urls)

List_Urls_PageRanks = calculatePageRank(g, d, diffPR, maxIterations);
Ordered_List_Urls_PageRanks = order (List_Urls_PageRanks)

Output Ordered_List_Urls_PageRanks to “pagerankList.txt”

invertedIndex.c

List_of_Urls ← GetCollection( )
InvertedIndex invertedIdx ← GetInvertedList (List_of_Urls)

Output invertedIdx to “invertedIndex.txt”

searchPagerank.h

searchPagerank.c

searchPagerank.c

Get query words from arguments

matched_Url_list ← findMatchedUrls(“invertedIndex.txt”, queryWords)
matched_Urls_with_PR ← findPagerank(“pagerankList.txt”, matched_Url_list)

Output ordered urls on stdout