

# COMP2521 (18s2)

## Ass2 (part-1) : 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 labs/lecture material, however, must acknowledge it and provide a reference. For example you can use graph ADT implementation from one of the labs, and adapt it 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

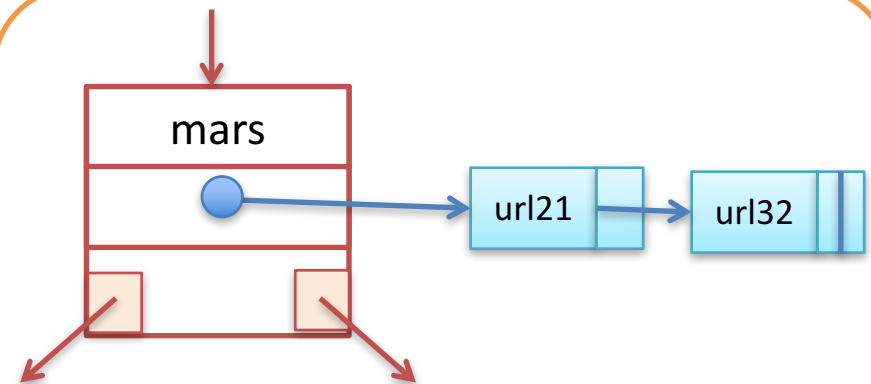
- See lectures and labs on Graph ADT

invertedIndex.h

invertedIndex.c

Implementation

- You **MUST** use BST where key is a string and value is a list as shown in the figure. You can use strcmp to compare key strings.



```
char *word; // key
```

```
ListNode *urlList;
```

```
Tree *left;
```

```
Tree *right
```

## **readData.c**

List\_of\_Urls  $\leftarrow$  GetCollection( )

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

Graph g  $\leftarrow$  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  $\leftarrow$  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.h

pagerank.c

### **pagerank.c**

Get args : d, diffPR, maxIterations

List\_of\_Urls  $\leftarrow$  GetCollection( )

Graph g  $\leftarrow$  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.h

invertedIndex.c

### **invertedIndex.c**

List\_of\_Urls  $\leftarrow$  GetCollection( )

InvertedIndex invertedIdx  $\leftarrow$  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