CENG 140

C Programming

Spring '2017-2018

Take Home Exam 2

Section: 02

Student Information

Full Name : Alper KOCAMAN
Id Number : 2169589

Answer of Part 2

Explanation of SortBook

Firstly, sortBook function takes 3 arguments which are

BookPtr bookPtr :
"BookPtr" data type is defined in Utilies.h header file and bookPtr is a variable of type BookPtr."bookPtr" variable is a global variable in CengBook.When main funciton of CengBook.c is called , it is the first thing configured."bookPtr" is said to keep everything about CengBook roughly.

ComparatorFunctionPointer compareFunctionPtr :
"compareFunctionPtr" is a pointer variable to a function.By setting up this variable properly, "compareByName" or "compareByPopularity" functions are called.

boolean ascending :
This variable is setted 1 if the input is "asc" or 0 if not.If "ascend" has 0 value ,then the "comparisonResult" multiplied with 1.

This function also uses 3 integers and 2 UserPtr variables.

"index1" is used to point the index of last user of bookPtr.

"index2" is a loop variable.

"comparisonResult" keeps the returned value from "compareByName" or "compareByPopularity" function which is called by "compareFunctionPtr".

Working Mechanism of SortBook Function :

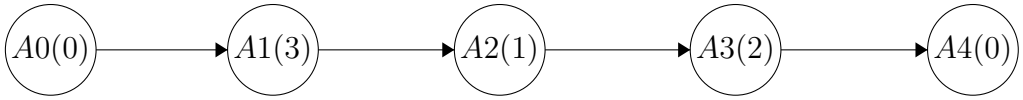
Firstly , by using outer for loop,"index1" is equal to "bookPtr->numberOfUsers - 1"."index1" is pointing the end of "userPointers".Inside loop of that loop equalize "index2" as 0 before starting.
Till the two indexes are same,every 2 users that have consecutive index number is investigated.In each iteration, "userPtr1" is assigned to userPointers[index2] and userPtr2 is assigned to userPointers[index2 +1].By sending two users "compareByName" or "compareByPopuBFS is a traversing algorithm where you should start traversing from a selected node (source or starting node) and traverse the graph layerwise thus exploring the neighbour nodes (nodes which are directly connected to source node). You must then move towards the next-level neighbour nodes.larity" function ,they are compared and returned value is hold in "comparisonResult".If "ascending" is false,then "comparisonResult" is multiplied by -1 to get the correct result.
If comparison is greater than 0 ,two users should be swapped.In the "if" block , the users are swapped according the their sorting criterias.
When two indexes are same,the inside for loop terminates.At that position ,the last element in the "bookPtr->userPointers" has the highest/lowest friend/alphabetical order.More clearly,the last element is sorted at the end of first iteration.
Before starting second iteration,"index1" is decremented 1 because the last one is ordered already.(This is an optimization and if this wasn't be done ,program still works fine but it takes longer time to give output.)
At the second iteration , the same things applied as in the first iteration.The only difference is this time inside for loop terminated with n-1 iterations rather than n iterations as in 1^{st} iteration.At the end of 2^{nd} iteration,the new last(i.e. the element previous of real last :)) element is ordered. At each iteration,the same things occurred.

This is the working mechanism of "bubble sort".If "bubble sort" is not optimized ,the time complexity is Θn^2.(since ,if there exist n elements and whole elements compared to other (n-1) elements,the total number of comparison is $n \times (n-1) = (n^2 - n)$).

An Example Figure for SortBook

Let A0,A1,A2,A3,A4 are 5 users and their friend numbers are given in the paranthesis.Assume that this users should be sorted by popularity in the ascending order.

First Positions:



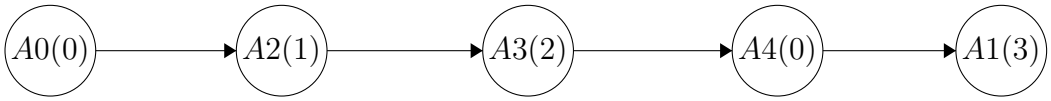
After First :

A1 has more friends than A0 so no swap between them.

Then compare A1 with A2.A1 has more friends so their place should be changed.

Then compare A1 with A3.A1 has more friends so their place should be changed.

Then compare A1 with A4.A1 has more friends so their place should be changed.



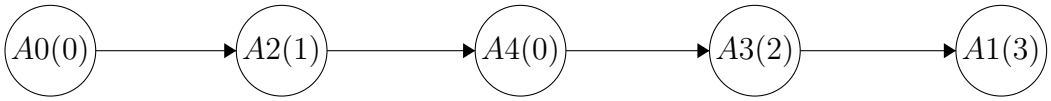
Then:

A0-A2(no swap)

A2-A3(no swap)

A3-A4(swap)

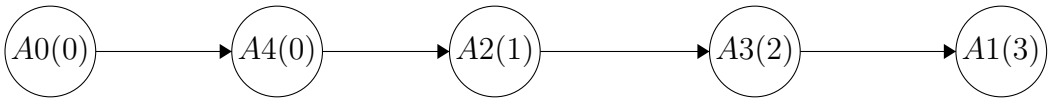
Stop because the last element is already ordered and no need to control.



Then:

A0-A2(no swap)

A2-A4(swap)



This is the explanation of how Bubble Sort Algorithm works.

Explanation of ShortestPath

This shortest path algorithm uses Breadth First Search since the weights of edges are all same which is 1.If this weights are different,shortest path algorithm cannot use BFS.There exist some other algorithms for that case(for example Dijktra's Algorithm).

Firstly, sortBook function takes 3 arguments which are

BookPtr bookPtr :(explained above)

UserPtr startUserPtr :
In the BFS algorithm , traversing should start from a selected node.This node is called as source or starting node."startUserPtr" is source node in the shortestPath function.

UserPtr targetUserPtr :
This is the node which we are looking for.The searching should stop if it is found.

In that function,it is required to use a queue.This need is satisfied with "static QueuePtr queuePtr". "queuePtr" variable is defined as static because it must store it's values even we call another functions.(If it wasn't like this,"queuePtr" lost it's values since it was in the stack segment.By defining it as static,this variable is in data segment now.)
"static int * followerIndicesArray" is used to keep data about if the element is investigated before or not.In that function , -1 represents start node,-2 represents element is not investigated before and other values represent the index of the follower user.
"static UserPtrArray friendPtrChainArray" is used to store pointers in the path.This array has 1 more (sizeof(UserPtr)) byte than maxNumberOfUsers$\times$(sizeof(UserPtr)) since it's last element should be "NULL".This value is returned at the end of the function.
Above data structures are search tools shortestPath function utilizes."static boolean isSearchToolsInitialized" is used to avoid creating these search tools more than one.It is initialized as False, then function takes memory for these tools.After than it is made True to prohibit function creating these tools again.

Working Meschanism Of ShortestPath Function:

"startUserIndex" and "targetUserIndex" are found with the help of "getUserIndexByPtr" function."followerIndicesArray" is initialized by all -2's except one location,index of start node namely "startUserIndex".This location holds -1 to indicate start node.Queue is resetted(head and tail equalize to 0.Head indicates the front of the queue where the element read and tail represents the first empty place to put new element in queue.) and the pointer of start user namely "startUserPtr" is put in the Queue.

Till the queue is empty, the first element of queue (first in first out so taken element is putted queue before all other elements)is taken.After than,all friends of that user(this user represented with "userPtr" and all friends of that user represented with "friendPtr" iteratively) is investigated in the inside for loop.This is done in that way:

1-Take the first element in the queue.(If this code block is starting for the first time,it is the source node.)Put it in the "userPtr" variable.

2-Find it's index and put it into "userIndex".

3-Take the friends of "userPtr".(respectively,in each turn take the next friend.)

4-Find this friend's pointer variable and put it into "friendPtr".Also, find it's index and put it into "friendIndex".

5-If that index numbered value in the "followerIndicesArray" is -2(which means not investigated yet) , change that -2 with "userIndex".

Fifth step means that there exist a path from "userPtr" to "friendPtr".

6-If this "friendPtr" is the element that we search for,than go to step 7,if not go to step 8.

6-Function calls "qReset" function.This function makes the tail and head of queue is 0(equal).

7-Afterwards,break terminates the inside for loop and the continuation condition of while is not correct("qIsEmpty" function returns true if head and tail is equal),so while loop terminates as well.

8-If for loop didn't terminated , continue to investigate friends of "userPtr"(go to step 3).

If for loop terminated but queue has elements , than repeat these steps for the first element of queue,head points the first element of queue.(go to step 1).

If there is no element left in the queue, this means that there doesn't exist a path from "startUserPtr" to "targetUserPtr"."targetUserIndex " in the "followerIndicesArray" is still left -2,that means this function returns "NULL".

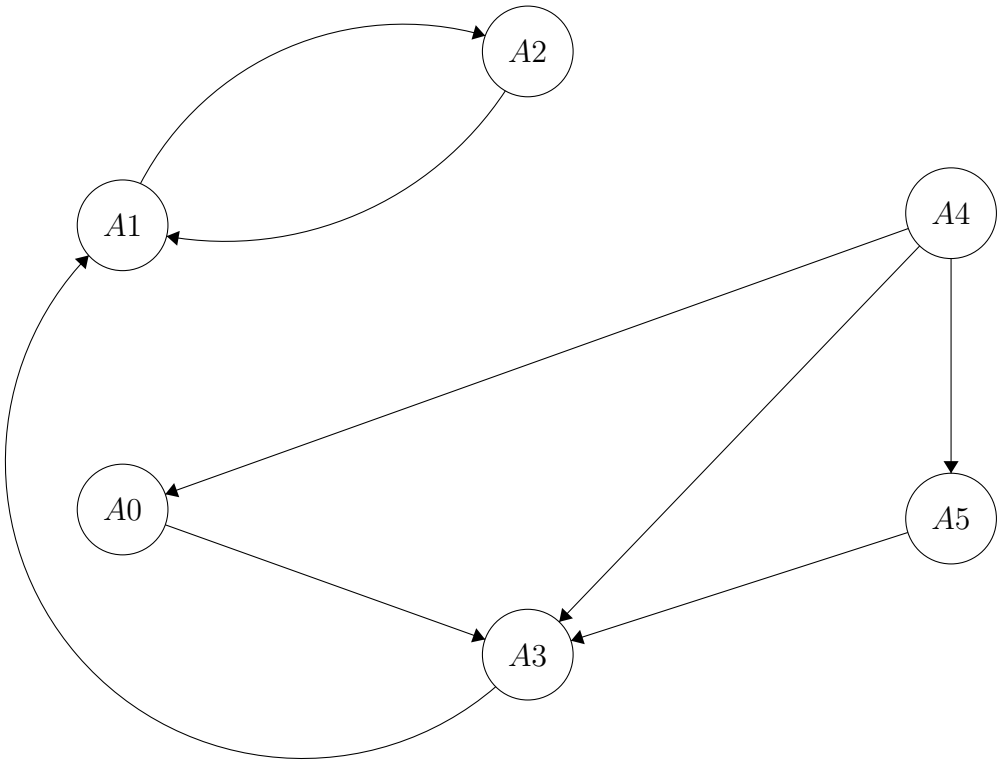
Let's say this target element is found,then new "userIndex" is "targetUserIndex".
"maxNumberOfUsers" gives it's value to "i" and "friendPtrChainArray" last element is made "NULL".Then in the while loop,the last empty place (which is before the "NULL" character for the first time) is filled with pointer which is userIndex'th place in the "userPointers"(for the first time this is the "targetUserPtr")."i" is decremented in every turn so "friendPtrChainArray" is filled from back to front."userIndex" is changed at the end of every iteration as the variable that located at the userIndex'th position at "followerIndicesArray".After than "(friendPtrChainArray + i)" is returned since "i" represents the first empty place from back,so if the path is not as long as

"friendPtrChainArray"(this is the situation only all elements are used for path),it is meaningless to print empty places may occur at the beginning of "friendPtrChainArray".

Roughly , this function takes the "startUserPtr",looks it's friends.If target is found,then return.If not,it begins to look friends' of it's friends.Then ,it spreads step by step in that way.

An Example Figure for ShortestPath

Let A0,A1,A2,A3,A4,A5 are 6 users and "maxNumberOfUsers" be 7.Also, let Pointers for corresponding users are PtrA0,PtrA1,PtrA2,PtrA3,PtrA4,PtrA5.Let the user wants the shortest path between A4 and A2.
"bookPtr = [PtrA0,PtrA1,PtrA2,PtrA3,PtrA4,PtrA5,](Assume they are registered in the order.)



Beginning:

"queuePtr","followerIndicesArray","friendPtrChainArray" are "NULL","isSearchToolsInitialized" is False.
"startUserPtr" = PtrA4 , "targetUserPtr" = PtrA2.
Then after the if block which initializes search tools,

"queuePtr","followerIndicesArray","friendPtrChainArray" have allocated memory for them,isSearchToolsI
is True.

After than,
"startUserIndex" = 4
"targetUserIndex" =2
followerIndicesArray = [-2,-2,-2,-2,-1,-2,-2].(it's length should be "maxNumberOfUsers").
After qReset,both head and tail are 0.
After qPut , queuePtr = [PtrA4],tail is 1.

Inside While Loop:

Since $head \neq tail$ while loop is executed.

qGet returns PtrA4 and this value is putted in the userPtr.

After qGet ,head is 1.

UserIndex = 4.

Inside For Loop:

friendPtr = PtrA0 and friendIndex=0.

Since the 0 indexed element of followerIndicesArray is -2,if block executes.

Now, followerIndicesArray = [4,-2,-2,-2,-1,-2,-2]

Since this is not target,we put PtrA0 to queue.(Tail = 2)

Same applies for A3 and A5 and after than

followerIndicesArray = [4,-2,-2,4,-1,4,-2]

queuePtr = [PtrA4,PtrA0,PtrA3,PtrA5].(Tail=4,Head=2)

Then for loop terminates,again in while loop($head \neq tail$)

qGet returns PtrA0 and this value is putted in the userPtr.

After qGet ,head is 2.

UserIndex = 0.

Inside For Loop:

friendPtr = PtrA3 and friendIndex=3.

Since followerIndicesArray 3^rd indexed element is not -2,there is nothing to do.

Then for loop terminates,again in while loop($head \neq tail$)

qGet returns PtrA3 and this value is putted in the userPtr.

After qGet ,head is 3.

UserIndex = 3.

Inside For Loop:

friendPtr = PtrA1 and friendIndex=1.

Since the 1 indexed element of followerIndicesArray is -2,if block executes.

Now, followerIndicesArray = [4,3,-2,4,-1,4,-2]

Since this is not target,we put PtrA3 to queue.(Tail = 5)

Then for loop terminates,again in while loop($head \neq tail$)

qGet returns PtrA5 and this value is putted in the userPtr.

After qGet ,head is 4.

UserIndex = 5.

Inside For Loop:

friendPtr = PtrA3 and friendIndex=3.

Since followerIndicesArray 3^rd indexed element is not -2,there is nothing to do.

Then for loop terminates,again in while loop($head \neq tail$)

qGet returns PtrA1 and this value is putted in the userPtr.

After qGet ,head is 5.

UserIndex = 1.

Inside For Loop:

friendPtr = PtrA2 and friendIndex=2.

Since the 2 indexed element of followerIndicesArray is -2,if block executes.

This is our target!!!

We found it!!!

Now, followerIndicesArray = [4,3,1,4,-1,4,-2]

qReset equalizes head and tail(end of while loop) and there is a "break" for the inner loop.

queuePtr = [PtrA4,PtrA0,PtrA3,PtrA4,PtrA1];

followerIndicesArray = [4,3,1,4,-1,4,-2];

isSearchToolsInitialized = True;

Generating "friendPtrChainArray":

userIndex=2(target);

i=7;

Last element is "NULL";

friendPtrChainArray = [, , , , , , , NULL] (size should be 1 more than max number)

In the While Loop

1.st iteration = friendPtrChainArray = [, , , , , , PtrA2 , NULL] (i = 6)

2.nd iteration = friendPtrChainArray = [, , , , , PtrA1 , PtrA2 , NULL] (i = 5)

3.rd iteration = friendPtrChainArray = [, , , , PtrA3 , PtrA1 , PtrA2 , NULL] (i = 4)

4.th iteration = friendPtrChainArray = [, , , PtrA4 , PtrA3 , PtrA1 , PtrA2 , NULL] (i = 3)

Return value should (friendPtrChainArray + i) = (friendPtrChainArray + 3) because first 3 element(i.e. up to and including index 2).

Answer of Part 3

1-By using "getUserPtrByName" function , corresponding pointer is obtained.(This function iterativelt investigate every element of bookPtr->userPointers to find the pointer which has name target.)

2-If this is "NULL" then an error message is printed and function returns but this is not the worst case.To find complexity,worst case should be investigated.

3-Search every element in the CengBook to find followers of this user.(Iteratively send an user with this will delete user to remove conection to look they are connected.)If a match found, then friend index of delete user should be found(a for needed again).Shift all friend indexes right to that index iteratively as well.At the end ,friend number of that user is decremented.

4-Step 3 is repeated till whole followers of that delete user are found and their connection is removed.

5-Remove the connection between this user and it's friends.In a for loop,follower number of those friends should be decremented.

6-The index of that user should be found."getUserIndexByPtr" function does this for us iteratively.

7-This delete user should be deallocated.

8-Iteratively shift bigger indexes to 1 place left.

9-Number of users should be decremented 1.

10-The last element of "userPointers" should be made "NULL" 11-For each removed connection,there is a message printed and at the end,function should prints User ('userName') is deleted from CengBook to inform user that these steps are done successfully.