Name: Omar Yossuf
SID: 900212166

Report

Outline

4.1.0 getTillChar

4.2.0 Webpage

4.3.0 Case sensitivity

4.4.0 “” “”

Side Notes

- I created a webpage class that contains all the needed information to each website.
- Created a vector of pointers to the webpage class which I use to access any data wanted.
- Objects in class webgraph:
    - string URL;
    - Vector of string keywords;
    - Vector of pointers to webpage hyperlinks;
    - Vector of pointers to inbound;
    - int impressions;
    - int clicks;
    - double score;
    - double PR;
    - double ctr;
- **gitTillChar** is a function that I have created that takes a string and a char, and returns the substring from the beginning till the char, then deletes this substring from the original one

Pseudocode

Search

ANDSearch:

Function purpose:

Takes a string and returns a vector of pointers to webpage as a result.

```
Vector of pointers to webpage ANDSearch(string input) // n (k + m) :. O(nm)
{
    Declare res as a vector of pointers to webpage;

    Declare results and urls as strings and initialize results to “”;

    Declare URLs and KWs as vectors of types strings;

    While (input is not empty) // loop to extract the user search without AND
    {
        Declare temp as string
        Do
        {
```

```
        Temp = getTillChar (input, ‘ ‘ );
    }while (temp == “AND”)

    Erase quotation marks from string temp;

    If ( temp[0] equals to space character)
        Delete the first character (i.e. space)

    Capitalize temp to eliminate case sensitivity;

    Pushback temp into KWs;
}

For i <- 0 to number of webpages // n times
{
    Declare Flag as Boolean and initialize to true;

    Declare URL as string and set to   getURL of webpages[i];

    Declare keys as string ;

    For j <- 0 to number  of keywords in webpages[i] // k times
        keys = keys +  keywords[j] of webpages[i] + “ “ ;
        Capitalize keys

        For i <-0 to size of vector KWs // m times
        {
            If (!Flag)
                Break
            Else
                If (KWs[i] does not exists in keys )
                    Flag = false
        }
        If (Flag)
            If ( URL is not found in results)
            {
                results = results + URL + " "
                Pushback webpage[i] in res
            }

    }
    Return res
}
```

ORSearch:

Function purpose:

Takes a string and returns a vector of pointers to webpage as a result.

```
Vector of pointers to webpage ORSearch(string input) // O(nm) same as ANDSearch
{
    Declare res as a vector of pointers to webpage;

    Declare results and urls as strings and initialize results to “”;

    Declare URLs and KWs as vectors of types strings;

    While (input is not empty) // loop to extract the user search without OR
    {
        Declare temp as string
        Do
        {
            Temp = getTillChar (input, ‘ ‘ );
        }while (temp == “OR”)

        Erase quotation marks from string temp;

        If ( temp[0] equals to space character)
            Delete the first character (i.e. space)
        Capitalize temp to eliminate case sensitivity;

        Pushback temp into KWs;
}

For i <- 0 to number of webpages
{
        Declare Flag as Boolean and initialize to false;

        Declare URL as string and set to   getURL of webpages[i];

        Declare keys as string ;

        For j <- 0 to number  of keywords in webpages[i]
                keys = keys +  keywords[j] of webpages[i] + " " ;
        Capitalize keys

        For i <-0 to size of vector KWs
        {
                If (!Flag)
                        Break
                Else
                If (KWs[i] does not exists in keys )
                    Flag = true
        }
        If (Flag)
            If ( URL is not found in results)
            {
                results = results + URL + " "
                Pushback webpage[i] in res
            }

    }
    Return res
}
```

QuoteSearch:

Function purpose:

Takes a string and returns a vector of pointers to webpage as a result.

```
Vector of pointers to webpage QuoteSearch(string input) // O(nm)
{
    Declare res as a vector of pointers to webpage;
    Erase quotation marks from string temp;

    For i <- 0 to number of webpages // n times
    {
        Declare Flag as Boolean and initialize to false;

        For j <- 0 to number  of keywords in webpages[i] // k times
        {
            Declare key as a string and initialize it to getKeywords[j] of webpages[i];

            If (key == input)
                Flag = true;
        }

        If (Flag)
            Pushback webpage[i] in res
    }
    Return res
}
```

Ranking

Page Rank

```
Void CalculatePR() // O(nm)
{
    Declare numOfSites as a const double and initialize it to the size of the vector of webpage

    Declare PRVec as a vector of double

    Declare MOE as a double and initialize it to 0.000001

    Declare j as an int and initialize it to 0

    Declare moe as a boolean vector with size numOfSites and initialized to false

    Declare allmoe as a boolean and initialized to false

    For i <- 0 to numOfsites // n times
    {
        Set the pagerank of webpage[i] to 1 / numOfSites
        PRVec[i] <- pagerank of webpage[i]
    }
```

```
While (!allmoe && j < 100) // 100 times
{
    For i <- 0 to numOfsites // n times
    {
        Declare tempPR as a double and initialize it to 0.0

        For k <- 0 to size of the inbound vector of webpage[i] // m times
            tempPR = tempPR + inbound[k] of webpage[i] / the size of the
hyperlinks vector of inbound[k] of webpage[i]

        tempPR = (0.15 / numOfSites) + 0.85 * tempPR

        PRVec[i] <- tempPR
    }

    allmoe <- true

    For i <- 0 to numOfSites // n times
    {
        If (absolute(pagerank of webpage[i] - PRVec[i] > MOE))
            Pagerank of webpage[i] <- PRVec[i]
        Else
```

```
                    Moe[i] = true

                if(!moe[i])
                    allmoe <- false
            }
            j <- j + 1;
        }
}
```

CTR

Note: CTR is a function inside the webpage class

```
Void calculateCTR() // O(1)
{
    CTR <- double (clicks) / double (impressions)
}
```

Score

```
Void calculateScore() // O(1)
{
    Score <- (0.4 * PR) +
    ((1 - (0.1 * impressions / (1 + (0.1 * impressions)))) * PR + ((0.1 * impressions / (1 + (0.1 *
impressions)))) * ctr) * 0.6;
}
```

Time and Space Complexity

Space complexity

Search Space Complexity

Since only vectors are used and it's a vector of pointers, we know that a space complexity of a pointer is the same as that of any other primitive - i.e. O(1), and the space complexity of a vector of objects is O(n*space complexity of each object) s.t. n is the size of that vector. Thus, the space complexity of the vector used in the program is O(n). Also, we know that the space complexity of a string is O(m) s.t. m is the size of the string. Additionally, we have a vector of strings used. Thus, its space complexity is O(nm) s.t. n is the size of the vector and m is the size of the string. Finally, the dominant one of these complexities is O(nm). Thus, the space complexity of the search functions is O(nm).

Ranking space complexity

Pagerank

Pagerank only uses:

- Vector of double O(n)

- Vector of boolean O(n)
- Vector of pointers O(n)

Thus, the space complexity of the pagerank algorithm is O(n).

CTR

CTR only uses double so it is O(1)

Score

Score only uses double so it is O(1)

Time complexity

Details are in the pseudocode

Search

ANDSearch

O(nm)

ORSearch

O(nm)

QuoteSearch

O(nm)

Ranking

PageRank

O(nm)

CTR

O(1)

Score

O(1)

Data structures used

Only vectors were used.

Design Tradeoffs

getTillchar

I created a new function called getTillChar that takes an address to a string and a char. This function then separates the string into two strings. The first string represents the substring from the beginning of the original string till the char given. And the second string is the original string without the previous substring. Finally, the function returns the substring and overwrites the original string to the second one. This function helps with reading any file with a different stopping conditions like the csv (comma separated value) file.

Webpage

I created a class named webpage that contains all the needed information for the web page. Two of these objects are vectors of pointers to the webpage class (the same class) representing both the hyperlinks and the inbounds. This helped me integrate between everything as the vector is of pointers so it's the same location in memory. Hence, everything is always accessible.

Case Sensitivity

In the ANDSearch and ORSearch functions, I used std::transform to transform the input's character to be all caps and transformed the keywords to be all caps, this way both search types are not case sensitive. However, in the quotation search I didn't use it as quotation should be the exact thing.

“” “”

I made my code detect if the user added two words within quotations, but not together to read it as and i.e. “data” “structures” = data AND structures