Building a search engine

CSCE 2203

Analysis and Design of Algorithms Lab

Dr. Mohamed Elhalaby

American University in Cairo

Name: Mariam Fawzy

ID: 900192318

1) Websites indexing:

I used the URL of each website as its index. I created it while reading the “keyword” file. I chose this file specifically because all significant websites are included in it. I used a vector of strings to carry all the websites and iterate over it when needed. Consequently, most of the data structures used in the project are maps which their keys are strings.

pseudo-code:

for(i 0: n) //where n is the number of websites in the keyword file

{

Add the URL of the website to a vector of strings

}

Complexity of the algorithm:

The algorithm loops over the websites and insert them in a vector, so it has complexity of O(n), where n is the number of websites. To optimize, I inserted them while reading the file and not in a separate function.

2) Ranking algorithms:

a) Page ranking algorithm(page_rank_algo):

I determined the rank of each website depending on some iterations. The first iteration gives each website a value equals $\frac{1}{total\ number\ of\ vertices}$. Then, in the following iterations I give each website a value according to the following equation: pr(website) = $\sum_{0}^{n} \frac{pr(p)}{L(p)}$, where pr is the page rank, p is the parent of the website in the web graph and L is the number of out vertices of the parent of the website.

There are several ways to determine the number of iterations, but I found that the most convenient one is to do iterations till the difference between the value of the current page rank and previous page rank is less than 0.001. At this point the iterations does not make huge difference and it breaks.

Next, I give the websites ranks (1, 2, …, n) to use rank in the CTR function.

I used the URL of each website as its index. I created it while reading the "keyword" file. I chose this file specifically because all significant websites are included in it. I used a vector of strings to carry all the websites and iterate over it when needed. Consequently, most of the data structures used in the project are maps which their keys are strings.

pseudo-code:

```
page_rank_algo{
    for (i 0->n) // looping over the websites
        Pr(website) = 1/n;
    while(the error does not exceed 0.001){
        for(i 0->n)
            for(j 0 -> m) //looping over the children of each website
                pr[website] += pr[parent] + number of the children of this website;
    }
    //Giving the websites ranks between 1, 2, 3, …., n
    Great_prev = max double;
    for( i 0->n)// i is 1, 2, 3…
        for(website 0->n)
            if(pr(i) > pr(j) and page[j] < great_prev and pr(j) is not visited){
```

pr(website) = i;

mark this website as visited;

}

Thus after this code I have all websites ranked in the map called page_rank.

Complexity of the algorithm:

The algorithm loops over the websites once to calculate the page rank value in the first iteration, so it has O(n) complexity

Then, it loops over the web graph once in every iteration to calculate the page rank of each website in this iteration, so this has complexity of O(c(n + e)) where c is the number of iterations(constant), n is the number of websites(vertices) and e is the number of edges in the graph.

Then, I looped over the page_rank map twice to give each website an integer rank(1, 2, 3, 4, …). This has complexty of $O(n^2)$

Consequently, the total complexity of the algorithm = $O(n^2)$ where n is the number of websites.

b) Ranking the websites(CTR):

In this step, I only worked on the target websites in order to optimize the program. The target websites are the websites that include that include the keyword that the user is looking for. I calculated the page score of each website according to the equation provided in the documentation of the project, where CTR = $\frac{number\ of\ clicks}{number\ of\ impressions}$

pseudo-code:

```
CTR{
    for(i : target_sites){
        find number of impressions of i;
        find number of clicks of i;
        calculate the score using the mentioned equation;
    }
    for(i : target_sites)
        push back the score to every site and they get arranged descending.
    }
```

Complexity of the algorithm:

The two loops are of complexity O(m) where m is the number of websites that include the target keyword, so O(2m) = O(m).

Main data structures used by the algorithm:

1. **unordered_map:**
   Since I used the URL(string) as index for the websites, I frequently used unordered_maps of (string, double) in my calculations, for example:
   - unordered_map<string, vector<string>> web_graph:
     This is used as an adjacency list to express the graph connecting between all websites.
   - unordered_map<string, vector<string>> keyword;
     This is used to save the keywords that distinguish every website.
   - unordered_map<string, vector<int>> no_of_impressions;
     This is used to save and update number of impressions and number of clicks of each single website.
   - unordered_map<string, double> page_rank;
     This is used to save the rank of each website using integers(1, 2, 3, …., n)
2. **Vector:**
   - vector<string> websites:
     This is used as the index of each website, I am looping over I to get the key of any of the above maps
   - vector<string> key:
     This is used to receive the input of the user and determine the type of the search.

3. **Map:**
   - map<double, string, greater <double> > target_sites:
     This is an urdered map in order to save the results in an arranged manner in order to display them to the user.

Tradeoffs and justifications:

- I used the URL as the index of the website as this saved time and facilitate looping over the websites in any stage of the project.
- I used an unordered map instead of a map in the website indexing. This was mainly because maps are balanced binary search trees with insertion in O(log(n)) and retrieval O(log(n)). Unlike the unordered map, which is basically a hash table that has insertion and retrieval in O(1) according to the cpp reference.

How to operate the program:

Note that, always write the number of the option except if you would like to view a website, you have to write the URL as indicated in the following screenshoot: