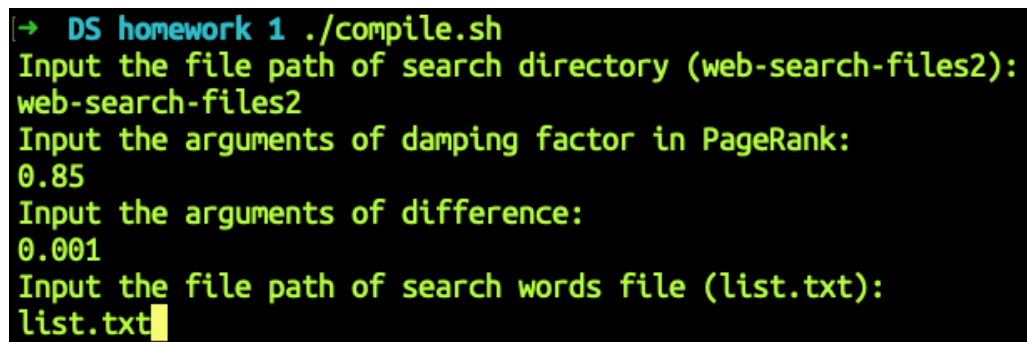


The program for the assignment is written by C++.

The compiled program, PageRank\_SearchEngine, was compiled and executed by compile.sh. The C++ source code files are main.cpp, pagerank.cpp and hash\_table.cpp with a header file, functions.h.

## Usage

After executing the program, the user can input the path of directory with pages, the damping factor and difference parameters, the path of searching words list (list.txt). After finishing the inputs, the program stores the results in multiple files named as the assignment requiring.



```

DS homework 1 ./compile.sh
Input the file path of search directory (web-search-files2):
web-search-files2
Input the arguments of damping factor in PageRank:
0.85
Input the arguments of difference:
0.001
Input the file path of search words file (list.txt):
list.txt

```

Figure 1. The execution interface of the program.

## Brief explanation of implementation

There are some comments explaining the purpose of respective functions in the header file, functions.h.

The pages linking information is stored in a matrix implemented by two dimensional vector storing float. While the program is reading the pages linking information, it also takes the searching words in each page and builds the hash table for searching.

### a. Page rank

To calculating the converged page rank, the program iterates the formula below.

$$PR(P)_{new} = PR_{matrix} \times PR(P) + \frac{(1 - d)}{N}$$

The values stored in the two dimensional vector are shown below.

$$PR_{matrix} = d \left[ \frac{1}{C(T_1)} \quad \frac{1}{C(T_2)} \quad \cdots \quad \frac{1}{C(T_n)} \right]$$

The final page rank is calculated while the difference with previous page rank is below the setting difference. The results with different combinations of arguments are output as Page Rank files.

### b. Hash table

The class Hash declared in the header file is the data structure of hash table and its codes are in the hash\_table.cpp.

The hash table is an array stored the address of items (self-defined structs) which can link to the next item, using the concept of separate chaining. If two strings has the same hash

value, the item of one will be added another's by link address. The hash function is shown below.

$$\text{hash}(\text{string}) = (\sum \text{int}(\text{char of string}) \times 17) \bmod (\text{hash table size})$$

## Performance

The time complexity of creating  $PR_{matrix}$  is  $O(nm)$ , where  $n$  is the number of total pages and  $m$  is the average number of links in each page. The time complexity of calculating the final page rank is  $O(\beta n^3)$ , where  $n$  is the number of total pages and  $\beta$  is the number of iterations. There is a  $n^3$  in the time complexity because of multiplication of the matrix and the previous page rank. At last, the time complexity for outputting the sorted page rank list is  $O(\beta n^3 + n \log n)$ .

The time complexity of building hash table is  $O(n(1 + \alpha))$ , where  $n$  is the size of input data size, and the time complexity of searching in hash table  $O(k \log k + (1 + \alpha))$ , where  $k$  is the number of pages searched by one string, and  $\alpha$  is the load factor of the hash table. The searching results have to be output by page rank, so there is an additional time,  $O(k \log k)$ , for sorting a vector with page structs (self-defined structs with page number and rank value). The time complexity of outputting the searching results file is  $O(s(k \log k + (1 + \alpha)))$ , where  $s$  is the number of all searching words and  $k$  is the number of pages searched by one string.

The method for building reverse index is searching in hash table for each word. Using the established hash table, the time complexity of outputting the reverse index file is  $O(s(k + (1 + \alpha)))$ , where  $s$  is the number of all searching words and  $k$  is the number of pages searched by one string.

The actual executing time of achieving all the requirements is shown below.

```
→ DS homework 1 time ./PageRank_SearchEngine
The number of processing pages: 501
Iteration counts: 9
Difference: 0.000865657
./PageRank_SearchEngine 0.11s user 0.02s system 91% cpu 0.148 total
```

Figure 2. The time performance of executing and output all the results with the arguments  $d=0.85$  and  $\text{DIFF}=0.001$ .