EE3980 Algorithms

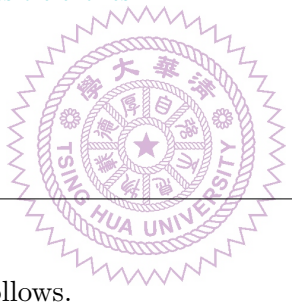
Homework 2. Random Data Searches

Due: Mar. 20, 2021

Given a set, search algorithms find the location of the needed data. This type of algorithms have many applications and have also been studied extensively. In this homework, we will study a set of primitive searching algorithms, in which, the data set been sought is assumed to be random. Four algorithms are given below.

The first one **linear search** is shown below.

```
// Find the location of word in array list.
// Input: word, array list, int n
// Output: int i such that list[i] = word.
1 Algorithm search(word, list, n)
2 {
3     for i := 1 to n do {  // compare all possible entries
4         if (list[i] = word) then return i;
5     }
6     return −1;  // unsuccessful search
7 }
```



The second search algorithm is as follows.

```
// Find the location of word in array list.
// Input: word, array list, int n
// Output: int i such that list[i] = word.
1 Algorithm search2(word, list, n)
2 {
3     for i := 1 to n step 2 do {  // compare all possible entries
4         if (list[i] = word) then return i;
5         if (list[i + 1] = word) then return i + 1;
6     }
7     return −1;  // unsuccessful search
8 }
```

The **odd-even search** is as follows.

```
// Find the location of word in array list.
// Input: word, array list, int n
// Output: int i such that list[i] = word.
Algorithm OEsearch(word, list, n)
{
    for i := 1 to n step 2 do {  // compare odd entries
        if (list[i] = word) then return i;
    }
    for i := 2 to n step 2 do {  // compare even entries
        if (list[i] = word) then return i;
    }
    return −1;  // unsuccessful search
}
```

And, the **randomized odd-even search** is as follows.

```
// Find the location of word in array list.
// Input: word, array list, int n
// Output: int i such that list[i] = word.
Algorithm ROEsearch(word, list, n)
{
    choose j randomly from the set {0, 1} ;
    if (j = 1) then {
        for i := 1 to n step 2 do  // compare odd entries
            if (list[i] = word) then return i;
        for i := 2 to n step 2 do  // compare even entries
            if (list[i] = word) then return i;
    }
    else {
        for i := 2 to n step 2 do  // compare even entries
            if (list[i] = word) then return i;
        for i := 1 to n step 2 do  // compare odd entries
            if (list[i] = word) then return i;
    }
    return −1;  // unsuccessful search
}
```

Your assignment is to write a C program that contains four functions:

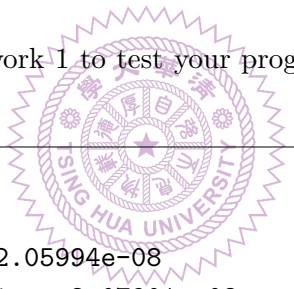
```
int Search(char *word, char **list, int n);      // Linear Search
int Search2(char *word, char **list, int n);     // Search 2 Algorithm
int OEsearch(char *word, char **list,int n);     // odd-even Search
int ROEsearch(char *word, char **list,int n);    // randomized Odd-Even Search
```

where `word` is the target `string` to be located; the `list` is an array of `string pointers`, and the size of the array is defined by the other parameter `n`. All functions return the index i such that $list[i]$ equals to *word*, if *word* cannot be found then -1 is returned.

To measure the performance of these functions, a `main` function should be implemented. It should

1. Read in a word list as homework 1.
2. Assuming successful searches, measure the average CPU time for each algorithm (Number of repetitions should be greater than or equal to 500).
3. Still assuming successful searches, measure the worst-case CPU time for each algorithm (number of repetition should be greater than or equal to 5000).

Use the nine wordlist files of homework 1 to test your program. Example of program execution is shown below.




```
$ a.out < s1.dat
  n: 10
  Linear search average CPU time: 2.05994e-08
  Search 2 algorithm average CPU time: 2.07901e-08
  Odd-even search average CPU time: 2.16007e-08
  Random odd-even search average CPU time: 3.76225e-08
  Linear search worse-case CPU time: 4.25816e-08
  Search 2 algorithm worse-case CPU time: 5.55992e-08
  Odd-even search worse-case CPU time: 4.94003e-08
  Random odd-even search worse-case CPU time: 5.24044e-08
```

The time complexities of these four algorithms should be analyzed and compared to those of the measured CPU times.

Notes.

1. One executable and error-free `C` source file should be turned in. This source file should be named as `hw02.c`.
2. A report file in `pdf` format is also needed. This file should be named as `hw02a.pdf`.
3. Submit your `hw02.c` and `hw02a.pdf` on EE workstations using the following command:

   `~ee3980/bin/submit hw02 hw02.c hw02a.pdf`

   where `hw02` indicates homework 2.
4. Your report should be clearly written such that I can understand it. The writing, including English grammar, is part of the grading criteria.
5. In comparing two strings, the following library function in the `<string.h>` package can be used.
   ```
   int strcmp(const char *s1, const char *s2);
   ```