EE3980 Algorithms

Homework 2. Random Data Searches

Due: Mar. 22, 2020

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. Three 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) return i;
5     }
6     return −1; // unsuccessful search
7 }
```

The **bidirection search** is as follows.

```
// Bidirectional search to find the location of word in array list.
// Input: word, array list, int n
// Output: int i such that list[i] = word.
1 Algorithm BDsearch(word, list, n)
2 {
3     for i := 1 to n/2 do { // compare all entries from both directions
4         if (list[i] = word) return i;
5         if (list[n − i − 1] = word) return n − i − 1;
6     }
7     return −1; // unsuccessful search
8 }
```

And, the **random-direction search** is as follows.

```
// Random-direction search to find the location of word in array list.
// Input: word, array list, int n
// Output: int i such that list[i] = word.
1 Algorithm RDsearch(word, list, n)
2 {
3     choose j randomly from the set {0, 1} ;
4     if (j = 1) then
5         for i := 1 to n do { // forward search
6             if (list[i] = word) return i;
7         }
8     else
9         for i := n to 1 step −1 do { // backward search
10            if (list[i] = word) return i;
11        }
12    return −1; // unsuccessful search
13 }
```

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

```
int Search(char *word, char **list, int n);      // Linear Search
int BDSearch(char *word, char **list,int n);     // Bidirection Search
int RDSearch(char *word, char **list,int n);     // Random-direction 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: 5.0211e-08
  Bidirection search average CPU time: 5.27859e-08
  Random-direction search average CPU time: 7.55787e-08
  Linear search worse-case CPU time: 6.7997e-08
  Bidirection search worse-case CPU time: 5.54085e-08
  Random-direction search worse-case CPU time: 5.25951e-08
```

The time complexities of these three 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 `pdf` file is also needed. This report 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 1.
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);`