Homework 2

Start of Report

```
hermod.ics.uci.edu - PuTTY
-bash-4.2$
-bash-4.2$
-bash-4.2$
 -bash-4.2$
 -bash-4.2$
-bash-4.2$
main main.cpp Makefile random.txt Timer.h UnorderedArrayList.cpp UnorderedLinkedList.cpp -bash-4.2$ make
echo -----
                       -compiling main.cpp to create executable program main--
cend construction main.cpp to create executable program main------
g++ -ggdb -std=c++11 -Wpedantic -Wall -Wextra -Werror -Wzero-as-null-pointer-constant -Weffc++ main.cpp Unordered
LinkedList.cpp UnorderedArrayList.cpp -o main -bash-4.2$ main
Testing UnorderedArrayList:
5e-05
Testing UnorderedLinkedList:
2e-05
 5e-05
0.000139
-bash-4.2$ valgrind main
 ==14640== Memcheck, a memory error detector
==14640== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==14640== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
 =14640== Command: main
 =14640==
Testing UnorderedArrayList:
0.015333
0.007627
0.006415
Testing UnorderedLinkedList:
  =14640== HEAP SUMMARY:
                in use at exit: 72,704 bytes in 1 blocks total heap usage: 114 allocs, 113 frees, 1,585,272 bytes allocated
 =14640==
  =14640==
  =14640== LEAK SUMMARY:
  =14640==
=14640==
                definitely lost: 0 bytes in 0 blocks indirectly lost: 0 bytes in 0 blocks possibly lost: 0 bytes in 0 blocks still reachable: 72,704 bytes in 1 blocks suppressed: 0 bytes in 0 blocks
 =14640== Rerun with --leak-check=full to see details of leaked memory
 =14640== For counts of detected and suppressed errors, rerun with: -v
 =14640== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
 -bash-4.2$
```

Files in directory followed by a normal run of the program followed by a valgrind run of the program. Random.txt only has like 100 words.

```
A hermod.ics.uci.edu - PuTT
compiling main.cpp to create executable program main---
0.009003
19.7532
Testing UnorderedLinkedList: 0.014632
18.4807
-bash-4.2$ valgrind main
 ==1908== Memcheck, a memory error detector
==1908== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==1908== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
 =1908== Command: main
 =1908==
Testing UnorderedArrayList:
0.534179
498.609
Testing UnorderedLinkedList:
469.72
484.05
 ==1908==
 =1908== HEAP SUMMARY:
              in use at exit: 72,704 bytes in 1 blocks
total heap usage: 47,728 allocs, 47,727 frees, 3,440,884 bytes allocated
 =1908==
  =1908==
  =1908== LEAK SUMMARY:
               definitely lost: 0 bytes in 0 blocks indirectly lost: 0 bytes in 0 blocks possibly lost: 0 bytes in 0 blocks still reachable: 72,704 bytes in 1 blocks
 ==1908== suppressed: 0 bytes in 0 blocks
==1908== Rerun with --leak-check=full to see details of leaked memory
 ==1908== For counts of detected and suppressed errors, rerun with: -v
==1908== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
 -bash-4.2$
-bash-4.2$
-bash-4.2$
 -bash-4.2$
 -bash-4.2$
 -bash-4.2$
-bash-4.2$
-bash-4.2$
-bash-4.2$
 -bash-4.2$
 -bash-4.2$
 -bash-4.2$
 -bash-4.2$
-bash-4.2$
```

Here is my out put of a normal run with valgrind as well using ~45,000 words.

```
UnorderedArrayList& operator = (const UnorderedArrayList& a) = del
         if(size == capacity)//1
bool isEmpty()//T(N) = 3; O(1)
         return 0;//1
void insert(string word)//T(N) = 1002; O(1)
         \label{eq:bufsize} \begin{array}{l} \text{buf[size] = word;//1 + 1000(because of new?)} \\ \text{++size;//1} \end{array}
bool find(string word)//T(N) = 5N + 2; O(N)
         for (int i = 0; i < size; ++i)//1 + N + N
                  if(buf[i] == word)//N + N
         return 0;//1
void remove(string word)//T(N) = 6N + 1; O(N)
         for(int i = 0; i < size; ++i)//1 + N + N
void print(ostream& out)//T(N) = 3N + 1; O(N)
         for(int i = 0; i < size; ++i)//1 + N + N
         delete[] buf;//1000
```

Here is O(N) for the ArrayList methods. I wrote the T(N) to help me but O(N) is followed by that on the same line.

```
Putty
                 static void print(ostream& out, ListNode* L)//T(N) = 1000; O(1)
                          if(L)
                          return new ListNode(s, L);//1 + 1000
                 static ListNode* find(string s, ListNode* L)//T(N) = 4N + 2; O(N)
                                   //ListNode::print(cout, p);
                                  if(p->info == s)//N
                                           return p;//N
                          return nullptr;//1
                 static ListNode* remove(string s, ListNode* L)//T(N) = 6N + 10; O(N)
                          ListNode* p = L;//1
if(p == nullptr)//1
                                  return nullptr;///throw error 1
                          if(p->next == nullptr)//1
                                   if(p->info == s)//1
                                           return nullptr;//1
                                  return nullptr;//1
                          ListNode* prev = p;//1
while(p != nullptr)//N
                                  p = p->next;//N
                                   if(p->info == s)//N
                                           prev->next = p->next;//N
return p;//N
                          return nullptr;//1
        };
ListNode* head;
public:
        UnorderedLinkedList()
```

Here is the O(N) for the static functions in the Linked List class. I wrote the T(N) to help me but O(N) is followed by that on the same line.

```
bool isEmpty()//T(N) = 3; O(1)
void insert(string word)//T(N) = 1002; O(1)
         head = ListNode::insert(word, head);//1 + 1001
bool find(string word)//T(N) = 4N + 5; O(N)
         ListNode* temp;
temp = ListNode::find(word, head);//1 + 4N + 2
if(temp)
                 return 1;//1
         return 0;//1
void remove(string word)//T(N) = 6N + 1012; O(N)
         ListNode* temp;
temp = ListNode::remove(word, head);//1
         if(temp != nullptr)//1
                       te temp;//1000
         for(ListNode* p = head; p != nullptr; p = p->next)//1 + N + N
                  out << p->info << endl;//1000 + N
                  ListNode* next = head->next;//N
delete head;//1000N
head = next;//N
```

Here are the methods for the Linked List class. I wrote the T(N) to help me but O(N) is followed by that on the same line.

```
void insert_all_words(string file_name, UnorderedArrayList& L)//O(N)
          declare time object open file
           start timer
          for each word, w, in file L.insert(w);
           stop time
           close file
           report time
          Timer t;
double eTime;
           ifstream file;
file.open(file_name);
          string word;
while(file >> word)
                      L.insert(word);//1002N
cout << word << endl;
          t.elapsedUserTime(eTime);
cout << eTime << endl;
file.close();</pre>
void find all words(string file name, UnorderedArrayList& L)//O(N^2)
          Timer t;
double eTime;
ifstream file;
file.open(file_name);
t.start();
          string word;
while(file >> word)
                      //cout << word << endl;</pre>
                      L.find(word);//5N^2 + 2N
          t.elapsedUserTime(eTime);
cout << eTime << endl;
file.close();</pre>
 void remove_all_words(string file_name, UnorderedArrayList& L)//O(N^2)
          file.open(file_name);
t.start();
          string word;
while(file >> word)
                      L.remove(word);//6N^2 + N
           t.elapsedUserTime(eTime);
```

Here is the O(N) for the test functions of Array List.

```
void insert_all_words(string file_name, UnorderedLinkedList& L)//O(N)
          Timer t;
          double eTime;
          ifstream file;
          file.open(file_name);
t.start();
          string word;
while(file >> word)
          t.elapsedUserTime(eTime);
cout << eTime << endl;</pre>
          file.close();
void find all words(string file name, UnorderedLinkedList& L)//O(N^2)
          ///*
Timer t;
          double eTime; ifstream file;
          file.open(file_name);
t.start();
          string word;
while(file >> word)
                     //cout << L.find(word) << endl;
L.find(word);//4N^2 + 5N
          t.elapsedUserTime(eTime);
          cout << eTime << endl;
file.close();
void remove_all_words(string file_name, UnorderedLinkedList& L)//O(N^2)
          Timer t;
double eTime;
          ifstream file;
file.open(file_name);
          string word;
while(file >> word)
                     L.remove(word);//6N^2 + 1012N
          t.elapsedUserTime(eTime);
cout << eTime << endl;
file.close();</pre>
```

Here is the O(N) for the test functions of Array List.

End of Report