Burt Rosenberg

Problem Set 3

OUT: 5 OCTOBER, 1992
DUE: 12 OCTOBER, 1992

Reading Assignment

Read

- Chapter 3, Sections "Programming with Linked Lists" and "Variations of the Linked List".

Goals

More practice using pointers and linked lists: dummy header technique, searching a linked list and moving elements in a list. Introduction to code instrumentation.

Assignment

You will write two programs, modifications of last week's assignment. The first program, *ps3a.pas*, will use a linked list with a dummy header element to keep track of all the words in the file *datafile.dat* and for each word a count of how many times that word appears in *datafile.dat*. This program will be a modification of last week's *ps2.pas*. In *ps3b.pas*, the second program, we will attempt to improve the performance of *ps3a.pas* by using the move-to-front heuristic.

The output will include,

1. The words in the file, in reverse order of their first occurrence, and with each word, the number of times it appears in the file.

2. A count of the total number of words in the file.

3. A count of the total number of unique words in the file.

4. The total number of elements that were searched during the program.

An example will be given in a following subsection.

Assignment Step-by-step

Many people write too much code before testing. It is often hard to know where to break a large programming assignment into steps. The following step-by-step instructions will help you organize your progress.

1. Copy *[.ps2]ps2.pas* to *[.ps3]ps3a.pas* and modify the list subroutines to use a dummy header element. Test that the program gives the same outputs as ps2.pas.

2. Write functions

$$\text{eq\_string}(s,t) = \begin{cases} T & s = t, \text{ with } s,t \text{ of stringType} \\ F & \text{else.} \end{cases}$$

$$\text{search\_list}(l,s) = \begin{cases} p & p \wedge .\text{next} \wedge .\text{str} = s \\ \text{nil} & \text{else.} \end{cases}$$

   Change your main-line program to use search_list to insert an element only if it has not be found in the list. Test thoroughly before proceeding!

3. Finish ps3a.pas by writing subroutines,

```
procedure increment_count( p:listPntr ) ;
  {increments p^.next^.cnt}
function length_list( l:listPntr ) :integer ;
  {returns length of list l, except header}
function sum_count( l:listPntr ) :integer ;
  {returns sum of all p^.cnt in l, except header}
```

   Modify create_list and search_list to keep track of the total number of elements searched. Test your finished *ps3a.pas* on several inputs, both long and short.

4. Copy *ps3a.pas* to *ps3b.pas* and continue. Write,

```
procedure move_to_front( l,p:listPntr )
```

   which takes element p and from its current position in list l and moves it to the front. Modify the main-line program to use this procedure to implement the move-to-front search heuristic. Test this program on several inputs and compare the number of searched elements made by this program with the number made by ps3a.pas.

Example

Suppose the file datafile.dat was:

one fish two fish red fish blue fish

The output of ps3a.pas would be:

```
 8 words in the text
 5 unique, they are:
 1 blue
 1 red
 1 two
 4 fish
 1 one
27 elements searched
```

Let's go through this example in detail. We will keep the total number of elements searched in the count field of the dummy header. Before reading any words, the list is,

$$(\text{X}, 0)$$

Where we represent a record by *(str,cnt)*. The word "one" was read, the list was searched and one was inserted at its head. (We will count the dummy header element),

$$(\text{X}, 1) \longrightarrow (\text{one}, 1)$$

"Fish" is read, the list searched and fish inserted at its head. The list was two long before the insertion of fish, so the new number of total elements searched is three,

$$(\text{X}, 3) \longrightarrow (\text{fish}, 1) \longrightarrow (\text{one}, 1)$$

"Two" was inserted at the head after the unsuccesful search of a three element list so now,

$$(\text{X}, 6) \longrightarrow (\text{two}, 1) \longrightarrow (\text{fish}, 1) \longrightarrow (\text{one}, 1)$$

When the second "fish" was read, the program found that fish was the third element in the list,

$$(\text{X}, 9) \longrightarrow (\text{two}, 1) \longrightarrow (\text{fish}, 2) \longrightarrow (\text{one}, 1)$$

"Red" was tacked on the the front after an unsuccessful search of a 4 element list,

$$(X, 13) \longrightarrow (\text{red}, 1) \longrightarrow (\text{two}, 1) \longrightarrow (\text{fish}, 2) \longrightarrow (\text{one}, 1)$$

The third "fish" was found buried as the fourth element of the list,

$$(X, 17) \longrightarrow (\text{red}, 1) \longrightarrow (\text{two}, 1) \longrightarrow (\text{fish}, 3) \longrightarrow (\text{one}, 1)$$

"Blue" was added to the front after an unsuccessful search of a 5 element list,

$$(X, 22) \longrightarrow (\text{blue}, 1) \longrightarrow (\text{red}, 1) \longrightarrow (\text{two}, 1) \longrightarrow (\text{fish}, 3) \longrightarrow (\text{one}, 1)$$

The final "fish" was found 5 elements from the list root,

$$(X, 27) \longrightarrow (\text{blue}, 1) \longrightarrow (\text{red}, 1) \longrightarrow (\text{two}, 1) \longrightarrow (\text{fish}, 4) \longrightarrow (\text{one}, 1)$$

One this example, ps3b.pas would give the output:

```
8 words in the text
5 unique, they are:
4 fish
1 blue
1 red
1 two
1 one
24 elements searched
```