Lab 2: Debugging in MATLAB and A Quicksort Implementation

DD2325

November 7, 2007

1 Introduction

In this lab, you will

- learn how to do debugging, and
- implement a quicksort algorithm

in MATLAB.

2 Debugging In MATLAB

2.1 Theory

Check out the notes for debugging (or MATLAB help pages).

Two things you should learn is to set breakpoints in the program so that you can run the program controllably (that is you can stop execution at certain points) and examine values of variables at these points. This way you can follow if data is manipulated correctly or check if certain control points are ever reached.

2.2 Practice

Below an attempt to program a shellsort algorithm is given in pseudocode:

```
procedure shellsort(a)
int i, j, h, v;
h = 4;


while (h <= length(a))
   h=3* h+1
end


while (h > 1)
   h = h/3;
   for i h+1:length(a)
      v = a[i]; j = i;
      while a[j-h] > v
          a[j] = a[j-h];
          j = j- h;
          a[j] = v
```

```
      end
end
```

Program shellsort in MATLAB. Using the debugger try to see what is wrong with the program. This you should do by starting with a short array to be sorted. Then you should run the program on this array and check what happens to the array at certain positions by observing the array at breakpoints. Finally, fix the program so that it sorts any array.

3 A Quicksort Algorithm

In this section you will implement a part of the quicksort algorithm. The main part of the algorithm is given below as a MATLAB function. The `quicksort` function misses the partition function in order to execute.

```
function quicksort(l,r)
     if (r >= l)
       i = partition(l,r);
       quicksort(l,i-1);
       quicksort(i+1,r);
     end
```

3.1 Algorithm

You will implement the partition function in MATLAB. This is the part of the sorting algorithm which actually makes changes on the data sequence. Your implementation will assume that the number sequence is stored in the global vector `array`.

1. Choose pivot as the element at position p where p = (l + r) / 2
2. Exchange the pivot element with the last element (which is at position r)
3. Scan from left of the array until an element greater than the pivot element is found (or the two pointers cross).
4. Scan from right of the array until an element less than the pivot element is found (or the two pointers cross).
5. If the position of the first element (the one greater than pivot) is less than or equal to thr position of the second element (less than pivot), then exchange these two elements.
6. Repeat this process until the two pointers cross.
7. Exchange the pivot element and the element pointed to by (left and right) the pointer if this element is greater than pivot, if it is less, than exchange the pivot element with the element to the pointers right.
8. Return pivot position

3.2 Input

Your program will read input from a text file which is comprised of a sequence of natural numbers between 1 and 10000:

"4 99 500 45 ..."

The size of the file is not fixed. (Hint: check MATLAB programming demo for "Reading input files..")

The program will read these numbers into a vector named `array`.

3.3 Output

The main script of the program will simply call the quicksort function, which in turn will sort the vector and print the resulting vector on the screen.

3.4 Challenge

Does the above algorithm work when you have multiple instances of the same element? If not, how can you change partition function so that it works correctly event in this case?

3.5 Demos

Demonstrations will be done during lab hours. Please make sure you know how to use debugging features and explain the quicksort algorithm.