Samy Masadi

CSCI 313

10/18/2018

Bubble Sort Algorithm

- Allocate .space for three 10-integer arrays: 4 bytes * 10 = 40 bytes each
- for (int i = 0; i < 10; i++) { collect int from user; save to array1[i] }
- Perform bubble sort. Save to array2.
    - Outer loop: for (int i = 0; i < array.length – 1; i++) /* Use i not for iterating elements, but for determining the current limit for iterator j.*/
    - Set reg for i to 0. Set reg for i limit to 9.
    - Inner loop: for (int j = 1; j < array.length – i; j++) { if (array[j-1] < array [j]) {swap;}} /* Use j to iterate each element in array until current limit. */
    - Ie, first loop: j < 10. Start with comparing array[0] and array [1] . . . end with comparing array[8] and array[9]. After first loop, array[9] is considered sorted. Second loop: j < 9. Start with comparing array[0] and array[1] . . . end with comparing array[7] and array[8].
    - For each inner loop, set register for j to 1 every time. Set reg for current j limit to calculation of (10-i).
    - Load addresses of array[j-1] and array[j]
    - Load int words for array[j-1] and array[j]
    - Compare array[j-1] and array[j]. If array[j-1] < array[j], swap.
    - j++. If j >= current j limit, exit to outer loop. Otherwise, continue inner loop.
    - i++. If i >= 9, array sorted, exit outer loop. Otherwise, continue outer loop.

- Java bubble sort code referenced:

```
static void bubblesort(int[] A) {
        for (int i=0; i<A.length-1; i++)
                for (int j=1; j<A.length-i; j++)
                        if (compareTo(A[j-1], A[j]))
                                swap(A, j-1, j);
}

static boolean compareTo(int i, int j) {
        if (i < j) {
                return true;
        }
        else
                return false;
}

static void swap(int[] A, int smallIndex, int lastIndex) {
        int temp = A[lastIndex];
        A[lastIndex] = A[smallIndex];
        A[smallIndex] = temp;
}
```

- For swap function, load addresses for array[j-1] and array[j]. Load int words from array[j-1] and array[j]. Store word array[j-1] to address array[j]. Store word array[j] to address array[j-1].
- For efficiency, reverse order of already sorted array instead of doing a second bubble sort. The complexity of two bubble sorts: $n^2 + n^2$. The complexity of bubble sort followed by reverse copy: $n^2 + n$.
- Iterate from end of array2 while iterating from beginning of array3. Copy elements from end of sorted array2 to beginning of array3 to create sorted array in reverse order.
- Print all three arrays for user.