Sorting

Problem: Given an array of items that contain keys that can be ordered, arrange the items so they are in increasing order of their keys.

There are many solutions. Comparison sorts compare keys to put items in order. Bubblesort, selection sort and insertion sort require $O(n^2)$ execution time in the worst case. Heapsort, Mergesort, and Quicksort require $O(n \lg n)$ execution time. Some algorithms that work under special conditions and don’t use comparisons can sort in $O(n)$ time.
Bubble Sort

Bubble Sort compares adjacent array elements and, if necessary, swaps the larger element into the array position with the larger index.

Example: TBD
Algorithm 1

bubblesort(array, size)
1. for i = 0 to size-1 do
2.   for j = 0 to size-2 do
3.     if array[j] > array[j+1] then
4.       swap(array[j], array[j+1]);
Algorithm 2

Exploits the fact that the end of the array is sorted. Where is the sorted part?

```plaintext
bubblesort(array, size)
1. for i = 0 to size-1 do
2.   for j = 0 to size-2-i do
3.     if array[j] > array[j+1] then
4.       swap(array[j], array[j+1]);
```
Algorithm 3

Also exploits the fact that the array is sorted if no swaps are needed.

```
bubblesort(array, size)
1. swapflag ← true;
2. i ← 0;
3. while (swapflag and i < size-1) do
4.   swap ← false;
5.   for j = 0 to size-2-i do
6.     if array[j] > array[j+1]
7.       then
8.         swap(array[j], array[j+1]);
9.         swapflag ← true;
```

C++ Code is in the textbook, pages 471-472.
Efficiency

- Bubble Sort on an array of size $n$ requires time proportional to $n^2$, or $O(n^2)$. (Doubling the problem multiplies the required time by four. Tripling the size of the problem, multiplies the required time by nine.)

- The solution to a problem of size $n$ requires $a_1 + a_2 n$ space to solve, where $a_1$ and $a_2$ are constants, or $O(n)$.
Selection Sort

Selection Sort repeatedly finds the minimum element in the array and swaps it into the first unsorted position in the array.

Efficiency: Selection sorting an array of size n requires $O(n^2)$ time and $O(n)$ space.

Example: TBD
Algorithm

selectionsort(array, size)
1. for i= 0 to size-1 do
2.     min = array[i];
3.     minIndex = i;
4.     for j= i+1 to size-1 do
5.         if min > array[j]
6.             then
7.                 min = array[j];
8.                 minIndex = j;
9.             array[minIndex] ← array[i];
10.            array[i] = min;