void selectionSort(int A[], int n) {
    int cur, j, min;

    // Loop through each index of the array. At each loop iteration
    // we will be placing the smallest unplaced item left in this
    // location of the array.
    for (cur = 0; cur < n; cur++) {

        // At first, the smallest unplaced element is at cur
        ________________;

        // Look through the rest to find a value < list[cur]
        // If we find one, update WHERE it is located, min
        for (__________________) {
            // We found a smaller element!
            if (__________________)
                ________________;
        }

        // Now, swap A[min] into its sorted
        // location, A[cur].
        ________________;
    }
}

void insertionSort(int A[], int n) {
    int i, j;

    // Loop through each element to insert.
    for (i=1; i<n; i++) {
        j=i;

        // Continue swapping the element with
        // its left neighbor, until it hits the correct
        // location in the sorted elements.
        while (j > 0 && ________________) {
            // Example of pointer arithmetic.
            ________________;
        }
    }
}
void bubbleSort(int A[], int n) {
    int i, j;
    // Loop through each element, if two consecutive
    // elements are out of order swap them.
    for (i=n-2; i>=0; i--) {
        for (________________________)
            if (________________________)
                __________________________;
    }
}

// Swaps the integers pointed to by a and b.
void swap(int *a, int *b) {
    int temp = *a;
    *a = *b;
    *b = temp;
}

What’s the limitation of sorts that only swap adjacent elements?