CS 372/469, Fall 2018
Code listing for sorting algorithms

September 17, 2018

Algorithm 1: Insertion sort

Input : Unsorted array A
Output: A with sorted elements

```
1  for j = 2 to A.length do
2      key = A[j]
       /* Insert A[j] into sorted sequence A[1..j − 1]                    */
3      i = j − 1
4      while i > 0 and A[i] > key do
5          A[i + 1] = A[i]
6          i = i − 1
7      end
8      A[i + 1] = key
9  end
10 return A
```

Algorithm 2: Quick sort

```
1  Function Quicksort (A[p..r])
       Input : Unsorted array A[p..r]
       Output: A with sorted elements
2      if p < r then
3          q = Partition(A, p, r)
4          Quicksort(A, p, q − 1)
5          Quicksort(A, q + 1, r)
6      end
7  return A
```

To sort an entire array, initial call will be Quicksort$(A, 1, A.length)$.

Algorithm 3: Partition

Input : A, p, r
Output: Index of pivot element

```
1  x = A[r]
2  i = p − 1
3  for j = p to r − 1 do
4  |  if A[j] ≤ x then
5  |  |  i = i + 1
6  |  |  swap(A[i], A[j])
7  |  end
8  end
9  swap(A[i + 1], A[r])
10 return i + 1
```

Algorithm 4: Counting sort

Input : Unsorted array A, Empty array B for final, sorted output
Output: B

```
   /* Create working array C                                   */
1  let C[0..k] be a new array
   /* Initialize C to all zeroes                               */
2
3  for i = 0 to k do
4  |  C[i] = 0
5  end
   /* Values in C are the number of times each index of C occurs in A
      */
6  for j = 1 to A.length do
7  |  C[A[j]] = C[A[j]] + 1
8  end
   /* Cumulatively increment values in C                       */
9
10 for i = 1 to k do
11 |  C[i] = C[i] + C[i − 1]
12 end
   /* Now, populate B; see whiteboard for how the below steps work   */
13 for j = A.length to 1 do
14 |  B[C[A[j]]] = A[j]
15 |  C[A[j]] = C[A[j]] − 1
16 end
17 return B
```

References

[1] T. Cormen, C. Leicerson, R. Rivest and C. Stein. *Introduction to Algorithms*. MIT Press, Cambridge, Massachusetts, 2009.