Tutorial 5 Exercises

COMP526: Efficient Algorithms

3–4 November, 2024

Exercise 1. Suppose an array a is "almost sorted" in the sense that if a stores the values $c_0 \leq c_0 \leq c_3 \leq \cdots \leq c_{n-1}$, then $c_i = a[j]$ where $\left|j - i\right| \leq k$. That is, the final (sorted) index of each value in a is no more than k from its initial index in a. Argue that on input a, the InsertionSort algorithm will terminate after at most $O(nk)$ steps.

```
procedure InsertionSort(a, n)
    for i = 1, 2, ..., n − 1 do
        j ← i
        while j > 0 and a[j] < a[j − 1] do
            Swap(a, j, j − 1)
            j ← j − 1
        end while
    end for
end procedure
```

Exercise 2. Suppose we are given two arrays a and b of size n that store two distinct permutations of $\{1, 2, \ldots, n\}$. That is, both a and b store each of the numbers from 1 to n, but the two arrays differ in their values at at least one index. Consider a sequence of swap operations $S_1, S_2, \ldots, S_m$ that are applied to both a and b, where each S_i swaps the values at two indices of the array it is applied to. Argue that after performing the swap operations, a and b are still distinct. In particular, the same sequence of swaps cannot sort both arrays.

Exercise 3. Suppose we apply RadixSort to an array a of size n that stores n distinct numerical values. Explain why in this scenario the running time of RadixSort is $\Omega(n \log n)$.

Exercise 4. Suppose a function T satisfies the recursion relation

$$T(n) = T(cn) + O(n) \quad \text{for some } c \text{ satisfying } \frac{1}{2} \leq c < 1.$$

for $n \geq 1$, and $T(1) = O(1)$. Argue that $T(n) = O(n)$. You may find the following fact useful: for any value of $a < 1$, we have

$$\sum_{i=0}^{k} a^i = 1 + a + a^2 + \cdots, + a^k = \frac{1 - a^{-k-1}}{1 - a} < \frac{1}{1 - a}. \tag{1}$$