



Algorithms and Data Structures – Conditional Course

Winter Term 2026/27

Exercise Sheet 3

Due: Wednesday, 11 November 2026, 12:00

Exercise 1: Bucket Sort

(7 Points)

Bucket Sort is a simple algorithm for *stably* sorting an array $A[0 \dots n-1]$ of n data items whose sorting keys lie in $\{0, \dots, k\}$. A function `key` assigns each item $x \in A$ a key `key(x)  $\in \{0, \dots, k\}$` .

First, create an array $B[0 \dots k]$ of FIFO queues,¹ where $B[i]$ is the i th queue. Iterate through A . If the current item $A[j]$ has key $i := \text{key}(A[j])$, append it to $B[i]$ using `enqueue`.

Finally, empty the queues $B[0], \dots, B[k]$ using `dequeue`, writing the returned items back to A in that order. The array is now sorted by `key`. Items x, y with equal keys retain their relative order: this is what makes the sort *stable*.

Implement Bucket Sort from this description using `BucketSort.py`. FIFO queue implementations are supplied in `Queue.py` and `ListElement.py`.²

¹In a FIFO (First In, First Out) queue, items are removed in the order in which they were inserted.

²You may instead use a library queue, but note that its method names may differ.

Exercise 2: Radix Sort

(13 Points)

Let $A[0 \dots n-1]$ contain integers from $\{0, 1, \dots, k\}$ for some $k \in \mathbb{N}$. Our implementation of *Radix Sort* uses the stable Bucket Sort algorithm as a subroutine.

Let $m = \lfloor \log_b k \rfloor$. Represent every key $x \in A$ in base b as $x = \sum_{i=0}^m c_i b^i$. You may use $b = 10$, so that the c_i are decimal digits. First use Bucket Sort to sort by the least significant digit c_0 , then by c_1 , and so on up to c_m .³ The resulting array is sorted.

Example: Let $A = [329, 457, 657, 839, 436, 720, 355, 009]$ and $b = 10$. Sorting by c_0 gives

$$A = [720, 355, 436, 457, 657, 329, 839, 009].$$

Stability preserves the relative order of items with the same digit: for example, 839 still precedes 009. Sorting by c_1 gives

$$A = [009, 720, 329, 436, 839, 355, 457, 657].$$

Finally, sorting by c_2 gives

$$A = [009, 329, 355, 436, 457, 657, 720, 839].$$

- (a) Implement Radix Sort from this description, using your Bucket Sort implementation as a subroutine. (7 Points)
- (b) Compare the running times of your Bucket Sort and Radix Sort implementations. Use arrays of size $n = 10^4$ containing random keys from $\{0, 1, \dots, k\}$. Plot the running times for $k \in \{2i \cdot 10^4 \mid i = 1, \dots, 60\}$; helper functions are provided in the templates. Briefly discuss the results in `experience.txt`. (3 Points)
- (c) Give and justify the asymptotic running times of both implementations as functions of n and k . (3 Points)

³The i th digit of x in base b is $c_i = (x \bmod b^{i+1}) \operatorname{div} b^i$, where `div` denotes integer division.