## 4.4 Quicksort

```
Function quickSort(s: Sequence of Element) : Sequence of Element
    if |s| ≤ 1 then return s                                 // base case
        else
            pick p ∈ s uniformly at random                   // pivot key
            a := ⟨e ∈ s : e < p⟩                             // (A)
            b := ⟨e ∈ s : e = p⟩                             // (B)
            c := ⟨e ∈ s : e > p⟩                             // (C)
            return append(a,b,c)
```

Figure 4.3: Quicksort

Quicksort is a divide-and-conquer algorithm that is complementary to the merge-sort algorithm we have seen in Section 4.2. Quicksort does all the difficult work *before* the recursive calls. The idea is to distribute the input elements to two or more sequences that represent disjoint ranges of key values. Then it suffices to sort the shorter sequences recursively and to concatenate the results. To make the symmetry to mergesort complete, we would like to split the input into two sequences of equal size. Since this would be rather difficult, we use the easier approach to pick a random splitter elements or *pivot* $p$. Elements are classified into three sequences $a$, $b$, or $c$ of elements that are smaller, equal to, or larger than $p$ respectively. Figure 4.3 gives a high level realization of this idea. This simple algorithm is already enough to show expected execution time $O(n \log n)$ in Section 4.4.1.

### 4.4.1 Analysis

To analyze the running time of quicksort for an input sequence $s = \langle e_1, \ldots, e_n \rangle$ we focus on the number of element comparisons performed (we allow three outcomes of a comparison here — smaller, equal, and larger). Other operations contribute only constant factors and small additive terms in the execution time.

**Theorem 4.4** *The expected number of comparisons performed by quicksort is*

$$\bar{C}(n) \leq 2n \ln n \leq 1.4 n \log n \ .$$

We concentrate on the case that all elements are different. Other cases are easier because a pivot that occurs several times results in a larger middle sequence $b$ that need not be processed any further.

Let $s' = \langle e'_1, \dots, e'_j \rangle$ denote the elements of the input sequence in sorted order. Note that elements $e'_i$ and $e'_j$ are compared only if one of them is picked as a pivot. Hence, we can count comparisons by looking at the random variables $X_{ij}$ where $X_{ij} = 1$ if $e'_i$ and $e'_j$ are compared and $X_{ij} = 0$ otherwise. $X_{ij}$ is an *indicator random variable* because it indicates that $e'_i$ and $e'_j$ are compared. Indicator variables are a useful tool for the probabilistic analysis of algorithms because they encode the behaviour of complex algorithms into very simple mathematical objects. In our case, we get

$$\bar{C}(n) = \mathrm{E}[\sum_{i=1}^{n} \sum_{j=i+1}^{n} X_{ij}] = \sum_{i=1}^{n} \sum_{j=i+1}^{n} \mathrm{E}[X_{ij}] = \sum_{i=1}^{n} \sum_{j=i+1}^{n} \mathrm{prob}(X_{ij} = 1) \quad .$$

The middle transformation uses the linearity of expectation (Equation (**??**)). The last transformation follows from the definition of the expectation of a random variable (Equation (**??**)) by observing that $\mathrm{E}[X_{ij}] = 0 \cdot \mathrm{prob}(X_{ij} = 0) + 1 \cdot \mathrm{prob}(X_{ij} = 1)$.

Before we can further simplify the expression for $\bar{C}(n)$, we need to find $\mathrm{prob}(X_{ij} = 1)$.

**Lemma 4.5** *For any $i < j$,* $\mathrm{prob}(X_{ij} = 1) = \dfrac{2}{j - i + 1}$.

**Proof:** Consider the $j - i + 1$ element set $M = \left\{ e'_k : i \le k \le j \right\}$. As long as no pivot from $M$ is selected, $e'_i$ and $e'_j$ are not compared but all elements from $M$ are passed to the same recursive calls. Eventually, a pivot $p$ from $M$ is selected. Each element in $M$ has the same chance $1/|M|$ to be selected. If $p = e'_i$ or $p = e'_j$ we have $X_{ij} = 1$. The probability for this event is $2/|M| = 2/(j - 1 + 1)$. Otherwise, $e'_i$ and $e'_j$ are passed to different recursive calls so that they will never be compared. ∎

Now we can finish the proof of Theorem 4.4 using relatively simple calculations.

$$\bar{C}(n) = \sum_{i=1}^{n} \sum_{j=i+1}^{n} \mathrm{prob}(X_{ij} = 1) = \sum_{i=1}^{n} \sum_{j=i+1}^{n} \frac{2}{j - i + 1} = 2 \sum_{i=1}^{n} \sum_{j=i+1}^{n} \frac{1}{j - i + 1} =$$
$$2 \sum_{i=1}^{n} \sum_{k=2}^{n-i+1} \frac{1}{k} \le 2 \sum_{i=1}^{n} \sum_{k=2}^{n} \frac{1}{k} = 2 \cdot n \sum_{k=2}^{n} \frac{1}{k}$$

Since sums of reciprocals occur so often they have been given a name — $H_n = \sum_{k=1}^{n} 1/k$ is called the $n$-th *harmonic number*. By Equation **??** that we ask you to verify in Exercise 4.20, we have $H_n \le \ln n + 1$ so that we get

$$\bar{C}(n) \le n(H_n - 1) \le n \ln n$$

∎