8.4 Bucket sort

Bucket sort runs in linear time when the input is drawn from a uniform distribution. Like counting sort, bucket sort is fast because it assumes something about the input. Whereas counting sort assumes that the input consists of integers in a small range, bucket sort assumes that the input is generated by a random process that distributes elements uniformly over the interval $[0, 1)$. (See Section C.2 for a definition of uniform distribution.)

The idea of bucket sort is to divide the interval $[0, 1)$ into n equal-sized subintervals, or ***buckets***, and then distribute the n input numbers into the buckets. Since the inputs are uniformly distributed over $[0, 1)$, we don't expect many numbers to fall into each bucket. To produce the output, we simply sort the numbers in each bucket and then go through the buckets in order, listing the elements in each.

Our code for bucket sort assumes that the input is an n-element array A and that each element $A[i]$ in the array satisfies $0 \leq A[i] < 1$. The code requires an auxiliary array $B[0 \mathinner{..} n-1]$ of linked lists (buckets) and assumes that there is a mechanism for maintaining such lists. (Section 10.2 describes how to implement basic operations on linked lists.)

```
BUCKET-SORT(A)
1  $n \leftarrow length[A]$
2  for $i \leftarrow 1$ to $n$
3       do insert $A[i]$ into list $B[\lfloor nA[i] \rfloor]$
4  for $i \leftarrow 0$ to $n - 1$
5       do sort list $B[i]$ with insertion sort
6  concatenate the lists $B[0], B[1], \ldots, B[n-1]$ together in order
```

Figure 8.4 shows the operation of bucket sort on an input array of 10 numbers.

To see that this algorithm works, consider two elements $A[i]$ and $A[j]$. Assume without loss of generality that $A[i] \leq A[j]$. Since $\lfloor nA[i] \rfloor \leq \lfloor nA[j] \rfloor$, element $A[i]$ is placed either into the same bucket as $A[j]$ or into a bucket with a lower index. If $A[i]$ and $A[j]$ are placed into the same bucket, then the **for** loop of lines 4–5 puts them into the proper order. If $A[i]$ and $A[j]$ are placed into different buckets, then line 6 puts them into the proper order. Therefore, bucket sort works correctly.

To analyze the running time, observe that all lines except line 5 take $O(n)$ time in the worst case. It remains to balance the total time taken by the n calls to insertion sort in line 5.

To analyze the cost of the calls to insertion sort, let n_i be the random variable denoting the number of elements placed in bucket $B[i]$. Since insertion sort runs in quadratic time (see Section 2.2), the running time of bucket sort is

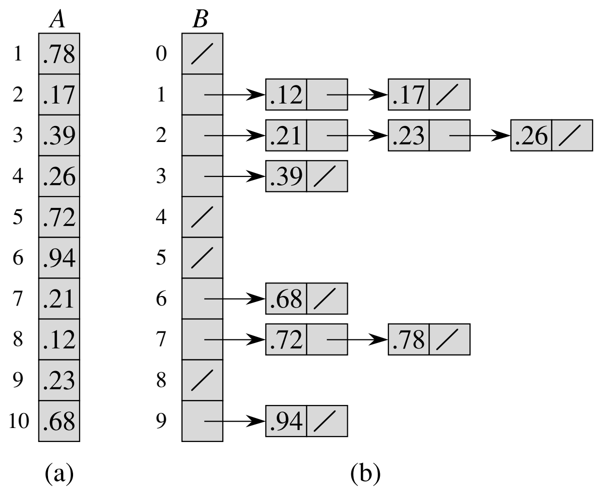

Figure 8.4 The operation of BUCKET-SORT. **(a)** The input array $A[1..10]$. **(b)** The array $B[0..9]$ of sorted lists (buckets) after line 5 of the algorithm. Bucket i holds values in the half-open interval $[i/10, (i+1)/10)$. The sorted output consists of a concatenation in order of the lists $B[0], B[1], \ldots, B[9]$.

$$T(n) = \Theta(n) + \sum_{i=0}^{n-1} O(n_i^2) .$$

Taking expectations of both sides and using linearity of expectation, we have

$$\begin{aligned}
\mathrm{E}[T(n)] &= \mathrm{E}\left[\Theta(n) + \sum_{i=0}^{n-1} O(n_i^2)\right] \\
&= \Theta(n) + \sum_{i=0}^{n-1} \mathrm{E}[O(n_i^2)] \qquad \text{(by linearity of expectation)} \\
&= \Theta(n) + \sum_{i=0}^{n-1} O(\mathrm{E}[n_i^2]) \qquad \text{(by equation (C.21))} . \qquad (8.1)
\end{aligned}$$

We claim that

$$\mathrm{E}[n_i^2] = 2 - 1/n \qquad (8.2)$$

for $i = 0, 1, \ldots, n-1$. It is no surprise that each bucket i has the same value of $\mathrm{E}[n_i^2]$, since each value in the input array A is equally likely to fall in any bucket. To prove equation (8.2), we define indicator random variables

$$X_{ij} = \mathrm{I}\{A[j] \text{ falls in bucket } i\}$$

for $i = 0, 1, \ldots, n-1$ and $j = 1, 2, \ldots, n$. Thus,

$$n_i = \sum_{j=1}^{n} X_{ij} \; .$$

To compute $\mathrm{E}[n_i^2]$, we expand the square and regroup terms:

$$\begin{aligned}
\mathrm{E}[n_i^2] &= \mathrm{E}\left[\left(\sum_{j=1}^{n} X_{ij}\right)^2\right] \\
&= \mathrm{E}\left[\sum_{j=1}^{n}\sum_{k=1}^{n} X_{ij}X_{ik}\right] \\
&= \mathrm{E}\left[\sum_{j=1}^{n} X_{ij}^2 + \sum_{1\le j\le n}\;\sum_{\substack{1\le k\le n \\ k\ne j}} X_{ij}X_{ik}\right] \\
&= \sum_{j=1}^{n} \mathrm{E}\left[X_{ij}^2\right] + \sum_{1\le j\le n}\;\sum_{\substack{1\le k\le n \\ k\ne j}} \mathrm{E}[X_{ij}X_{ik}] \; , \qquad (8.3)
\end{aligned}$$

where the last line follows by linearity of expectation. We evaluate the two summations separately. Indicator random variable X_{ij} is 1 with probability $1/n$ and 0 otherwise, and therefore

$$\begin{aligned}
\mathrm{E}\left[X_{ij}^2\right] &= 1\cdot\frac{1}{n} + 0\cdot\left(1-\frac{1}{n}\right) \\
&= \frac{1}{n} \; .
\end{aligned}$$

When $k \ne j$, the variables X_{ij} and X_{ik} are independent, and hence

$$\begin{aligned}
\mathrm{E}[X_{ij}X_{ik}] &= \mathrm{E}[X_{ij}]\,\mathrm{E}[X_{ik}] \\
&= \frac{1}{n}\cdot\frac{1}{n} \\
&= \frac{1}{n^2} \; .
\end{aligned}$$

Substituting these two expected values in equation (8.3), we obtain

$$\begin{aligned}
\mathrm{E}[n_i^2] &= \sum_{j=1}^{n}\frac{1}{n} + \sum_{1\le j\le n}\;\sum_{\substack{1\le k\le n \\ k\ne j}} \frac{1}{n^2} \\
&= n\cdot\frac{1}{n} + n(n-1)\cdot\frac{1}{n^2} \\
&= 1 + \frac{n-1}{n} \\
&= 2 - \frac{1}{n} \; ,
\end{aligned}$$

which proves equation (8.2).

Using this expected value in equation (8.1), we conclude that the expected time for bucket sort is $\Theta(n) + n \cdot O(2 - 1/n) = \Theta(n)$. Thus, the entire bucket sort algorithm runs in linear expected time.

Even if the input is not drawn from a uniform distribution, bucket sort may still run in linear time. As long as the input has the property that the sum of the squares of the bucket sizes is linear in the total number of elements, equation (8.1) tells us that bucket sort will run in linear time.

Exercises

8.4-1

Using Figure 8.4 as a model, illustrate the operation of BUCKET-SORT on the array $A = \langle .79, .13, .16, .64, .39, .20, .89, .53, .71, .42 \rangle$.

8.4-2

What is the worst-case running time for the bucket-sort algorithm? What simple change to the algorithm preserves its linear expected running time and makes its worst-case running time $O(n \lg n)$?

8.4-3

Let X be a random variable that is equal to the number of heads in two flips of a fair coin. What is $\mathrm{E}[X^2]$? What is $\mathrm{E}^2[X]$?

8.4-4 ⋆

We are given n points in the unit circle, $p_i = (x_i, y_i)$, such that $0 < x_i^2 + y_i^2 \le 1$ for $i = 1, 2, \ldots, n$. Suppose that the points are uniformly distributed; that is, the probability of finding a point in any region of the circle is proportional to the area of that region. Design a $\Theta(n)$ expected-time algorithm to sort the n points by their distances $d_i = \sqrt{x_i^2 + y_i^2}$ from the origin. (*Hint:* Design the bucket sizes in BUCKET-SORT to reflect the uniform distribution of the points in the unit circle.)

8.4-5 ⋆

A ***probability distribution function*** $P(x)$ for a random variable X is defined by $P(x) = \Pr\{X \le x\}$. Suppose that a list of n random variables $X_1, X_2, \ldots, X_n$ is drawn from a continuous probability distribution function P that is computable in $O(1)$ time. Show how to sort these numbers in linear expected time.