CS 240: Data Structures and Data Management
Module 3 Study Guide

Taylor J. Smith — Spring 2017

Key Concepts

- **Randomized algorithms** take a set of random numbers R in addition to the usual input I. Since we can't predict random numbers in advance, we can't do a deterministic analysis, so we use the **expected case**.
- The expected running time of an algorithm is $T^{(exp)}(I) = \mathrm{E}[T(I,R)] = \sum_R T(I,R) \cdot \Pr[R]$.
- **Selection problem**: given an unsorted array A containing n values, find the element in position k of the sorted array, where $0 \leq k < n$.
  - Heap-based solution – $\Theta(n + k \log(n))$
  - Median selection solution – $\Theta(n \log(n))$
  - QuickSelect – $\Theta(n^2)$ worst case, $\Theta(n)$ best/average case
  - RandomizedQuickSelect – $\Theta(n)$ expected case
- We can sort values using the same ideas we used for QuickSelect.
  - QuickSort – $\Theta(n^2)$ worst case, $\Theta(n \log(n))$ best/average case
  - RandomizedQuickSort – $\Theta(n \log(n))$ expected case
- Comparison-based sorting (comparing values directly using $<$, $>$, $=$, etc.) has a lower bound of $\Omega(n \log(n))$.
- Non-comparison-based sorting can perform as good as $O(n)$.
  - CountingSort – $\Theta(n + R)$, where R is the radix (base) of the values
  - MSDRadixSort/LSDRadixSort – $\Theta(m(n + R))$, where m is the length of the values
  - (We assume that LSDRadixSort uses CountingSort, but any **stable sort** will work)

Suggested Readings

- **Sedgewick:** 5.2 (Divide and Conquer), Chapter 7 (Quicksort), 10.3 (MSD Radix Sort), 10.5 (LSD Radix Sort), 10.6 (Performance Characteristics of Radix Sorts)
- **CLRS:** Chapter 7 (Quicksort), Chapter 8 (Sorting in Linear Time)
- **Goodrich/Tamassia:** 4.3 (Quick-Sort), 4.4 (A Lower Bound on Comparison-Based Sorting), 4.5 (Bucket-Sort and Radix-Sort)

Practice Questions

Sedgewick

5.16. Write a recursive program that finds the maximum element in an array, based on comparing the first element in the array against the maximum element in the rest of the array (computed recursively).

7.1. Show how quicksort sorts the sequence E A S Y Q U E S T I O N.

7.4. Develop a stable quicksort for linked lists.

7.5. What is the maximum number of times during the execution of quicksort that the largest element can be moved, for a sequence of n elements?

CLRS

7-2. An alternative analysis of the running time of randomized quicksort focuses on the expected running time of each individual recursive call to QUICKSORT, rather than on the number of comparisons performed.

(a) Argue that, given an array of size n, the probability that any particular element is chosen as the pivot is $1/n$. Use this to define indicator random variables $X_i = \mathrm{I}\{i\text{th smallest element is chosen as the pivot}\}$. What is $\mathrm{E}[X_i]$?

(b) Let $T(n)$ be a random variable denoting the running time of quicksort on an array of size n. Argue that

$$\mathrm{E}[T(n)] = \mathrm{E}\left[\sum_{q=1}^{n} X_q(T(q-1) + T(n-q) + \Theta(n))\right].$$

(c) Show that the equation given in (b) can be rewritten as

$$\mathrm{E}[T(n)] = \frac{2}{n}\sum_{q=2}^{n-1} \mathrm{E}[T(q)] + \Theta(n).$$

(d) Show that

$$\sum_{k=2}^{n-1} k\log(k) \le \frac{1}{2}n^2\log(n) - \frac{1}{8}n^2.$$

(*Hint:* Split the summation into two parts, one for $k = 2, 3, \ldots, \lceil n/2 \rceil - 1$ and one for $k = \lceil n/2 \rceil, \ldots, n-1$.)

(e) Using the bound from the equation given in (d), show that the recurrence in the equation given in (c) has the solution $\mathrm{E}[T(n)] = \Theta(n\log(n))$. (*Hint:* Show, by substitution, that $\mathrm{E}[T(n)] \le an\log(n)$ for sufficiently large n and for some positive constant a.)

7-5. One way to improve the RANDOMIZEDQUICKSORT procedure is to partition around a pivot that is chosen more carefully than by picking a random element from the subarray. One common approach is the *median-of-3* method: choose the pivot as the median (middle element) of a set of 3 elements randomly selected from the subarray. For this problem, let us assume that the elements in the input array $A[1..n]$ are distinct and that $n \ge 3$. We denote the sorted output array by $A'[1..n]$. Using the median-of-3 method to choose the pivot element x, define $p_i = \Pr\{x = A'[i]\}$.

(a) Give an exact formula for p_i as a function of n and i for $i = 2, 3, \ldots, n-1$. (Note that $p_1 = p_n = 0$.)

(b) By what amount have we increased the likelihood of choosing the pivot as $x = A'[\lfloor (n+1)/2 \rfloor]$, the median of $A[1..n]$, compared to the ordinary implementation? Assume that $n \to \infty$, and give the limiting ratio of these probabilities.

(c) If we define a "good" split to mean choosing the pivot as $x = A'[i]$, where $n/3 \le i \le 2n/3$, by what amount have we increased the likelihood of getting a good split compared to the ordinary implementation? (*Hint:* Approximate the sum by an integral.)

(d) Argue that in the $\Omega(n\log(n))$ running time of quicksort, the median-of-3 method affects only the constant factor.

8.1-1. What is the smallest possible depth of a leaf in a decision tree for a comparison sort?

8.2-1. Using Figure 8.2 as a model, illustrate the operation of CountingSort on the array $A = \langle 6, 0, 2, 0, 1, 3, 4, 6, 1, 3, 2\rangle$.

```
COUNTING-SORT(A, B, n, k)
  let C[0..k] be a new array
  for i = 0 to k
      C[i] = 0
  for j = 1 to n
      C[A[j]] = C[A[j]] + 1
  for i = 1 to k
      C[i] = C[i] + C[i − 1]
  for j = n downto 1
      B[C[A[j]]] = A[j]
      C[A[j]] = C[A[j]] − 1
```

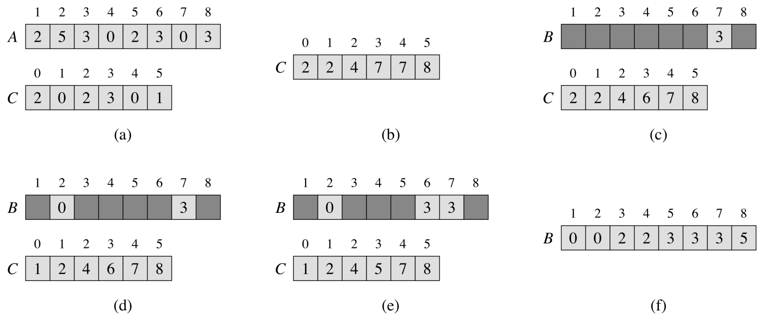

Figure 8.2: The operation of CountingSort on an input array $A[1..8]$, where each element of A is a nonnegative integer no larger than $k = 5$. (a) The array A and the auxiliary array C after line 5. (b) The array C after line 7. (c)-(e) The output array B and the auxiliary array C after one, two, and three iterations of the loop in lines 8–10, respectively. Only the lightly shaded elements of array B have been filled in. (f) The final sorted output array B.

8.2-4. Describe an algorithm that, given n integers in the range 0 to k, preprocesses its input and then answers any query about how many of the n integers fall into a range $[a..b]$ in $O(1)$ time. Your algorithm should use $\Theta(n + k)$ preprocessing time.

8.3-1. Using Figure 8.3 as a model, illustrate the operation of RadixSort on the following list of English words: COW, DOG, SEA, RUG, ROW, MOB, BOX, TAB, BAR, EAR, TAR, DIG, BIG, TEA, NOW, FOX.

```
RADIX-SORT(A, d)
  for i = 1 to d
      use a stable sort to sort array A on digit i
```

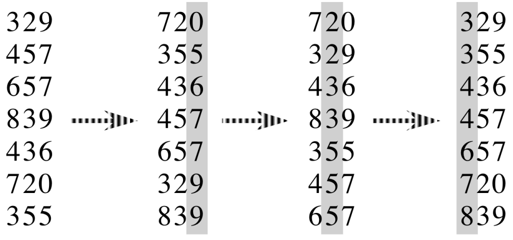

Figure 8.3: The operation of RadixSort on a list of seven 3-digit numbers. The leftmost column is the input. The remaining columns show the list after successive sorts on increasingly significant digit positions. Shading indicates the digit position sorted on to produce each list from the previous one.

8.3-4. Show how to sort n integers in the range 0 to $n^2 - 1$ in $O(n)$ time.

8-2. Suppose that we have an array of n data records to sort and that the key of each record has the value 0 or 1. An algorithm for sorting such a set of records might possess some subset of the following three desirable characteristics:

1. The algorithm runs in $O(n)$ time.
2. The algorithm is stable.
3. The algorithm sorts in place, using no more than a constant amount of storage space in addition to the original array.

(a) Give an algorithm that satisfies criteria 1 and 2 above.

(b) Give an algorithm that satisfies criteria 1 and 3 above.

(c) Give an algorithm that satisfies criteria 2 and 3 above.

(d) Can any of your sorting algorithms from parts (a)-(c) be used to sort n records with b-bit keys using RadixSort in $O(bn)$ time? Explain how or why not.

(e) Suppose that the n records have keys in the range from 1 to k. Show how to modify CountingSort so that the records can be sorted in place in $O(n + k)$ time. You may use $O(k)$ storage outside the input array. Is your algorithm stable? (*Hint:* How would you do it for $k = 3$?)

8-3. (a) You are given an array of integers, where different integers may have different numbers of digits, but the total number of digits over *all* the integers in the array is n. Show how to sort the array in $O(n)$ time.

(b) You are given an array of strings, where different strings may have different numbers of characters, but the total number of characters over all the strings is n. Show how to sort the strings in $O(n)$ time. (Note that the desired order here is the standard alphabetical order; for example, $\texttt{a} < \texttt{ab} < \texttt{b}$.)

Goodrich/Tamassia

R-4.9. Suppose we modify the deterministic version of the quicksort algorithm so that, instead of selecting the first element in an n-element sequence as the pivot, we choose the element at rank (index) $\lfloor n/2 \rfloor$, that is, an element in the middle of the sequence. What is the running time of this version of quicksort on a sequence that is already sorted?

R-4.11. Show that the best-case running time of quicksort on a sequence of size n with distinct elements is $O(n \log(n))$.

R-4.15. Describe a radix-sort method for lexicographically sorting a sequence S of triplets (k, l, m), where k, l, and m are integers in the range $[0..n-1]$, for some $n \geq 2$. How could this scheme be extended to sequences of d-tuples $(k_1, k_2, \ldots, k_d)$, where each k_i is an integer in the range $[0..n-1]$?

C-4.13. Suppose we are given two sequences A and B of n elements, possibly containing duplicates, on which a total order relation is defined. Describe an efficient algorithm for determining if A and B contain the same set of elements (possibly in different orders). What is the running time of this method?

Additional Practice Questions

1. (Knuth, *The Art of Computer Programming*, volume 2)
   Suppose that you wish to obtain a decimal digit at random, not using a computer. Which of the following methods would be suitable?

   (a) Open a telephone directory to a random place (i.e., stick your finger in it somewhere) and use the units digit of the first number found on the selected page.

   (b) Same as (a), but use the units digit of the page number.

   (c) Roll a die that is in the shape of a regular icosahedron, whose twenty faces have been labeled with the digits $0, 0, 1, 1, \ldots, 9, 9$. Use the digit that appears on top, when the die comes to rest. (A felt table with a hard surface is recommended for rolling dice.)

   (d) Expose a geiger counter to a source of radioactivity for one minute (shielding yourself) and use the units digit of the resulting count. Assume that the geiger counter displays the number of counts in decimal notation, and that the count is initially zero.

   (e) Glance at your wristwatch; and if the position of the second-hand is between $6n$ and $6(n+1)$ seconds, choose the digit n.

   (f) Ask a friend to think of a random digit, and use the digit he names.

   (g) Ask an enemy to think of a random digit, and use the digit he names.

   (h) Assume that 10 horses are entered in a race and that you know nothing whatever about their qualifications. Assign to these horses the digits 0 to 9, in arbitrary fashion, and after the race use the winner's digit.