2.3-7 ★

Describe a $\Theta(n \lg n)$-time algorithm that, given a set S of n integers and another integer x, determines whether or not there exist two elements in S whose sum is exactly x.

4.1-5

Use the following ideas to develop a nonrecursive, linear-time algorithm for the maximum-subarray problem. Start at the left end of the array, and progress toward the right, keeping track of the maximum subarray seen so far. Knowing a maximum subarray of $A[1 \mathinner{..} j]$, extend the answer to find a maximum subarray ending at index $j+1$ by using the following observation: a maximum subarray of $A[1 \mathinner{..} j+1]$ is either a maximum subarray of $A[1 \mathinner{..} j]$ or a subarray $A[i \mathinner{..} j+1]$, for some $1 \le i \le j+1$. Determine a maximum subarray of the form $A[i \mathinner{..} j+1]$ in constant time based on knowing a maximum subarray ending at index j.

6.3-3

Show that there are at most $\lceil n/2^{h+1} \rceil$ nodes of height h in any n-element heap.

4-1 Recurrence examples

Give asymptotic upper and lower bounds for $T(n)$ in each of the following recurrences. Assume that $T(n)$ is constant for $n \leq 2$. Make your bounds as tight as possible, and justify your answers.

a. $T(n) = 2T(n/2) + n^4$.

b. $T(n) = T(7n/10) + n$.

c. $T(n) = 16T(n/4) + n^2$.

d. $T(n) = 7T(n/3) + n^2$.

e. $T(n) = 7T(n/2) + n^2$.

f. $T(n) = 2T(n/4) + \sqrt{n}$.

g. $T(n) = T(n-2) + n^2$.

8.1-3

Show that there is no comparison sort whose running time is linear for at least half of the $n!$ inputs of length n. What about a fraction of $1/n$ of the inputs of length n? What about a fraction $1/2^n$?

8.3-4

Show how to sort n integers in the range 0 to $n^3 - 1$ in $O(n)$ time.

8.4-2

Explain why the worst-case running time for bucket sort is $\Theta(n^2)$. What simple change to the algorithm preserves its linear average-case running time and makes its worst-case running time $O(n \lg n)$?

9.3-3

Show how quicksort can be made to run in $O(n \lg n)$ time in the worst case, assuming that all elements are distinct.