27.3-5 ★

Give a multithreaded version of RANDOMIZED-SELECT on page 216. Make your implementation as parallel as possible. Analyze your algorithm. (*Hint:* Use the partitioning algorithm from Exercise 27.3-3.)

27.3-6 ★

Show how to multithread SELECT from Section 9.3. Make your implementation as parallel as possible. Analyze your algorithm.

Problems

27-1 Implementing parallel loops using nested parallelism

Consider the following multithreaded algorithm for performing pairwise addition on n-element arrays $A[1..n]$ and $B[1..n]$, storing the sums in $C[1..n]$:

```
SUM-ARRAYS(A, B, C)
parallel for i = 1 to A.length
    C[i] = A[i] + B[i]
```

a. Rewrite the parallel loop in SUM-ARRAYS using nested parallelism (**spawn** and **sync**) in the manner of MAT-VEC-MAIN-LOOP. Analyze the parallelism of your implementation.

Consider the following alternative implementation of the parallel loop, which contains a value *grain-size* to be specified:

```
SUM-ARRAYS'(A, B, C)
n = A.length
grain-size = ?              // to be determined
r = ⌈n/grain-size⌉
for k = 0 to r - 1
    spawn ADD-SUBARRAY(A, B, C, k · grain-size + 1,
                                min((k + 1) · grain-size, n))
sync
```

```
ADD-SUBARRAY(A, B, C, i, j)
for k = i to j
    C[k] = A[k] + B[k]
```

b. Suppose that we set *grain-size* $= 1$. What is the parallelism of this implementation?

c. Give a formula for the span of SUM-ARRAYS$'$ in terms of n and *grain-size*. Derive the best value for *grain-size* to maximize parallelism.

27-2 Saving temporary space in matrix multiplication

The P-MATRIX-MULTIPLY-RECURSIVE procedure has the disadvantage that it must allocate a temporary matrix T of size $n \times n$, which can adversely affect the constants hidden by the Θ-notation. The P-MATRIX-MULTIPLY-RECURSIVE procedure does have high parallelism, however. For example, ignoring the constants in the Θ-notation, the parallelism for multiplying 1000×1000 matrices comes to approximately $1000^3/10^2 = 10^7$, since $\lg 1000 \approx 10$. Most parallel computers have far fewer than 10 million processors.

a. Describe a recursive multithreaded algorithm that eliminates the need for the temporary matrix T at the cost of increasing the span to $\Theta(n)$. (*Hint:* Compute $C = C + AB$ following the general strategy of P-MATRIX-MULTIPLY-RECURSIVE, but initialize C in parallel and insert a **sync** in a judiciously chosen location.)

b. Give and solve recurrences for the work and span of your implementation.

c. Analyze the parallelism of your implementation. Ignoring the constants in the Θ-notation, estimate the parallelism on 1000×1000 matrices. Compare with the parallelism of P-MATRIX-MULTIPLY-RECURSIVE.

27-3 Multithreaded matrix algorithms

a. Parallelize the LU-DECOMPOSITION procedure on page 821 by giving pseudocode for a multithreaded version of this algorithm. Make your implementation as parallel as possible, and analyze its work, span, and parallelism.

b. Do the same for LUP-DECOMPOSITION on page 824.

c. Do the same for LUP-SOLVE on page 817.

d. Do the same for a multithreaded algorithm based on equation (28.13) for inverting a symmetric positive-definite matrix.