Problem Set 3, CS 5800 Spring 2017

Due: Monday, 2/6, 11AM

Problem 1 (20 pts)**.** Give a sequence of compare-exchange operations which sorts any array $A[0..3]$. Explain why your sequence works. Make the sequence as short as you can.

Problem 2 (30 pts)**.** A farmer lined up n pumpkins and n watermelons arbitrarily on a shelf. He wants to rearrange the fruits so that all the pumpkins appear before the watermelons. Since the shelf has no extra space, he can only do so by flipping the order of any contiguous group of fruits. Assume that it costs time t to flip the order of any contiguous group of t fruits. Describe an algorithm for the farmer to rearrange the fruits in $O(n \log n)$ time.

For example, suppose each of the pumpkins and watermelons is represented by `P` and `W` respectively. When $n = 5$ and the initial ordering of the 10 fruits is `PWPPWPPWWW`. He can place the pumpkins before the watermelons by first flipping the 3rd through 5th fruits to get `PWWPPPPWWW`, and then flipping the 2nd through 7th fruits to get `PPPPPWWWWW`. In this way the farmer would spend time $3 + 6 = 9$ to rearrange the fruits.

Problem 3 (20 pts)**.** Let $A[1..n]$ be an array A of length n. Suppose that the comparison of any number in A to $A[1]$ takes time n, and any other comparison takes time $O(1)$. Show that randomized quick sort sorts A in expected running time $O(n \log n)$.

Problem 4 (30 pts)**.** You just received a large sum of money for your startup. You have decided to use the money to renovate your office by filling the floor with some cool L-shaped bricks. These bricks have the form of 2×2 squares with one corner missing. Your office has size $2^k \times 2^k$ squares with a pillar standing at position (r, c) in the office. The pillar has a base of size 1×1 square and you cannot put any brick at (r, c). The bricks cannot overlap, so you will use exactly $(2^k \cdot 2^k - 1)/3$ bricks.

Design a recursive algorithm that on input k and (r, c), outputs a possible filling of the floor. Your algorithm should output a 2-dimensional array $A[1..2^k][1..2^k]$ of size $2^k \times 2^k$, where for $(i, j) \neq (r, c)$, $A[i, j] \in \{1, 2, \dots, (2^k \cdot 2^k - 1)/3\}$, where each value indicates the brick number. For example, if $k = 2$ and $(r, c) = (2, 2)$ then a correct filling is

```
3322
3012
4115
4455
```

1. Give a divide-and-conquer algorithm for this problem. (Hint: divide the floor in four, place one brick, and recurse.)

2. State the recurrence of the running time of your algorithm.

3. Solve the recurrence. Explain how you obtain your solution.