

Homework 3, CS420, Fall 2011

Due 9/20 at recitation

1. This problem is based on 27-2, page 721 of the handout.
 - (a) Draw odd-even merging networks for two, four, eight, and sixteen elements.
 - (b) By copying merging networks of these sizes onto a set of 16 lines, draw a sorting network for 16 elements.
 - (c) Suppose the inputs to the odd-even network are 0's and 1's. Consider the pre-condition on the inputs to the merging network (that they consist of two sorted halves). Let i be the number of 1's on the odd lines and let j be the number of 1's on the even lines. In terms of j , give a tight lower bound and a tight upper bound on i . That is, tell by how much i can be below and above j .
 - (d) Give the proof asked for in part c. It should work by induction on k , where 2^k is the number of lines. If $k = 1$, it's trivial. Suppose $k > 1$ and the odd-even merging network works for 2^{k-1} lines. Why does this imply that an odd-even merging network works for 2^k lines?
Hint: Use your answer to the previous problem in your proof.
 - (e) Solve part d by giving recurrences and then getting closed-form big- Θ bounds.
2. 27-3 Permutation Networks (handout)
 - (a) Part a. Let π be a permutation of n elements. That is, it is a function, from $\{1, 2, \dots, n\}$ to $\{1, 2, \dots, n\}$, where $\pi(i)$ tells what position element i is moved to, and where each element of $\{1, 2, \dots, n\}$ is $\pi(i)$ for some i . (Such a function is called a *bijection*.)

A permutation can be represented by a sequence of n distinct numbers from 1 to n , where the i^{th} number tells where element in position i is moved to.

Let $\pi^{-1}(i)$ be the *inverse* of this permutation. It tells the starting position of the element that goes to position i . Because π is a bijection, this inverse exists. Convince yourself that $(\pi^{-1})^{-1} = \pi$.

Suppose you want the sorting network to implement a permutation π . What sequence of numbers could you put on the inputs to force it to do this?
 - (b) Do part b.
 - (c) For part c, the first constraint is that two elements that start off at the same switch have to go to different $P_{n/2}$'s. The second constraint is that two elements that end up at the same switch have to go to different $P_{n/2}$'s. If you satisfy these two constraints, then it is easy to see that the two $P_{n/2}$'s can get each element from its starting switch to its ending switch. A way to route them that does this tells how to set the input switches. The settings of the output switches are then trivial; you swap them if the output that goes earlier came from the second $P_{n/2}$. Therefore, the trick is to figure out how to route the elements to the two $P_{n/2}$'s without violating either constraint.

We could draw a graph where the vertices are the elements to be permuted. Between each pair of elements that starts at the same switch, put a red edge. Between each pair that ends at the same switch, put a green edge.

To answer: What's special about the connected components, and why? What is special about the arrangement of edge colors in a component? What is special about the parity of the sizes of a component?

- (d) How can you use this to label each vertex with 1 to route it to the first $P_{n/2}$ or 2 to route it to the second $P_{n/2}$, so that they don't violate the constraints?
- (e) Do Part d, by writing recurrences and getting them into closed form.
- (f) For Part e, consider a first comparator. Think about why there are exactly as many permutations where the first comparator is crossed as there are where it is not crossed. Use this to show the result.