

## Lecture #5:

### 0.0.1 Divide & Conquer Method (Continued – More Examples):

**Example 1** *Multiplication of two  $n \times n$  matrices by Strassen's Method: We are given two  $n \times n$  matrices  $A$  and  $B$  and we want  $C = A \cdot B$ . (Page 739-745 of your book)*

Recall

$$C_{i,j} = \sum_{k=1}^n a_{i,k} \cdot b_{k,j}$$

Using this, in the usual manner, we will do  $\Theta(n^3)$  multiplications and additions. For convenience, we assume that  $n$  is a power of 2. Let

$$\begin{aligned} A &= \begin{bmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{bmatrix} \\ B &= \begin{bmatrix} B_{1,1} & B_{1,2} \\ B_{2,1} & B_{2,2} \end{bmatrix} \\ C &= \begin{bmatrix} C_{1,1} & C_{1,2} \\ C_{2,1} & C_{2,2} \end{bmatrix} \end{aligned}$$

where each of these is  $\frac{n}{2} \times \frac{n}{2}$  submatrix of the corresponding matrix. If we do multiplication the usual way, we have 8 multiplications of pairs of  $\frac{n}{2} \times \frac{n}{2}$  matrices and 4 additions. Addition of two  $n \times n$  matrices is  $\Theta(n^2)$ . V. Strassen devised an algorithms that does fewer multiplications.

For this, let

$$\begin{aligned} M_1 &= [A_{2,1} + A_{2,2} - A_{1,1}] \cdot [B_{2,2} - B_{1,2} + B_{1,1}] \\ M_2 &= [A_{1,1}] \cdot [B_{1,1}] \\ M_3 &= [A_{1,2}] \cdot [B_{2,1}] \\ M_4 &= [A_{1,1} - A_{2,1}] \cdot [B_{2,2} - B_{1,2}] \\ M_5 &= [A_{2,1} + A_{2,2}] \cdot [B_{1,2} - B_{1,1}] \\ M_6 &= [A_{1,2} - A_{2,1} + A_{1,1} - A_{2,2}] \cdot [B_{2,2}] \\ M_7 &= [A_{2,2}] \cdot [B_{1,1} + B_{2,2} - B_{1,2} - B_{2,1}] \end{aligned}$$

Then

$$\begin{aligned} C_{1,1} &= M_2 + M_3 \\ C_{1,2} &= M_1 + M_2 + M_5 + M_6 \\ C_{2,1} &= M_1 + M_2 + M_4 - M_7 \\ C_{2,2} &= M_1 + M_2 + M_4 + M_5 \end{aligned}$$

Thus, we do 7 multiplication of  $\frac{n}{2} \times \frac{n}{2}$  matrices and few more additions. Recall additions are cheaper! This gives us the recursion

$$t(n) = 7t\left(\frac{n}{2}\right) + c(n^2)$$

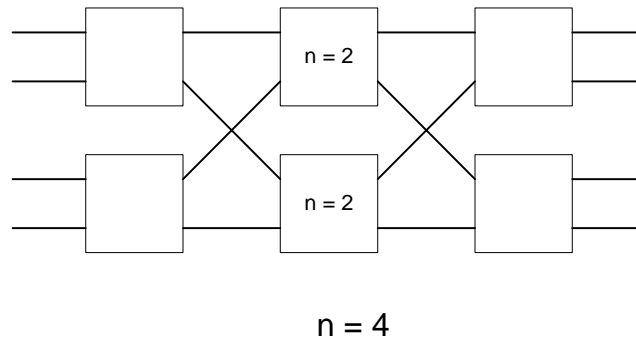
This is one of the examples we saw under the master theorem and we obtained  $t(n) = \Theta(n^{\lg 7}) = \Theta(n^{2.81})$  which is better than before. The current best is  $\Theta(n^{2.376})$ .

**Example 2** *Permutation Networks (Problem 28-3, page 652)*

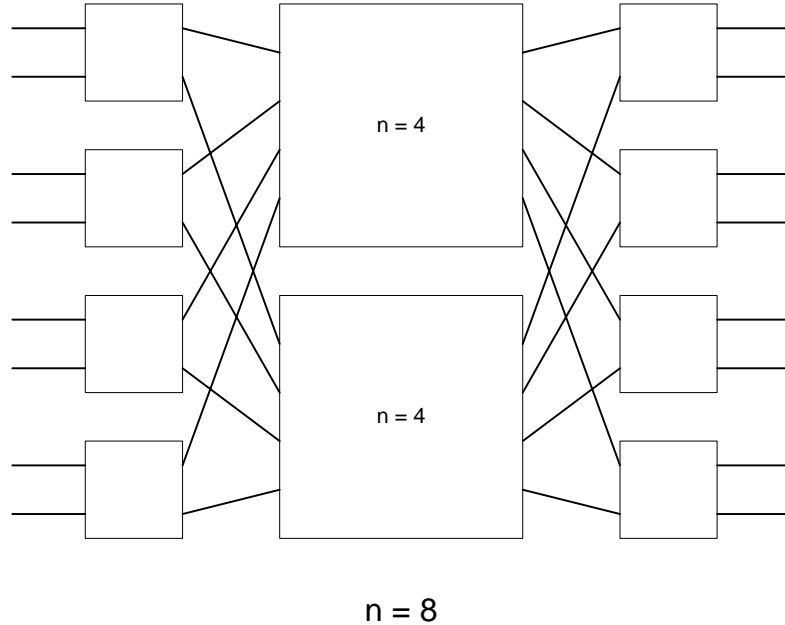
This section follows the notes from Professor Hal Sudborough. A switch is a circuit with two inputs, two outputs, and a control. The control can be set to one of two positions as shown below:



We want to construct a device with  $n$  inputs and  $n$  outputs using these switches; this device should be able to output any of the  $n!$  possible permutations of the inputs by setting the switches in an appropriate manner. Our goal is to use  $O(n \lg n)$  switches to do this. We will assume again that  $n$  is a power of 2. For example, for  $n = 4$  we have:



and for  $n = 8$



You should get the idea in general. If we let  $t(n)$  be the number of switches used for constructing a network with  $n$  inputs/outputs, we have the recurrence relation:

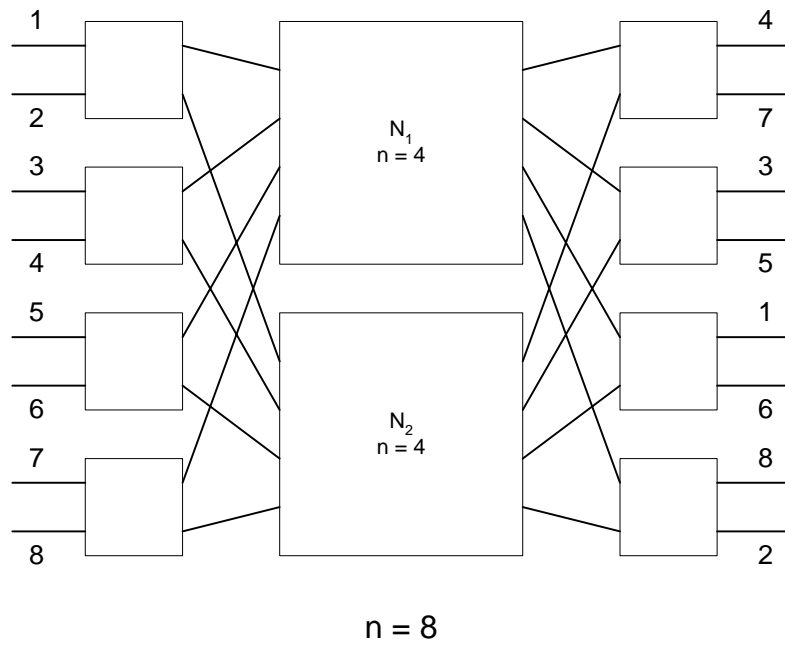
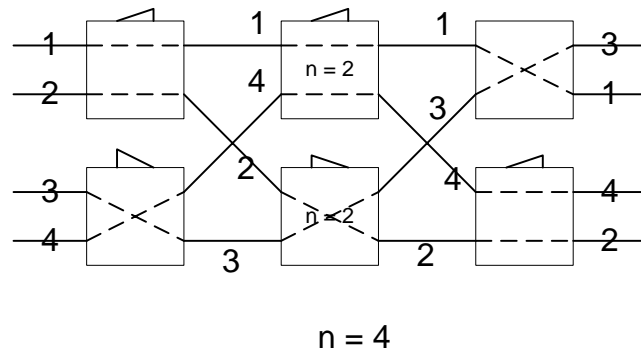
$$t(n) = 2t\left(\frac{n}{2}\right) + n$$

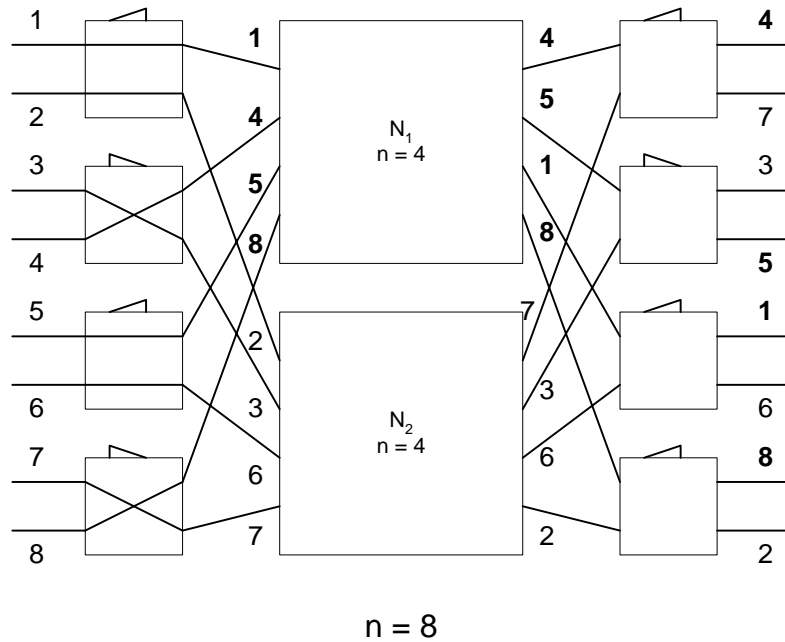
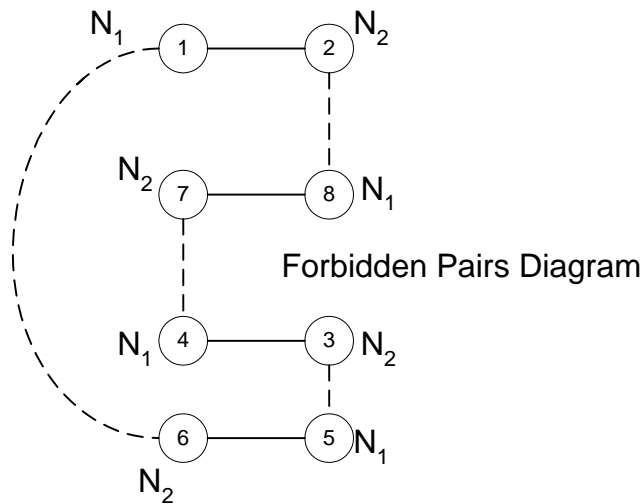
The solution of this by the master theorem is  $t(n) = \Theta(n \lg n)$ . The exact solution is  $t(n) = n \lg n - \frac{n}{2}$  and is obtained by iteration method.

We need to show one more thing: that this works correctly – we can get all the permutations of the inputs as outputs by setting the switches. For this we show how to set the switches properly. We assume that the input is  $[1, 2, \dots, n]$  in that order from the top to bottom. Let us look at some specific examples>

Let  $n = 4$ , and let the output be  $[3, 1, 4, 2]$  This creates what are called "forbidden pairs" and in this case they are  $:(3, 1), (4, 2)$  This means that the two elements in a forbidden pair should not be switched to the same "half"

network. See below for how the switches are set:





The general case is handled in a similar manner.  
 The next question: can we do with fewer switches? We show below that we can not improve the order -  $t(n) = \Theta(n \lg n)$ . Suppose we have  $m$  switches to handle  $n$  inputs. Since each switch can be in one of two positions, the system can be in no more than  $2^m$  possible configurations. Each configuration can only

yield one output permutation. Hence we must have

$$2^m \geq n!$$

Hence  $m \geq \lg[n!] = \Theta(n \lg n)$  as we showed earlier.