Homework 1: due 26 Aug 11:55pm

1. An FPGA (field programmable gate array) is a circuit on a chip, whose components can be activated/disabled so that it behaves like circuits we want to build. Simplistically, an FPGA contains processors located at points (i, j) on a square chip, for all $1 \leq i, j \leq N$ for some N. Processors differing by 1 in a single coordinate are connected by a wire. We can programmatically activate any subset of processors/wires. Further, if a certain processor is not activated, we can ask for the active wires entering it to be joined to each other however we wish. This way we can form connections between active processors that may be far apart.

   Thus to use the FPGA to implement a circuit, we should map each processor p in the circuit to some unique processor $f(p)$ in the FPGA and activate all such processors. Further if p, q are connected by a wire, then $f(p), f(q)$ should either differ in just one coordinate and be connected directly, or be connected by a long connection as described above. Only the wires in these connections are activated.

   In this exercise you are to show how to implement an L level complete binary tree network on an FPGA.

   A complete binary tree of L levels has 2^k processors in level k, for $k = 0, \ldots, L-1$. Each processor in level $i < L-1$ connects to 2 distinct processors in level $i+1$ (and hence one processor in level $i-1$ for $i > 0$). The tree thus has $2^L - 1$ processors. Assume L is odd with $L = 2M + 1$.

   Use the following implementation. Map the single tree processor in level 0 ("root") to the processor at the center of the FPGA (use an odd N). Map the two level 1 processors to conveniently chosen processors in the central row (or column). Consider the 4 quadrants obtained by ignoring the processors in the central row and the central column. Use each quadrant to implement one of the subtrees contained in levels 2 through L.

   What should N be to make this construction work? For this N, what fraction of the processors in the FPGA get used for large L? Draw a picture of the layout for $L = 5$. Please draw the N x N grid in pencil. On top of that mark in pen the processors and edges that are activated.

2. Consider the maximum subarray problem discussed in a pause point. Suppose A[1..n] is the input to the maximum subarray problem. Suppose we construct an array $B[1..n]$ by setting B[i] = A[1] + ... + A[i], for all i. Express the condition that you want a maximum subarray of A in terms of the elements of B. Show that the problem to be solved for B is equivalent to a problem you have seen.

3. Consider the problem of counting the number of ways of giving change for an amount n using denominations D[0..d-1]. As before, we represent a possible way of giving change by a nondecreasing sequence of the numbers in D. Let S denote the set of such sequences. Suppose now that we partition S into sets S_i where all elements in set S_i begin with i

occurrences of D[0], for all possible values of i. Develop a solution based on this in the style seen in the lectures. Write the recurrence. Give the details of the table you will use. Estimate the time required by the algorithm. Here it might be natural to estimate the time in terms of n,d,D[0],...,D[d-1]

The algorithm you get here will be slower than the algorithm we discussed in the lectures. However, it will be faster than most natural recursive approaches which will need time $\theta(c^n)$ for some c. The point of this exercise and the exercise in a pause point is just to alert you that there can be many strategies for partitioning the set we are trying to process.

Times of the order $\theta(c^n)$ are generally referred to as "exponential in n", just as those of order $\theta(n)$ are referred to as "linear in n".