Complexity Theory, WS 13/14: Solution Hints 3.

Markus Bläser, Thatchaphol Saranurak

Exercise 3.1 a) We can just have different conjunctions for $a_{ij} \wedge b_{jk}$ which we connect with a tree of $\vee$s. We do this for every c_{ij} but don't duplicate the input nodes. This gives us a depth of $\mathrm{O}(\log n)$ as our c_{ij} are done parallel. We also have a size of $n^2 + n^2 \cdot (2^{\log n} + n) \in \mathrm{O}(n^3)$.

b) We use a similar technique as square and multiply. Let $b_1 \dots b_m$ be the bit representation of n. We now construct a circuit of size $m \in \mathrm{O}(\log n)$ in the following way. We take m copies of our multiply circuit and connect them in order on the first input. On the second input we either connect a if $b_i = 1$ or the output of the circuit c_{i-1} if $b_i = 0$. It is easy to see that this computes the right value. The first circuit c_1 is connected to a and b and the last to our output.

Trivially our depth is $\mathrm{O}(m \cdot \log n) = \mathrm{O}(\log^2 n)$ and the size $\mathrm{O}(n^3 \cdot m) = \mathrm{O}(n^3 \log n)$.

Exercise 3.2 a) We are given a binary tree T rooted at r. We denote subtree rooted at a node u by T_u. For any node u with children v, v', we say v is a *big* child of u iff $|T_v| \geq |T_{v'}|$. Let $P = (r = u_1, \dots, u_d)$ be a path such that u_d is a leaf, and u_{i+1} is a big child of u_i.

Note that $|T_r| = n, |T_{u_d}| = 1$ and $|T_{u_{i+1}}| < |T_{u_i}| \leq 2|T_{u_{i+1}}| + 1$ because u_{i+1} is a big child of u_i. Therefore, there is k where $n/3 \leq |T_k| \leq 2n/3$. Otherwise, there exists i where $|T_{u_i}| \geq 2n/3+1$ and $|T_{u_{i+1}}| \leq n/3-1$ which contradicts $|T_{u_i}| \leq 2|T_{u_{i+1}}|+1$. Removing the edge (u_{k-1}, u_k) will do a job.

b) By a), given a boolean formula F, we can split F into F' and F_{sub} where $F' = F \backslash F_{sub}$ containing the output gate, and both have size at most $2n/3$. Let F'_0 and F'_1 be the formulas obtained by plugging 0 and 1 into F, respectively, at the gate which was the output gate of F_{sub}.

The formula $(F'_0 \wedge \neg F_{sub}) \vee (F'_1 \wedge F_{sub})$ computes the same function as F. We apply the construction recursively to F'_0, F'_1 and two copies of F_{sub}. Therefore the resulting formula has depth $D(n) = 3 + D(2n/3) = O(\log n)$ and size $S(n) = O(1) + 4S(2n/3) = n^{O(1)}$.

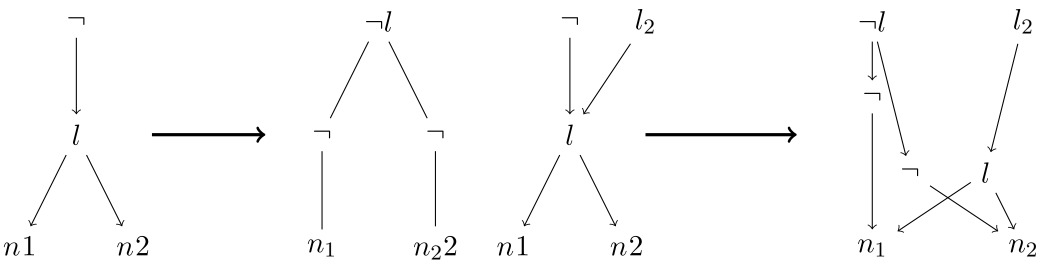
Exercise 3.3 There is a family of width-2 permutation program $(B_n)_n$ of length n deciding `PARITY` defined as follows: $B_n \triangleq (j_i, \sigma_i, \tau_i)_{i=1}^n = (i, (12), id)_{i=1}^n$, and $F_n = \{(12)\}$.

For any input $x \in \{0,1\}^n$, (12) is chosen as many times as a number of 1's in x. As $(12) \circ (12) = id$, we have $B(x) = (12) \in F_n$ iff $x \in$ `PARITY`.

Exercise 3.4 (Exercise 2.1) (a) We start at the top and whenever we have out-degree 1 with outputs $c_1, \ldots, c_p$ we make new copies for every p. This does not increase the depth of our circuit d and because all gates have at most two inputs, our size is bounded by $2^{\mathrm{O}(d)}$.

(b) For this we need to look at two cases. We want to push negations to the bottom and combine them. Assume we have our negation gate $n_\neg$, a gate with label l, n_l before this and two input gates n_1, n_2. If n_l has only one output or only outputs with negation, we can use DeMorgan rules to push the negations down. If however n_l has at least one not negated input, we copy the non-negative part, connect it with n_1, n_2 and apply DeMorgan rules on the negative part only. For an example see Figure **??**.

Figure 1: Example



(a) First case (b) Second Case

This adds at most one node for every node we touch and at most n additional nodes for negated input.