# CS 121 Homework 2: Fall 2020

**Some policies:** (See the course policies page at `http://madhu.seas.harvard.edu/courses/Fall2020/policy.html` for the full policies.)

- **Collaboration:** You may collaborate with other students that are currently enrolled in this course in brainstorming and thinking through approaches to solutions, but you should write the solutions on your own and may not share them with other students.
- **Owning your solution:** Always make sure that you "own" your solutions to this and other problem sets. That is, you should always first grapple with the problems on your own, and even if you participate in brainstorming sessions, make sure that what you write down to submit reflects your own understanding of the solution. This is in your interest as it ensures you have a solid understanding of the course material, which will help in the midterms and final. Getting 80% of the problem set questions right on your own will be much better to both your understanding and grade than getting 100% of the questions by gathering hints from others without true understanding.
- **Serious violations:** Sharing questions or solutions with anyone outside this course, including posting on outside websites, is a violation of the honor code policy. In particular, you may not get help from students or materials from past years of this or equivalent courses.
- **Submission Format:** The submitted PDF should be typed and in the same format as ours. Please include the text of the problems and write **Solution X:** before your solution. Please mark in Gradescope the pages where the solution to each question appears. We may deduct points if you submit in a different format.
- **Late Day Policy:** To give students some flexibility to manage your schedule, you are allowed a total of **eight** late days through the semester, but you may not take more than **two** late days on any single problem set.

**By writing my name here I affirm that I am aware of these policies and abided by them while working on this problem set:**

**Your name:** Todd Morrill

**Collaborators:**

**No. of late days used on previous psets (not including Homework Zero):**
**No. of late days used after including this pset:**

Please solve the following problems. Some of these might be harder than the others, so don't despair if they require more time to think or you can't do them all. Just do your best. Also, you should only attempt the bonus questions if you have the time to do so. If you don't have a proof for a certain statement, be upfront about it. You can always explain clearly what you are able to prove and the point at which you were stuck. You can always simply write **"I don't know"** and you will get 15 percent of the credit for this problem. If you are stuck on this problem set, you can use Piazza to send a private message to all staff.

**Problem 1:**

**Problem 1.1 (10 points):** Let $\mathrm{Maj}_3 : \{0,1\}^3 \to \{0,1\}$ be the function given by

$$\mathrm{Maj}_3(a,b,c) = \begin{cases} 0 & a+b+c \leq 1 \\ 1 & a+b+c \geq 2 \end{cases}.$$

Give a NAND circuit with at most six NAND gates that computes $\mathrm{Maj}_3$. (Hint: You might find it simpler to find an AND-OR-NOT circuit computing $\mathrm{Maj}_3$ and then convert it to a NAND circuit. For partial credit you can use more NAND gates—1 point off if you use one extra gate, and 2 points off for two or more extra gates.)

**Solution 1.1:**

T1 = NAND(X[0], X[1])
T2 = NAND(X[0], X[2])
T3 = NAND(X[1], X[2])
T4 = NAND(T1, T2)
T5 = NAND(T4, T4)
Y[0] = NAND(T5, T3)

**Problem 1.2 (20 points):** Let $\text{Comp}_n : \{0,1\}^{2n} \to \{0,1\}$ be the function that takes two $n$-bit integers as input and outputs 1 if and only if the first input is smaller than the second. Specifically,

$$\text{Comp}_n(A[0],\ldots,A[n-1],B[0],\ldots,B[n-1]) = \begin{cases} 1 & \sum_{i=0}^{n-1} A[i]2^i < \sum_{i=0}^{n-1} B[i]2^i \\ 0 & \sum_{i=0}^{n-1} A[i]2^i \geq \sum_{i=0}^{n-1} B[i]2^i. \end{cases}$$

Show that $\text{Comp}_n$ has $O(n)$-sized circuits.

**Hint:** Identify some intermediate variables $U[i]$, $V[i]$, $W[i]$, ... (you may use more or less) such that, for every $i$, $U[i]$, $V[i]$, and $W[i]$ can be computed by an $O(1)$-sized circuit from the inputs and $U[i-1]$, $V[i-1]$, and $W[i-1]$. Describe the $O(1)$-sized circuit completely, then show how this leads to an $O(n)$-sized circuit computing $\text{Comp}_n$.

**Solution 1.2:**

$T_0$ = AND(NOT(A[0]), B[0]) # check if $a < b$, final output for ith bits
$E_0$ = XNOR(A[0], B[0]) # check if equal

$G_1$ = AND(NOT(A[1]), B[1]) # check if $a < b$
$E_1$ = XNOR(A[1], B[1]) # check if equal
$T_1$ = AND($E_0$, $G_1$) # final output for ith bits
$F_1$ = AND($E_0$, $E_1$) # maintain a running signal that determines if we're still comparing bits

$G_2$ = AND(NOT(A[2]), B[2]) # check if $a < b$
$E_2$ = XNOR(A[2], B[2]) # check if equal
$T_2$ = AND($F_1$, $G_2$) # final output for ith bits
$F_2$ = AND($F_1$, $E_2$) # maintain a running signal that determines if we're still comparing bits
$\vdots$
$G_{n-1}$ = AND(NOT(A[n-1]), B[n-1]) # check if $a < b$
$E_{n-1}$ = XNOR(A[n-1], B[n-1]) # check if equal
$T_{n-1}$ = AND($F_{n-2}$, $G_{n-1}$) # final output for ith bits

Y[0] = OR($T_0$, OR($T_1$, OR($T_2$, OR( $\cdots$, $T_{n-1}$)))) # n-1 OR gates

As demonstrated above, $\text{Comp}_n$ requires a constant number of gates to compare each of the $n$ pairs of bits, which shows that $\text{Comp}_n$ has an $O(n)$-sized circuit.

**Problem 2 (Composition of functions):** In the following three problems, let $f, g, h : \{0,1\}^n \to \{0,1\}^n$ such that for all $x \in \{0,1\}^n$, $h(x) = g(f(x))$.

**Problem 2.1 (2 points):** Suppose $f$ can be computed by a NAND circuit with $s_1$ gates and $g$ can be computed by a NAND circuit with $s_2$ gates. Show that $h$ can be computed by a NAND circuit with $s_1 + s_2$ gates.

**Solution 2.1:** Since $h(x) = g(f(x))$ we can treat this function composition as a sequence of steps. In other words we can first compute $f$ with $s_1$ gates and then feed the output of $f$ to $g$ and compute $g$ with $s_2$ gates. This sequence of $s_1 + s_2$ gates is equivalent to computing $h$, which is what we wanted to show.

**Problem 2.2 (12 points):** Prove or disprove: If $f$ can be computed by an NAND circuit with at most $2n$ gates, and no NAND circuit of size $n^{10}$ computes $g$, then no NAND circuit of size $n^{10}/2 - 10n$ computes $h$.

**Solution 2.2:** We will disprove that no NAND circuit of size $n^{10}/2 - 10n$ computes $h$. Suppose $f$ is the function that maps all inputs to some fixed bit string in $\{0,1\}^n$ (e.g. $(Y[0] = 0, Y[1] = 0, \cdots, Y[n-1] = 0)$ for all inputs $(X[0], X[1], \cdots, X[n-1]) \in \{0,1\}^n$). This can be done with two NAND gates to map any input to 1 and five NAND gates to map any input to 0, both of which are O(1) circuits. By rearraging the input and output wires we can create any bit string in $\{0,1\}^n$. This can be done as follows:
NAND(X[0], NAND(X[0], X[0])) = 1
NAND(NAND(X[0], NAND(X[0], X[0])), NAND(X[0], NAND(X[0], X[0]))) = 0

With this function $f$, all inputs in $\{0,1\}^n$ collapse to one output in $\{0,1\}^n$ and therefore when $g$ is pre-composed with $f$, all outputs of $g$ collapse to one output. In other words, since $h(x) = g(f(x))$ we have shown that $h(x)$ can be computed with $O(1)$ NAND gates. This disproves that there is no NAND circuit of size $n^{10}/2 - 10n$ that computes $h$.

**Problem 2.3 (6 points):** Prove or disprove: If $f$ can be computed by an NAND circuit with at most $2n$ gates, and no NAND circuit of size $n^{10}$ computes $h$, then no NAND circuit of size $n^{10}/2 - 10n$ computes $g$.

**Solution 2.3:** The proof is by contradiction. Suppose there was a circuit of size $n^{10}/2 - 10n$ to compute $g$. Since $h(x) = g(f(x))$ then $n^{10} \leq |CIRC_h| = 2n + |CIRC_g| \leq n^{10}/2 - 8n$, which is a contradiction since $n^{10} \not\leq n^{10}/2 - 8n$. Therefore, there is no NAND circuit of size $n^{10}/2 - 10n$ that computes $g$.

**Problem 3:** For this question, let Blah : $\{0,1\}^3 \to \{0,1\}$ be the function $\text{Blah}(x,y,z) = x \vee (y \wedge \bar{z})$. Let $\mathbf{0}$ denote the constant function with one input and output "0". Similarly, let $\mathbf{1}$ denote the constant function with one input and output 1.

**Problem 3.1 (10 points):** Prove that $\{\text{Blah}, \mathbf{0}, \mathbf{1}\}$ is universal.

**Solution 3.1:**

We first show that we can create NOT from the provided functions. Blah$(0,1,z)$ = NOT(z). We then show that we can create AND from the provided functions. Blah$(0,y$,NOT(z)) = AND(y, z). Once we have NOT and AND, we can compute NAND, which we know to be universal.

**Problem 3.2 (10 points):** Prove that $\{\text{Blah}, \mathbf{1}\}$ is not universal. (Hint: Can you implement the $\mathbf{0}$ function using Blah and $\mathbf{1}$? It may help to look at problem 5 first, even if you don't solve it.)

**Solution 3.2:**

If all inputs are 1, we have no way of producing 0. In particular, $\mathbf{1}(x) = 1$ and $\text{Blah}(1,1,1) = 1$. Therefore, this set of functions cannot produce 0 and is therefore not universal.

**Problem 4:** Let $\text{Maj}_3$ be the function from Problem 1.1. Let $\text{XOR}(x, y)$ be the function $\text{XOR}(x, y) = (x + y) \mod 2$ (i.e., $\text{XOR}(x, y) = 1$ if exactly one of $x$ and $y$ is one, and zero otherwise.).

**Problem 4.1 (10 points):** Prove or Disprove: $\{\text{Maj}_3, \text{XOR}, \mathbf{0}\}$ is universal.

**Solution 4.1:**

$\{\text{Maj}_3, \text{XOR}, \mathbf{0}\}$ is not universal. If all inputs to the circut are 0 then we can never produce 1. $\mathbf{0}(x) = 0$, XOR(0,0) = 0, $\text{Maj}_3(0, 0, 0) = 0$.

**Problem 4.2 (10 points):** Prove or Disprove: $\{\text{Maj}_3, \text{XOR}, \mathbf{1}\}$ is universal.

**Solution 4.2:**

$\{\text{Maj}_3, \text{XOR}, \mathbf{1}\}$ is universal. We can produce NOT from $XOR(1, x) = NOT(x)$. We can then produce 0 with XOR(1, 1)=0. With this, we can produce AND from $\text{Maj}_3(0, y, z) = \text{AND(y, z)}$. Once we have NOT and AND, we can compute NAND, which we know to be universal.

**Problem 5.1 (0 points, bonus):** Prove that if $n$, $a_1$, ..., $a_n$ are nonnegative integers, $f_1 : \{0,1\}^{a_1} \to \{0,1\}$, $f_2 : \{0,1\}^{a_2} \to \{0,1\}$, ..., $f_n\{0,1\}^{a_n} \to \{0,1\}$ are functions, and any of the following conditions holds, then $\{f_1, \ldots, f_n\}$ is not universal.

1. For all $i$, $x_1$, ..., $x_{a_i}$, $1 - f_i(x_1, \ldots, x_{a_i}) = f_i(1 - x_1, \ldots, 1 - x_{a_i})$
2. For all $i$, $x_1 \geq y_1$, ..., $x_{a_i} \geq y_i$, $f_i(x_1, \ldots, x_{a_i}) \geq f_i(y_1, \ldots, y_{a_i})$
3. For all $i$, $f_i(0, \ldots, 0) = 0$.
4. For all $i$, $f_i(1, \ldots, 1) = 1$.
5. **(Hard):** The XOR function "almost" commutes with $f_i$ for all $i$, i.e., $f_i(x \oplus y) = f_i(x) \oplus f_i(y) \oplus f_i(0)$.[1] More elaborately (and using proper functional notation): For all $i$, $x_1$, ..., $x_{a_i}$, $y_1$, ..., $y_{a_i}$,

$$f_i(\mathrm{XOR}(x_1, y_1), \ldots, \mathrm{XOR}(x_{a_i}, y_{a_i})) = \mathrm{XOR}\left(\mathrm{XOR}\left(f_i(x_1, \ldots, x_{a_i}), f_i(y_1, \ldots, y_{a_i})\right), f_i(0, \ldots, 0)\right).$$

**Solution 5.1:**

**Problem 5.2 (0 points, bonus, very hard):** Prove that if none of those conditions holds, then $\{f_1, \ldots, f_n\}$ is universal. (Hint: you can get 0, 1, and NOT with negation of the first four conditions.)

**Solution 5.2:**

[1] Strict commuting would have not allowed the "$\oplus f_i(0)$" correction term at the end.