Problem 1

Pf. Let $n = |S|$ and $m = |T|$. Suppose for sake of contradiction that $n \neq m$, then we must have $n < m$ or $n > m$. Let us consider the second case.

Assume $f : S \to T$ is one to one. Let $S = \{s_1, \ldots, s_n\}, T = \{t_1, \ldots, t_m\}$. Without loss of generality (since we can reorder elements), we can say that

$$f(s_i) = t_i$$

for $1 \leq i \leq m$. Notice that there are still $n - m$ elements, namely $\{s_{m+1}, \ldots, s_n\}$ that we haven't defined a mapping for, but suppose they mapped to any $t_i \in T$, we would have $f(s_i) = f(s_j) = t_i$, where $m \leq j \leq n$, but note in particular we have $s_i \neq s_j$, which contradicts the fact that f is one to one (since definition of injective is: for all $x, y \in S$, $f(x) = f(y) \implies x = y$). Thus we must have $n \not< m$. By a symmetric argument, we also have $n \not> m$.

Thus we have shown that $n = m$.

$\square$

Problem 2

One way to represent directed graphs is with an adjacency list of sorts. First label all vertices from 1 to n. Recall that it will take $\log n$ bits to represent integers up to n, and a total of

$$2 \log n + 2$$

bits to make the encoding prefix-free. Our representation will be a list of lists, where the i-th list enumerates the neighbors of the i-th vertex. So for each vertex, it will take up to

$$10 \cdot (2 \log n + 2) = 20 \log n + 20$$

bits to represent its neighbors (since number of neighbors capped at 10). We need to again convert this list encoding into one that is prefix-free, which will take

$$2 \cdot (20 \log n + 20) + 2 = 40 \log n + 42$$

bits per list of neighbors. Now we need to concatenate n of these adjacency lists together, which would take

$$n \cdot (40 \log n + 42) = 40n \log n + 42n$$

total bits to represent. But for n sufficiently large, the first term will dominate and the encoding will take at most $40n \log n \leq 1000n \log n$ bits.

This is a valid encoding because it just uses compositions of integer encodings, conversion to prefix-free encodings, and concatenations of prefix-free encodings, which were all shown in lecture to be valid encoding functions with a respective decoding function.

$\square$

Problem 3

Pf. In class we proved that the set $\{AND, OR, NOT\}$ is universal, and that it is equivalent to the set $\{NAND\}$. Therefore, we just have to show that $\{AND, NOT\}$ can construct $NAND$ to show that it is also universal. Observe

$$NAND(a, b) = NOT(AND(a, b))$$

Thus $\{AND, NOT\}$ is universal.

$\square$

Problem 4

Pf. We will prove that C, any n-bit circuit computed only by $AND, OR, ZERO, ONE$ gates, is monotone by mathematical induction on the number of layers of C (call it m).

Base Step: We will prove the statement holds when C is a single layer ($m = 1$) of one of the following gates: $AND, OR, ZERO, ONE$.

Assume $x, x' \in \{0,1\}^2$ and that x is bitwise less than or equal to x'.

For the ZERO and ONE gates, we clearly always have

$$ZERO(x) = 0 \leq 0 = ZERO(x') \text{ and } ONE(x) = 1 \leq 1 = ONE(x')$$

For the AND gate, notice we will always have

$$C(x) \leq C(x')$$

because the only case where $C(x) > C(x')$ is if $x = 11$ and x' is something other than 11, which is not possible since it violates our assumption that x is bitwise less than or equal to x'.

For the OR gate, again observe we will always have

$$C(x) \leq C(x')$$

because the only case where $C(x) > C(x')$ is if $x \in \{01, 10, 11\}$ and $x' = 00$, which is not possible since it violates our assumption that x is bitwise less than or equal to x'.

Thus the base step is complete.

Inductive Step: Assume the inductive hypothesis holds for circuits of m layers. We want to show that it holds for circuits of $m+1$ layers. Notice that no matter which of the 4 gates the $m+1th$ layers consist of, if the inputs to it come from x or x', then the output is monotone as explained above. If the inputs a, a' to the $m+1th$ layer come from the results of a previous gate, we know that a is bitwise less than or equal to a' because of the inductive hypothesis. Thus ultimately we maintain

$$C(x) \leq C(x')$$

for circuits of $m+1$ layers, which completes the inductive step.

Thus C is monotone.

Now consider the function NAND. It is not monotone because if $x = 00, x' = 11$, we have

$$NAND(x) = 1 > 0 = NAND(x')$$

even though x is bitwise less than or equal to x'. Since every function computed by C is monotone, it certainly cannot compute the NAND function. Thus the set $\{AND, OR, 0, 1\}$ is not universal.

$\square$