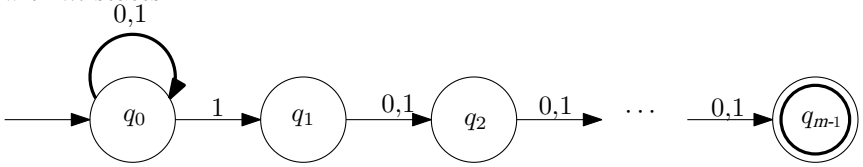
1

Strategy. Construct an NFA for a language we know the minimal DFA size of.

Outline. Consider the language of strings where the character $m-1$ before the end is a 1. We explicitly give an NFA with m states recognizing this language. However, the minimal DFA for this language has 2^{m-1} states, because fewer states cannot distinguish between all of the 2^{m-1} possibilities for the previous $m-1$ characters of the input.

Proof. Define L to be the language $\{w1x \mid w, x \in \{0,1\}^*, |x| = m-1\}$. L can be recognized by an NFA with m states:



On the other hand, any DFA M with less than 2^{m-1} states must by Pigeonhole reach the same state on two distinct strings $w = w_1w_2 \dots w_{m-1}$ and $x = x_1x_2 \dots x_{m-1}$ of length $m-1$. Let i be the largest such that $w_i \neq x_i$, say WLOG $w_i = 1$ and $x_i = 0$. We write $w' = w_1w_2 \dots w_{i-1}$, $x' = x_1x_2 \dots x_{i-1}$ and $z = w_{i+1} \dots w_{m-1} = x_1 \dots x_{m-1}$. Then $w = w'1z$ and $x = x'1z$. Now M must reach the same state on $w0^{m-1-i}$ and $x0^{m-1-i}$, but the former can be written $w'1(z0^{m-1-i})$ and the latter cannot be written in this form. Hence M cannot recognize L.

2

(a) Define L to be the language of binary strings with twice as many 0s as 1s.

Consider the CFG given by: $\{\{S\}, \{0,1\}, R, S\}$ with R containing the following rules:
$S \mapsto \epsilon|SS|1S00|00S1|0S1S0$

Every rule of this CFG that produces exactly twice as many terminal 0s as 1s. Hence all the strings produced by this CFG will be contained in L.

Now, we will prove by induction that every string w in L can be generated by this CFG. Let $O(w)$ represent number of 1s in w and $Z(w)$ represent number of zeros in w. Let $f(w) = Z(w) - 2O(w)$. Clearly $w \in L$ if and only if $f(w) = 0$.
We will induct on $|w|$.

Base Case: The empty string is derived by $S \mapsto \varepsilon$.

Inductive Hypothesis: Assume all strings with $|w| < n$ can be produced for some $n \geq 0$.

Induction Step: Suppose $|w| = n$.

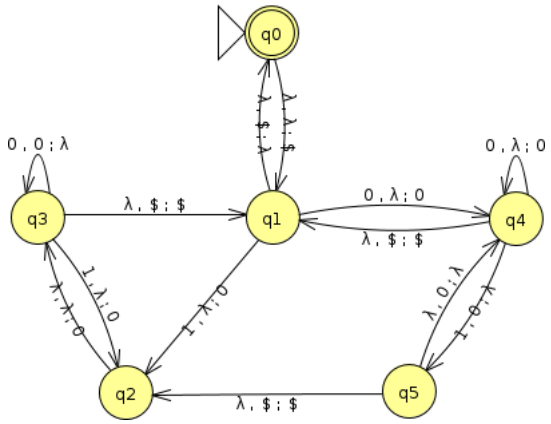
Case 1: w can be written as two nonempty strings $w = ab$ such that $f(a) = 0$. Since $f(w) = f(a) + f(b) = 0$, $f(b) = 0$ as well. By inductive hypothesis a and b are produced by S, so rule $S \mapsto SS$ gives w.

Case 2: For all possible proper nontrivial prefixes a of w, $f(a) > 0$. Then w must begin with 00. Since $f(w) = 0$, we know that $f(w_1 \dots w_{n-1})$ is negative if the last symbol is a 0, so we must have $w_n = 1$. Removing these three characters from the front and end gives a shorter string twice as many 0s and 1s, so by IH S derives this string. Hence $S \mapsto 00S1$ derives w.

Case 3: For all possible proper nontrivial suffixes b of w, we have $f(b) < 0$. As in the last case, the first character of w must be a 1, and the last two must be 0. Hence $S \mapsto 1S00$ derives w.

Case 4: There is some i with $f(w_1 \dots w_i) > 0$ and $f(w_1 \dots w_{i+1}) < 0$. Then $w_{i+1} = 0$. Write $a = w_1 \dots w_i$ and $b = w_{i+2} \dots w_n$. We know $f(w_1, \dots, w_i) = 1$ and no nontrivial prefix a' has $f(a') = 0$, so a must start with a 0. Similarly, $f(b) = f(w) - f(a) - f(1) = 0 - 1 - (-2) = 1$ and no nontrivial prefix b' has $f(b') = 0$, so b must end with a 0. Hence $f(w_2 \dots w_n) = f(w_{i+2} \dots w_{n-1}) = 0$ so these substrings are derived by S by IH, so the rule $S \mapsto 0S1S0$ derives w.

Next, we give a PDA:



This PDA accepts if and only if it reaches q_1 with an empty stack. Furthermore, note that all transitions to q_1 read a \$ from the stack, so it has an empty stack any time it is in q_1.

If q_1 sees a 1 in the input, it will push two 0s onto the stack to expect and go to q_3. From there, it will continue to pop a 0 for any 0 in the input and push two 0s for any 1 in the input. If the stack ever becomes empty, there have been twice as many 0s as 1s, so it moves back to the start state.

If q_1 sees a 0 in the input, it will push it onto the stack to be matched and go to q_4. From there it will push a 0 every time it sees a 0 and pop two 0s every time it sees a 1. Again, if the stack ever becomes empty, there have been twice as many 0s as 1s, so it moves back to the start state. Finally, if it sees a 1 and there is only one 0 to pop, it pushes a 0 to expect onto the stack, and moves to state q_3.

(b) Let the grammar of L_1 be given by $\{V_1, \Sigma, R_1, S_1\}$ and that of L_2 be given by $\{V_2, \Sigma, R_2, S_2\}$. We assume for this problem that the variables in V_1 are distinct from those of V_2 (this can be achieved via renaming).

Consider the CFG given by $\{V, \Sigma, R, S\}$ where $V = V_1 \cup V_2 \cup \{S\}$, $\Sigma = \Sigma_1 \cup \Sigma_2$ and $R = R_1 \cup R_2 \cup \{S \mapsto S_1 S S_2 | \epsilon\}$.

We see that there is only one rule in the new grammar using the start symbol. This rule either gives us ϵ, in which case we are done, or gives us $S_1 S S_2$.
S_1 will derive a string in L_1 and S_2 will derive a string from L_2. Since this is the only source of production of strings from these languages, the number of strings from each language are equal. We also see from the structure of the rule that all instances of the variable S_2 produces will follow all instances of S_1, hence maintaining the correct order.
Thus, the strings produced by this CFG are exactly those contained in the given language.

3

(a) Suppose $L = \{a^{2^n} \mid n \in \mathbb{N}\}$ is context-free, and let p be its pumping length. Consider the string $a^{2^p} \in L$. $2^p > p$, so by the pumping lemma, we can write this string as $uvxyz$, where $|vy| > 0$ and $|uvx| \leq p$ and for any i, $uv^i xy^i z \in L$. Define $d = |uyz|$, and consider $w = ux^{2d} vy^{2d} z \in L$. The length of this string is $|w| = 2d|vy| + d = d(2|vy| + 1)$. Thus $|w|$ has an odd factor greater than 2, so it is not a power of two, contradicting that $w \in L$.

(b) Suppose $L = \{w \in \{0,1\}^* \mid |w| \text{ is prime}\}$ is context-free, and let p its pumping length. Let q be a prime greater than p. Consider the string $0^q \in L$. By the pumping lemma, we can write this string as $uvxyz$, where $|vy| > 0$ and $|uvx| \leq p$ and for any i, $uv^i xy^i z \in L$. Define $d = |uyz|$, and consider $w = ux^d vy^d z \in L$. The length of this string is $|w| = d|vy| + d = d(|vy| + 1)$. This is composite, contradicting that it is in L.

4

Strategy. Proof by construction using pumping lemma.

Outline. For any sufficiently long string $wvxyz \in C$, the pumping lemma gives us wv^ixy^iz for each $i \in \mathbb{N}$. Then prefix-closure gives us every vw^i, so vw^* is an infinite regular subset of C *unless* $w = \epsilon$, in which case we take $vwxy^*$ instead.

Proof. Since C is infinite and context-free, we may fix p to be its pumping length. Since there are only finitely many strings of length less than p, C must include some string, say r, of length at least p. Then by the pumping lemma, we can write $r = wvxyz$ where $|vy| > 0$ and $wv^ixy^iz \in C$ for each $i \in \mathbb{N}$. Since $|vy| > 0$, $|v| > 0$ or $|y| > 0$. If the former, then let $p = w$ and $s = v$; otherwise, let $p = wvx$ and $s = y$. In either case, we see $ps^iz \in C$ for each i. Each ps^iz has ps^i as a prefix, so by prefix-closure each of these is in C. Thus $\{ps^i | i \in \mathbb{N}\} = ps^* \subseteq C$. Since this is given by a regular expression, it is regular, and clearly it is infinite. Therefore ps^* is an infite regular subset of C