1. The language A can be split into 2 languages that satisfies $i = j$ or $j = k$, so let they be A_1 and A_2, so we can construct a grammar for A: $S \to S_1 \mid S_2$, where S is the start variable, and S_1 and S_2 can generate A_1 and A_2.

Thus, the CFG for A_1 is:

$$S_1 \to S_1 c \mid \varepsilon \mid P$$

$$P \to a P b \mid \varepsilon$$

the CFG for A_2 is:

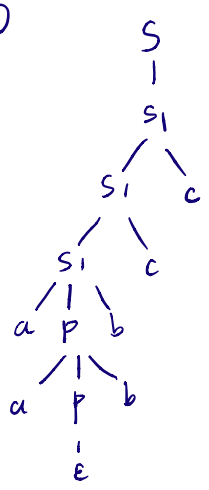
$$S_2 \to a S_2 \mid \varepsilon \mid Q$$

$$Q \to b Q c \mid \varepsilon$$

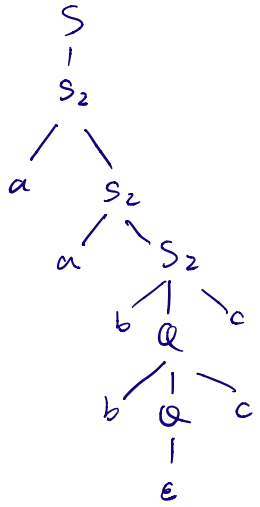
The grammar is ambiguous.

proof: given an example: $w = a^2 b^2 c^2$

①



②



both of 2 grammar analyiss tree can generate $a^2b^2c^2$ and either S_1 or S_2 can be nsed.

Hence, the CFG of language A is ambiguous.

2.

(a) language L(G) contain 2 kinds of string, let they be t and u.

t is a string contains 2 '#'s, and any numbers of 0's at any position

u is a string contains 1 '#', and the number of '0' on the right of '#' is twice than the left.

(b) Asume L(a) is regular, consider string $w = 0^p \# 0^{2p}$, $w \in L(G)$. where p is the pumping length.

Obviously, the length of w is larger than p, so by pumping lemma, $w = xyz$, which satisfies :

① $xy^iz \in L(G)$, $i \geq 0$

② $|y| > 0$

③ $|xy| \leq p$.

so, (1) if x contains '#', and then y is on the right of '#' and only contains '0', However, string 'xyyz' will not belong to L(G) because it neither satisfies the form of t nor u.
Hence, conflict with condition ① On the other hand, it is obvious the the length of $|xy|$ is larger than p, so it conflicts with ③

(2) if y contains '#', the length of xy will be at least p+1, conflict with condition ③

(3) if z contains '#', then, y is on the left of '#' and only contains '0', this case is similar with (1), conflict.

Hence, L(G) is not a regular language.

3. Design a TM T to check DFA accept all strings.

T = " On input <A>, where A is a DFA:

1. Mark the start state of A.
2. Repeat until no new states get marked:
3. Mark any state that has a transition coming into it from any state that is already marked
4. If all accept states are marked, accept; otherwise, reject.

4.

Step1. We can construct a CFG that generates all possible strings.

For example, construct $S_0 \to tV \mid \varepsilon$,

$V \to tV \mid \varepsilon$.

where t means **any** terminals in alphabet Σ and V is a variable.

Thus, from state rule S_0, we can generate any strings that belong to Σ^*. Let G_{i0} represent this CFG, so $L(G_{i0}) = \Sigma^*$, which means all languages generated by G_{i0}.

Step2.

Now we prove EQ_{CFG} is undecidable.

To complete the proof, we need to introduce "ALL_{CFG} is undecidable" to help prove; So, we prove it firstly (the proof process on ToC book is not complete)

Lemma: ALL_{CFG} is undecidable.

proof: Assume that there exists R that decides if a CFG G_0 can generate all possible string. Now we give the reduction from A_{TM}.
Construct another TM S:

S = " On input $\langle M, w \rangle$, where M is a TM, w is a string.

1. Let G_0 generate $\Sigma^* - C$, where C is all strings that are accepting computation histories for M on w.

   In other words, if M accepts w, $C \neq \emptyset$, $L(G_0) \subset \Sigma^*$

   otherwise, $C = \emptyset$, $L(G_0) = \Sigma^*$

2. Run R on G_0 (it can be implement by a push-down automata)
3. if R accepts, then reject "
4. Otherwise, accept.

Thus, We get A_{TM} is decidable with the help of ALL_{CFG} decidable.

However, we have known A_{TM} is undecidable, so, TM R is nonexistent,

so, ALL_{CFG} is undecidable.

Step 3.

Now, we use conclusions in step 1 and step 2 to prove.

proof: Assume that EQ_{CFG} is decidable, and there exists TM R that can decide it. Now we give the reduction from ALL_{CFG}

Construct another TM S, S decides ALL_{CFG}:

S = " On input $\langle G \rangle$,

1. Run R on $\langle G, G_0 \rangle$, where G_0 generates Σ^*
2. if R accepts, then accept
3. otherwise, reject

Thus, if the TM R decides EQ_{CFG}, S will decides ALL_{CFG}.

But according to the lemma in Step 2, ALL_{CFG} is undecidable, so

EQ_{CFG} is also undecidable.