1.

Example $S = a^i b^j$  
$SS = a^i b^j a^i b^j$  $i >$ number of states  
Pumping lemma gives words $a^i b^j a^i b^j$  $P$ big  

NOT SS

2.

For $m >$ number of states pumping lemma produces words $a^{m'} b^n$  $d^p b^q$  
$m > m' =$  $m' + n > p + q$

3.

The squares have an interesting property. Consecutive squares differ by consecutive odd numbers.

$1^2 = 1$  
$2^2 = 1 + 3$  
$3^2 = 1 + 3 + 5$  
$4^2 = 1 + 3 + 5 + 7$

This is because  
$(n+1)^2 - (n)^2 = 2n+1$  (an odd number.)

So the gaps between the squares grows larger and larger. For any number $M$ eventually no two squares will differ by $M$. Certainly if $x > M$ and $y > M$ then $x^2 - y^2 > M$ (unless $x= y$) since the closest they would be is $(M+1)^2 - M^2 = 2M+1 > M$. So when we pump, let $s = n^2$, $d^r = xyz = a^i a^r a^r$, the Pumping Lemma says that $xyz$, $xyyz$, $xyyyz$, ... are all in $a^*$, which are $a^{p+q}$, $a^{p+r+2q}$, $a^{p+r+3q}$, .... However, in this sequence consecutive terms differ by the constant $q$, while squares get further and further apart. Therefore these terms can not all be squares.

Alternate solution:
Assume the number of $a$’s in $y$ string is $k$ and the total number of $a$’s in $xyz$ string is $n^2$. The string $xyyz$ contains $k + n^2$ number of $a$’s. If $k + n^2 \neq (n+1)^2$, which means $xyyz$ is not in language SQUARE;  
If $k + n^2 = (n+1)^2$, which means $xyyz$ is in SQUARE, $k = 2n + 1$; obviously, the number of $a$’s of $xyyyz$ will be $2k + n^2 = 4n + 2 + n^2 \neq n^3$. So, $xyyyz$ is not in SQUARE. Therefore, it is impossible to find a division $xyz$ that guarantee $xyyz$ and $xyyyz$ will also in SQUARE. This language is therefore not regular.
4.

The procedure is:

Step 1: From the start state, find the edge that leads out of it with label $a$. If no such edge is found, stop; else, follow the edge found and paint the destination state blue.

Step 2: From every blue state, follow each edge that leads out of it and paint the destination state blue. Then delete each edge that was followed.

Step 3: Repeat step 2 until no new state is painted blue, and then stop.

Step 4: When the procedure has stopped, if any of the final states are painted blue, then the machine accepts at least one word that starts with an $a$. If not, it does not.

5.

$S \rightarrow SS|NMN$

$M \rightarrow aM|a$

$N \rightarrow aB|bA$

$A \rightarrow a|aS|bAA$

$B \rightarrow b|bS|aBB$

N corresponds to the language EQUAL

6.

A CFG is ambiguous if there is at least one word in the language that has at least two derivation trees. It is called unambiguous otherwise.

(i)$S \Rightarrow XaX \Rightarrow aXaX \Rightarrow a\Lambda aX \Rightarrow a\Lambda a = aa$
$S \Rightarrow XaX \Rightarrow \Lambda aX \Rightarrow aaX \Rightarrow aa\Lambda = aa$

(ii)$S \Rightarrow a.SX \Rightarrow aaSXX \Rightarrow aa\Lambda XX \Rightarrow aaXa \Rightarrow aaaa = aaaa$
$S \Rightarrow aSX \Rightarrow a\Lambda X \Rightarrow aaX \Rightarrow aaaaX \Rightarrow aaaa = aaaa$

(iii)$S \Rightarrow aS \Rightarrow aa.S \Rightarrow aa\Lambda = aa$
$S \Rightarrow aaS \Rightarrow aa\Lambda = aa$
(iv)

(i) This language defines the words with at least one $a$’s.

$S \rightarrow bS \mid aX$

$X \rightarrow aX \mid bX \mid \Lambda$

(ii) This language defines the words with at least two $a$’s or empty.

$S \rightarrow aX \mid \Lambda$

$X \rightarrow aX \mid a$

(iii) This language defines all words with $a$’s, $b$’s, empty, or both.

$S \rightarrow aS \mid bS \mid \Lambda$

(v)

(i) $S \rightarrow bS \mid aX \mid a$

$X \rightarrow aX \mid bX \mid a \mid b$

(ii) $S \rightarrow aX$

$X \rightarrow aX \mid a$

(iii) $S \rightarrow aS \mid bS \mid a \mid b$