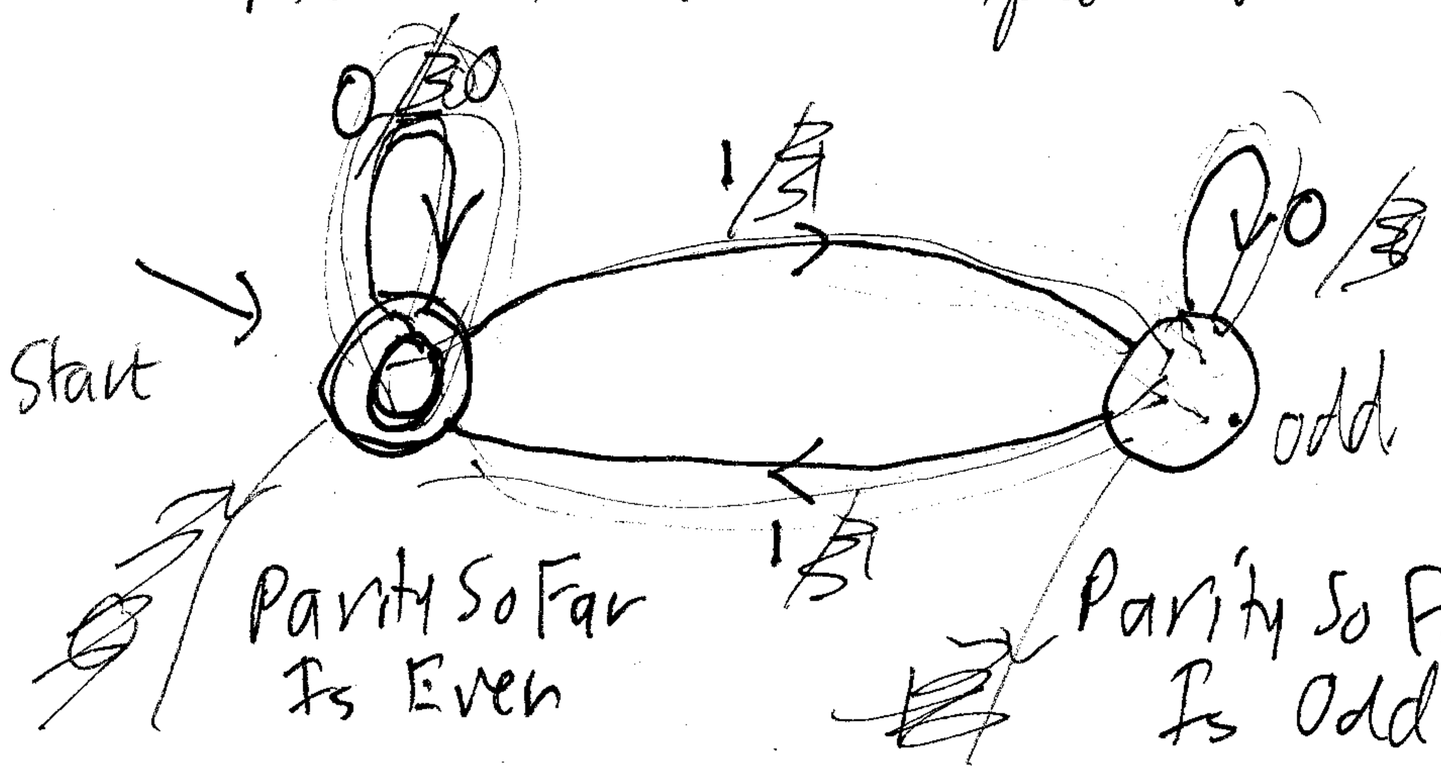
Notes for Feb 3: ① → ⑥

Finite Automata: Simplest Nontrivial Example.



At end, the machine reads an "silent" stop character, blank or UNIX $ say.

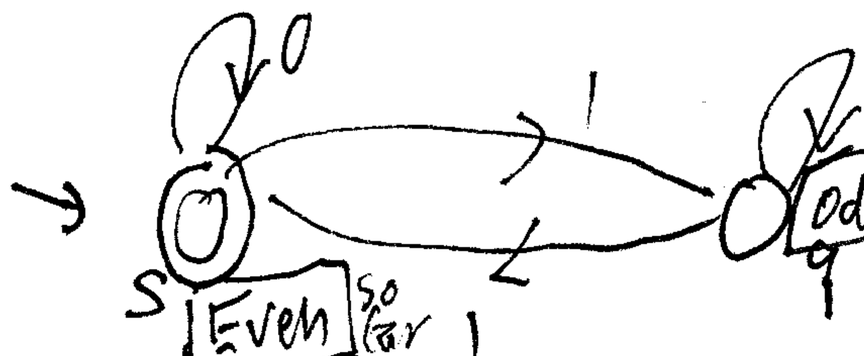
Keep Focus on <u>checking</u> the integrity of the 9-bit Parity checked byte.

<u>Start</u> is both the initial state and the desired <u>Final</u> state.

A <u>Final</u> state is also called an <u>accepting</u> state, others are <u>rejecting</u>

9-bit byte: 011010100 $\in L$ 011110100 $\notin L$

$L = \{$ 8-bit strings having even parity. (DFA)



A (deterministic) finite automaton is a 5-tuple $M = (Q, \Sigma, \delta, s, F)$ where:

Q is a finite set of states

Σ is a finite <u>input</u> alphabet

s, a member of Q, is the <u>start state</u>

F, a <u>subset</u> of Q, is the set of accepting states

δ, the transition function, is a function from $Q \times \Sigma$ to Q. $\delta: Q \times \Sigma \to Q$

$Q = \{s, q\}$ $\Sigma = \{0, 1\}$

s = literally s. $F = \{s\}$

$\delta(s,0) = s$ $\delta(q,0) = q$

$\delta(s,1) = q$ $\delta(q,1) = s$

<u>Altview:</u>

$\delta = \{(s,0/s), (q,0/q), (s,1/q), (q,1/s)\}$

"A DFA is a 5-tuple $M = (Q, \Sigma, \delta, s, F)$ where ---"

!!!

class DFA {

Set<State> Q;

set<char> Sigma;

State s;

set<State> F;

State delta(State p, char c);

Store pointer to function.

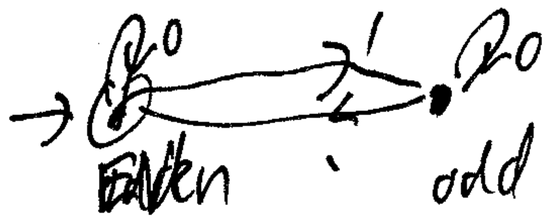
C#: "delegate" State (*delta)(State, char);

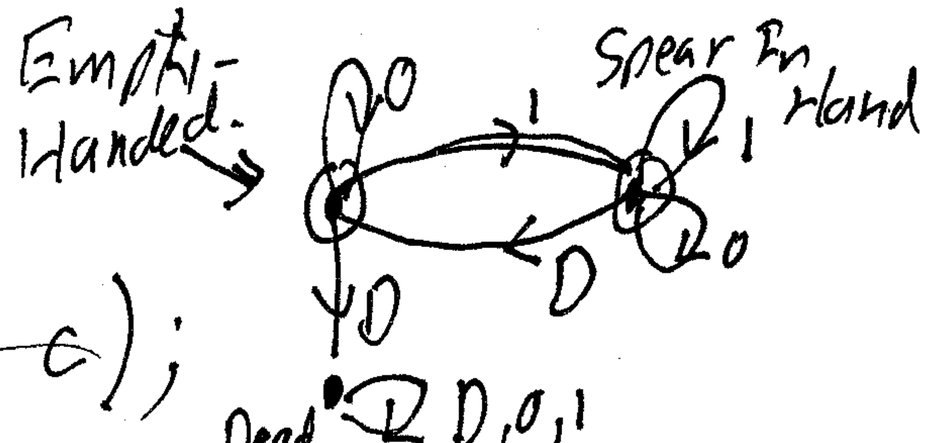
Set<triple<State, char, State>> delta;

{ State execDelta(State p, char c) ... return state

Constructor DFA(State, Set<State>, Set<char>, Set<Instruction>, State, Set<State>)

State is a finite enum or set type.





$\Sigma = \{0, 1, D\}$

dungeon X = 0 1 0 D 1 0 1 D D 0 1

{empty, spear, dead} {0, 1, D}

{empty} {empty, spear}

delta = { (empty, 0, empty) : (empty, 1, spear) : (empty, D, dead)
(spear, 0, spear) (spear, 1, spear) : (spear, D, empty)
(dead, 0, dead) (dead, 1, dead) : (dead, D, dead) }

Q \ Σ	0	1	D
→ empty	empty	spear	dead
← spear	spear	spear	empty
dead	dead	dead	dead