Graph-Theoretic Analysis of Finite Markov Chains

120170162 Siyeong Lee
Dept. of Electronic Engineering
Sogang University

Outline

- How to represent an MC using a graph?
  - Aperiodic, Recurrent or Transient
- Problem 1: State Classification
  - Algorithm: based on Depth-First Search Algorithm
- Problem 2: Periodicity
  - Algorithm
    - Breadth-First Search Algorithm
    - Calculate gcd

Introduction

- Specifications of Markov Models

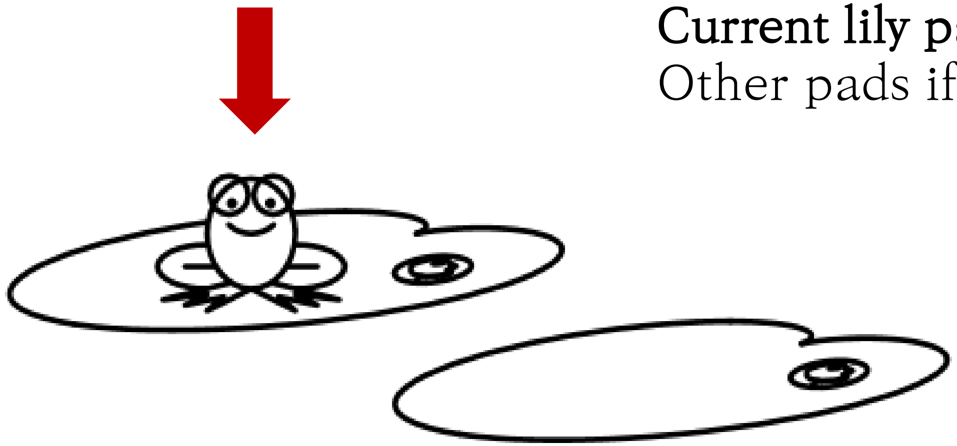
- A Markov chain model is specified by identifying
  (a) the set of states $\boldsymbol{S} = \{1, \dots, m\}$,
  (b) the set of possible transitions, namely, those pairs (i, j) for which $p_{ij} > 0$
  (c) the numerical values of those p_{ij} that are positive.
- The Markov chain specified by this model is a sequence of random variables $X_0, X_1, X_2, \dots,$ that take values in $\boldsymbol{S}$ and which satisfy

$$P(X_{n+1} = j \mid X_n = i, X_{n-1} = i_{n-1}, \dots, X_0 = i_0) = P(X_{n+1} = j \mid X_n = i),$$

for all times n, all states $i, j \in \boldsymbol{S}$, and all possible sequences $i_0, \dots, i_{n-1}$ earlier states.

How to represent an MC using a graph?

- Motivation: Markov Models look like *"frog on the lily pad"*



Current lily pad , → **Current Node**
Other pads if Possible → Adjacent Node

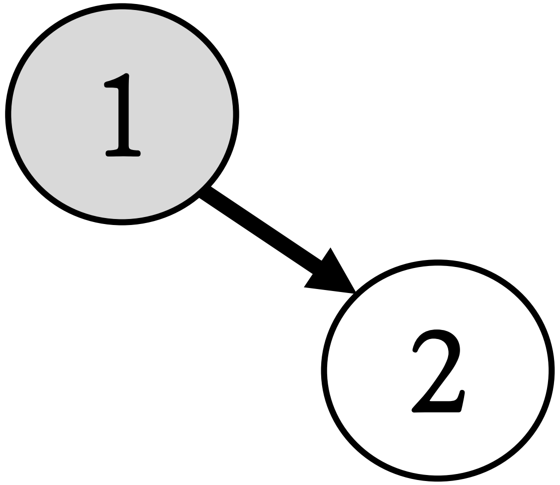
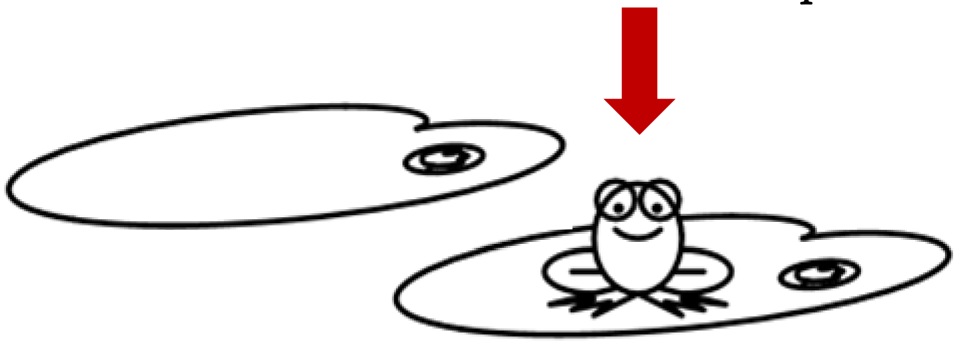


Fig 1. A randomly jumping frog.
Whenever he tosses heads, he jumps to the other lily pad. [1],page 3

How to represent an MC using a graph?

- Motivation: Markov Models look like *"frog on the lily pad"*

Current lily pad → Current Node
Other pads if possible → **Adjacent Node**



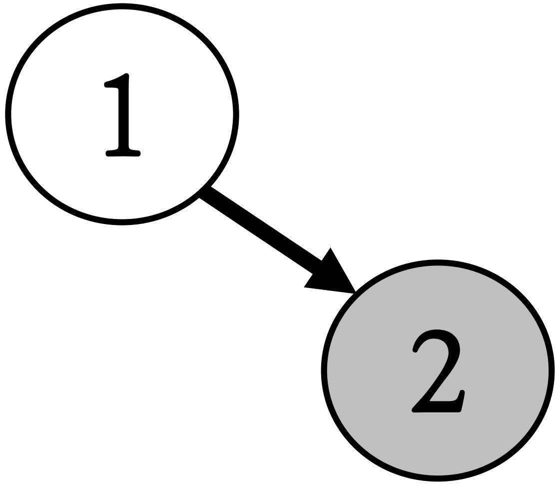
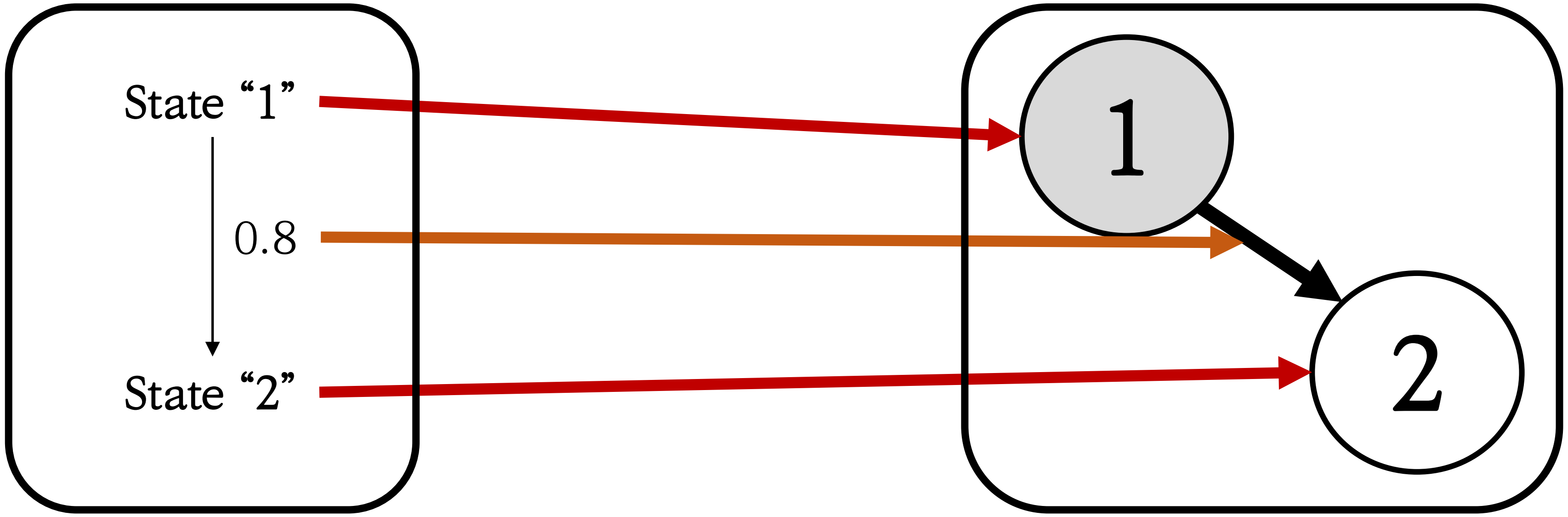


Fig 1. A randomly jumping frog.
Whenever he tosses heads, he jumps to the other lily pad. [1],page 3

How to represent an MC using a graph!

- Generate digraph $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$ which has two property
  1) Each node corresponds to a state of $\boldsymbol{M}$
     - e.g) $\boldsymbol{f}: \boldsymbol{S} \to \boldsymbol{V}$ be a bijection which maps "state i" onto "vertex i"
  2) $\boldsymbol{G}$ contains edge $\boldsymbol{e} = (i, j) \in \boldsymbol{E}$ if and only if $\boldsymbol{p_{ij}} > 0$

e.g)

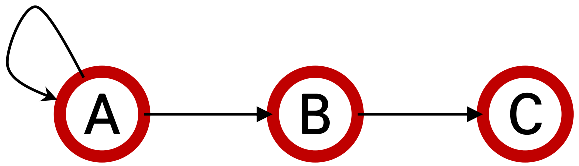


Recurrent or Transient?

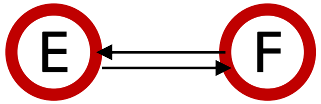
- Concepts of recurrent and transient
  - Transient
    - **vertex** i is **transient**

      iff **there exists some vertex** j **for which** $i \rightarrow j$ **but** $j \nrightarrow i$

  - Recurrent
    - **vertex** i is **recurrent** iff i is **not transient**,
    - In other words, for all $j \in \boldsymbol{V}$, i and j communicate. i.e $i \leftrightarrow j$



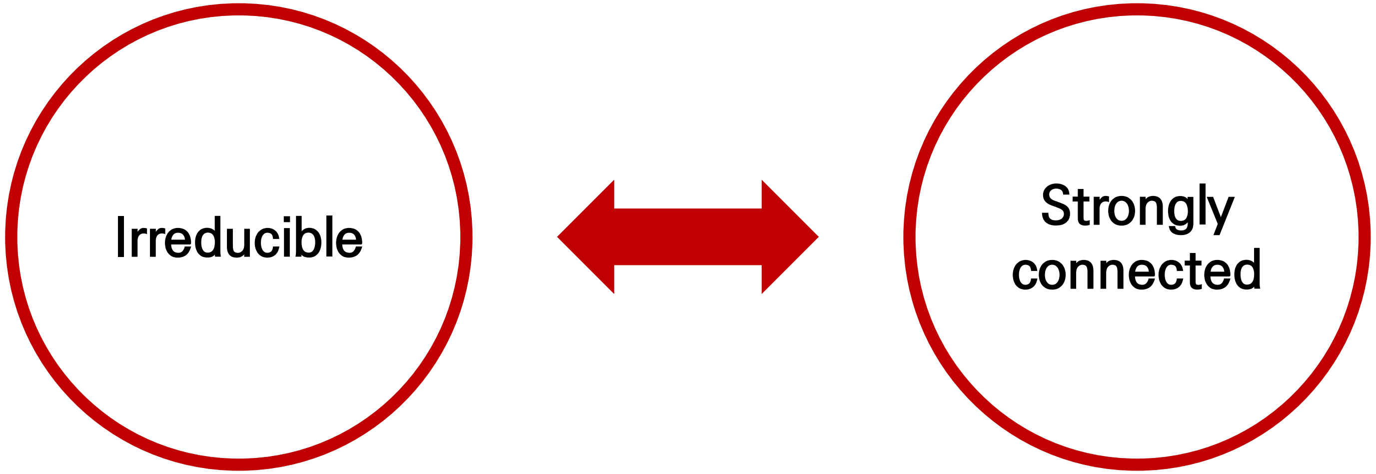
State A is transient;
Because of "C"



State E is recurrent

Irreducible or Strongly connected ?

- Additional
  - The Markov chain $\boldsymbol{M}$ is called irreducible if, for every pair of states i and j, there exist $r, s \geq 0$ with $\boldsymbol{p}_{ij}^{r} > 0$ and $\boldsymbol{p}_{ji}^{s} > 0$.
  - So, Since there is a path between every pair of nodes i, j, Graph $\boldsymbol{G}$ in which Markov chain $\boldsymbol{M}$ is expressed is strongly connected.



Markov Chain　　Graph

Problem 1: State Classification

- Goal
  - To find out which node is a recurrent.
- ***Main Idea***
  - Suppose that $\boldsymbol{M}$ is a Markov chain, $\boldsymbol{G_M} = (\boldsymbol{V}, \boldsymbol{E})$ be associated with the Markov chain $\boldsymbol{M}$, and $i, j \in \boldsymbol{V}$
  - **Lemma 1**: $The\ relation \leftrightarrow\ is\ an\ equivalence\ relation$:
    $(a)\ i \leftrightarrow i\ for\ all\ i \in \boldsymbol{V}$
    $(b)\ if\ i \leftrightarrow j, then\ j \leftrightarrow i\ for\ all\ i, j \in \boldsymbol{V}$
    $(c)\ if\ i \leftrightarrow j\ and\ j \leftrightarrow k\ , then\ i \leftrightarrow k\ for\ all\ i, j, k \in \boldsymbol{V}$.
  - So, we can reduce "State Classification of $\boldsymbol{M}$"
    to "Classification of equivalence classes" with respect to "$\leftrightarrow$".

Problem 1: State Classification

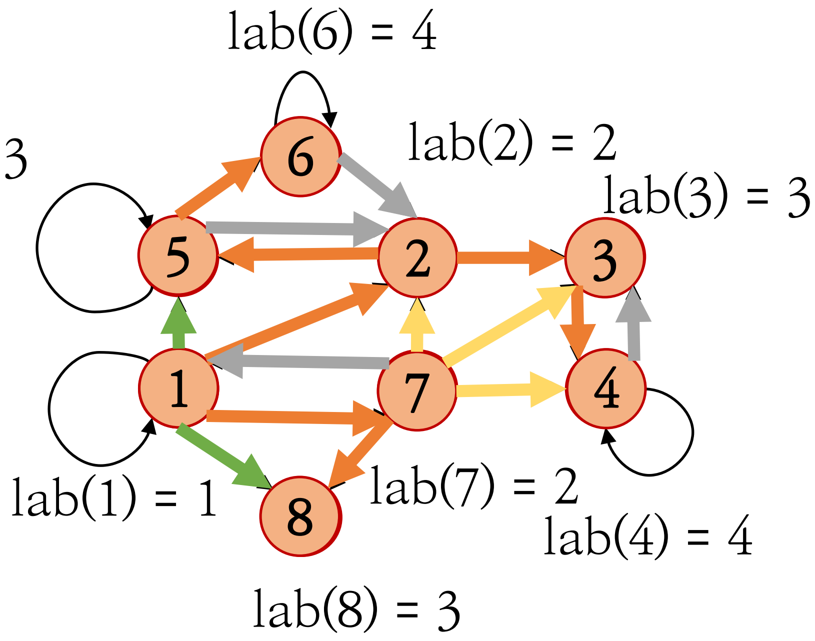
- **Problem Setting**
  - Let $\boldsymbol{M}$ be a finite-state Markov chain
    * The set of State $\boldsymbol{S} = \{1,2,3, \ldots, m\}$
    * The transitions probability matrix $\boldsymbol{M} = [p_{ij}] \in \boldsymbol{R}^{\boldsymbol{m} \times \boldsymbol{m}}$,
      $where\ p_{ij}\ is\ the\ transition\ probability\ from\ "state\ i"\ to\ "state\ j"$
      $and\ for\ evert\ p_{ij} > 0$
  - We can generate the directed graph $\boldsymbol{G} = \boldsymbol{G}_{\boldsymbol{M}}$
    having the set of nodes $\boldsymbol{V} = \{1,2,3, \ldots, m\}$ and the set of edges $\boldsymbol{E}$
    * Each node corresponds to a state of $\boldsymbol{M}$
    * $\boldsymbol{G}$ contains edge $\boldsymbol{e} = (i, j) \in \boldsymbol{E}$ if $P_{ij} > 0$ except $(\boldsymbol{i}, \boldsymbol{i})$

Real task: Should find the roots of each strong component!!

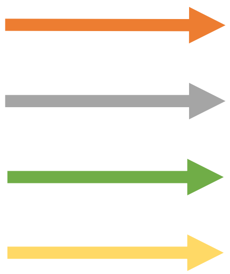
Problem 1: State Classification

- **Pseudocode**

```
  lab(v) = 1                  //label v as discovered
procedure DFS(G, v):

  for all edges from v to w in adj(v) do
     if vertex w is not yet labeled as discovered then
        lab(w) = lab(v) + 1;
        edge (v, w) is classified as a tree edge
        recursively call DFS(G, w)
     if vertex w has already been labeled then
        if lab(v) > lab(w) and v is a descendant of w in the tree then
           edge (v, w) is classified as a back edge
        if lab(v) < lab(w) and w is a descendant of v in the tree then
           edge (v, w) is classified as a forward edge
        if lab(v) > lab(w) and w is neither a descendant of v nor a ancestor of v then
           edge (v, w) is classified as a cross edge
```



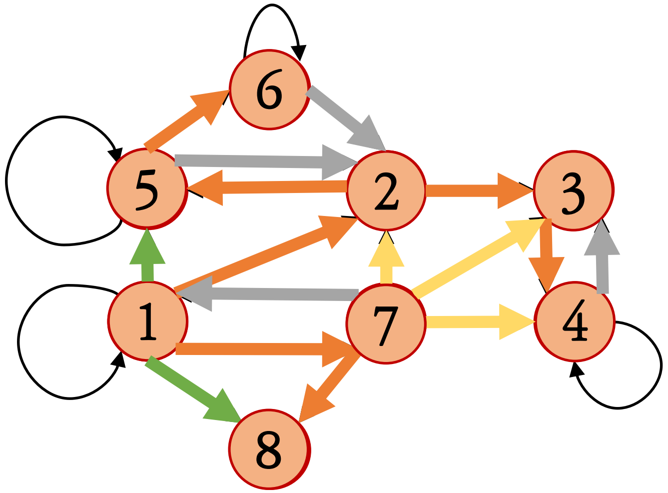
input: G is graph, v is a start node



Problem 1: State Classification

- Toy Problem ($\boldsymbol{R} = \emptyset$)

input: *Graph* $\boldsymbol{G}$
output: *Condensed Graph* $\widehat{\boldsymbol{G}}$



Def additional (G)
  list S = {back edges set}

  While S is not empty **do**
    $\boldsymbol{G_i} = \emptyset$
    For $e = (u, v)$ in S, find the vertex v
        which has the biggest value of lab(v) and is visited earlier

    v is a root of $\boldsymbol{G_i}$
    Eliminate e in S and Eliminate $\boldsymbol{G_i}$ in $\boldsymbol{G}$

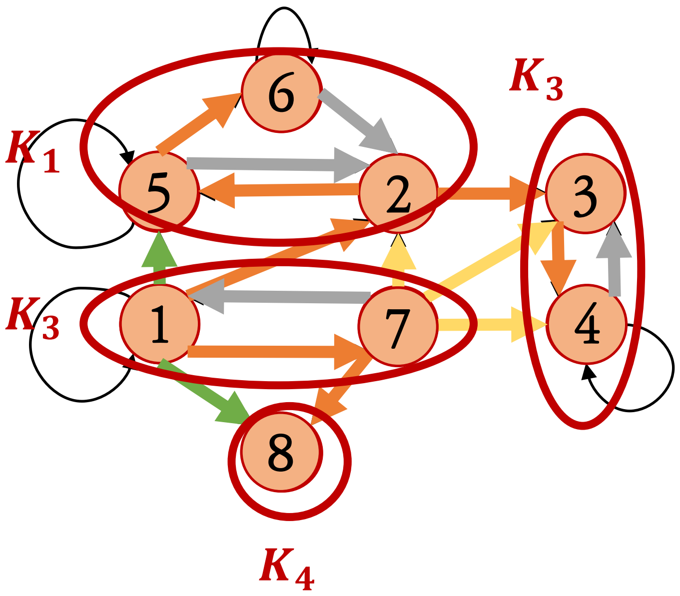
<u>$\boldsymbol{G_i}$: *Strongly component*. $\boldsymbol{R}$: the set of Roots</u>

Problem 1: State Classification

tree edge
back edge
forward edge
cross edge

input: Graph G
output: Condensed Graph $\hat{\boldsymbol{G}}$

- Toy Problem ($\boldsymbol{R} = \emptyset$)



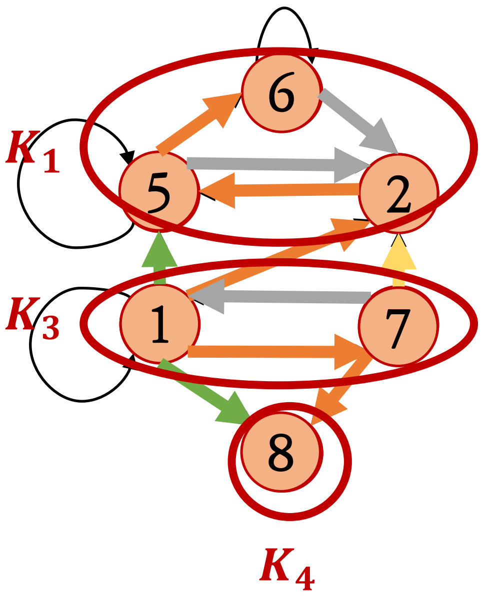

```
Def additional (G)
   list S = {back edges set}

   While S is not empty do
      G_i = ∅
      For e = (u, v) in S, find the vertex v
                which has the biggest value of lab(v) and is visited earlier

      v is a root of G_i
      Eliminate e in S and Eliminate G_i in G
```

Problem 1: State Classification

tree edge
back edge
forward edge
cross edge

- Toy Problem ($\boldsymbol{R} = \{3\}$)

input: Graph G
output: Condensed Graph $\hat{\boldsymbol{G}}$



Def additional (G)
  list S = {back edges set}

  While S is not empty **do**
    $\boldsymbol{G_i} = \emptyset$
    For $e = (u, v)$ in S, find the vertex v
        which has the biggest value of lab(v) and is visited earlier

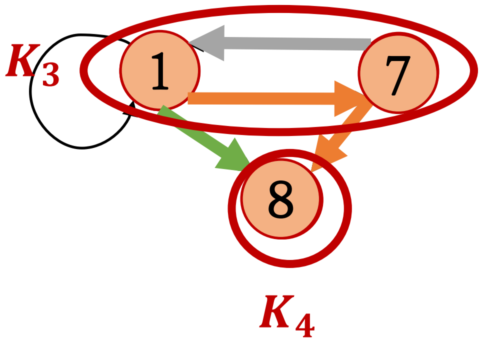
    v is a root of $\boldsymbol{G_i}$
    Eliminate e in S and Eliminate $\boldsymbol{G_i}$ in $\boldsymbol{G}$

tree edge
back edge
forward edge
cross edge

Problem 1: State Classification

- Toy Problem ($\boldsymbol{R} = \{3, 2\}$)

input: Graph G
output: Condensed Graph $\hat{\boldsymbol{G}}$



Def additional (G)
  list S = {back edges set}

  While S is not empty do
    $\boldsymbol{G_i} = \emptyset$
    For $e = (u, v)$ in S, find the vertex v
        which has the biggest value of lab(v) and is visited earlier

    v is a root of $\boldsymbol{G_i}$
    Eliminate e in S and Eliminate $\boldsymbol{G_i}$ in $\boldsymbol{G}$

tree edge
back edge
forward edge
cross edge

Problem 1: State Classification

- Toy Problem ($\boldsymbol{R} = \{3,2,1\}$)

input: Graph G
output: Condensed Graph $\hat{\boldsymbol{G}}$

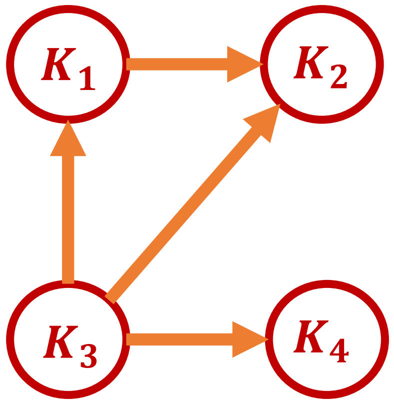

K_4

```
Def additional (G)
   list S = {back edges set}

   While S is not empty do
       G_i = ∅
       For e = (u, v) in S, find the vertex v
                     which has the biggest value of lab(v) and is visited earlier

       v is a root of G_i
       Eliminate e in S and Eliminate G_i in G
```

Then, We can identify the roots r_i of G_i

Problem 1: State Classification

- Toy Problem ($\boldsymbol{R} = \{3,2,1,8\}$)



Def additional (G)
    list S = {back edges set}

    While S is not empty do
        $\boldsymbol{G_i} = \emptyset$
        For $e = (u, v)$ in S, find the vertex v
                which has the biggest value of lab(v) and is visited earlier

        v is a root of $\boldsymbol{G_i}$
        Eliminate $\boldsymbol{e}$ in S and Eliminate $\boldsymbol{G_i}$ in $\boldsymbol{G}$

Additional, if there exist edges between $\boldsymbol{G_i}$ and $\boldsymbol{G_j}$,
Then plus an edge in condensed graph $\widehat{\boldsymbol{G}}$

Problem 1: State Classification

- **Convergence behavior to an Optimal solution**
  - By lemma 3, we can say that for every vertax $v \in \boldsymbol{K_1}, \boldsymbol{K_3}$, v is recurrent and for every vertax $w \in \boldsymbol{K_2}, \boldsymbol{K_4}$, w is recurrent

  - Time Complexity: $\boldsymbol{O}(|\boldsymbol{V}| + |\boldsymbol{E}|)$
    * Therefore, we are able to solve this problem using *DFS* in a polynomial time.

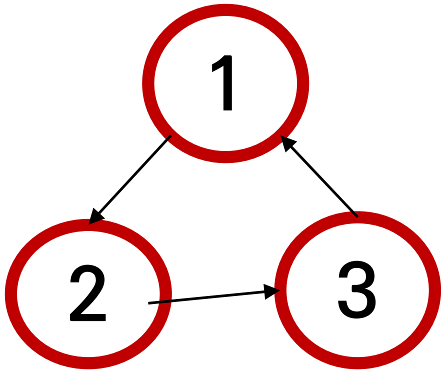
Lemma 3: The recurrent nodes of graph $\boldsymbol{G}$ are precisely those nodes whose corresponding supernodes have no leaving edges in $\widehat{\boldsymbol{G}}$.

Why we need to represent an MC using a graph?

- So we can induce that
  - $Theorem\ 1{:}\ If\ \boldsymbol{G}\ is\ strongly\ connected$
    $then\ there\ is\ a\ unique\ stationary\ distribution\ \boldsymbol{\pi}\ for\ \boldsymbol{M}.$
    * Proof: Since $\boldsymbol{G}$ is strongly connected, Markov chain $\boldsymbol{M}$ is irreducible. By steady-state convergence theorem in chap.6, we induce that Markov chain $\boldsymbol{M}$ has a unique stationary distribution $\boldsymbol{\pi}$. And so for $\boldsymbol{\pi_j}$, $\boldsymbol{\pi_j} > 0$.
  - $Theorem\ 2{:}\ If\ the\ condensed\ graph\ \hat{G}\ for\ G\ has\ a\ single\ supernode$
    $with\ no\ leaving\ edges$
    $then\ there\ is\ a\ unique\ stationary\ distribution\ \boldsymbol{\pi}\ for\ \boldsymbol{M}.$
    $Moreover, \boldsymbol{\pi_j} > 0\ for\ all\ j \in R, and\ \boldsymbol{\pi_j} = 0\ for\ all\ j \in T.$
    * Proof: Since $\hat{\boldsymbol{G}}$ has the supernode with no leaving edges, $\boldsymbol{G}$ is strongly connected and Markov chain $\boldsymbol{M}$ is irreducible and a single recurrent class.

Problem 2: Periodicity

- **Goal**
  - To find out the "limiting behavior" of a Markov chain
  - In other words, to find out the "Periodicity" of a Markov chain
    - * **Definition**
      - The period of state i is the largest integer d such that $p_{ii}^{k} = 0$ whenever k is not a positive integer multiple of d (that is, $k \neq d, 2d, 3d, \ldots$).



For any $k \neq 3, 6, 9, \ldots$,
If we start in state 1 , we can **not go back** to state 1

→ **period 3**

Problem 2: Periodicity

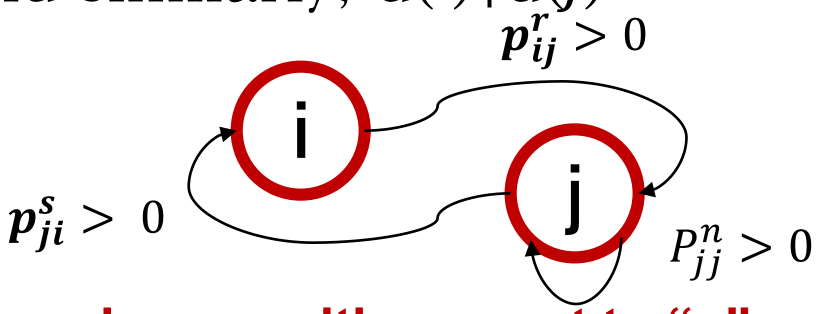
- ***Lemma* 5**: *if x has a period d and $x \leftrightarrow y$, y has also a period d*
  - Let $d(i)$ and $d(j)$ be period of i and j.
  - Suppose that $i \neq j$. so there exists $r, s \geq 0$ with $\boldsymbol{p_{ij}^r} > 0$ and $\boldsymbol{p_{ji}^s} > 0$. (i.e i $\rightarrow j$, $i \leftarrow j$)

(a) $\boldsymbol{P_{ii}^{r+s}} \geq \boldsymbol{p_{ij}^r} * \boldsymbol{p_{ji}^s} > 0$ and hence $r + s = 0 \; mod \; d(i)$

(b) Suppose that n is a positive integer with $P_{jj}^n > 0$.

Hence, $P_{xx}^{j+k+n} > 0$ and hence $j + k + n = 0 \; mod \; d(i)$

Since (a) and (b), we induce that d(i)|n. and then by the definition of period, d(j)|d(i) and similarly, d(i)|d(j)



And ~ is an equivalence relation

Therefore, we can get **a period of *M* by clustering equivalence classes with respect to "~"**

Problem 2: Periodicity

- Algorithm Design: How to calculate ?

$\boldsymbol{Theorem\ 4}$: $\boldsymbol{M}$ has period d
iff its digraph $\boldsymbol{G}$ can be partitioned into d sets $C_0, C_1, \ldots, C_{d-1}$ such that

(a) if $i \in C_k$ and $(i, j) \in E$ then $j \in C_{(k+1)\ mod\ d}$

(b) d is the largest integer with this property.

For every pair of nodes i and j in $\boldsymbol{G}$,
all (i, j) − paths in $\boldsymbol{G}$ have the same length modulo d.

Problem 2: Periodicity

- Algorithm Design

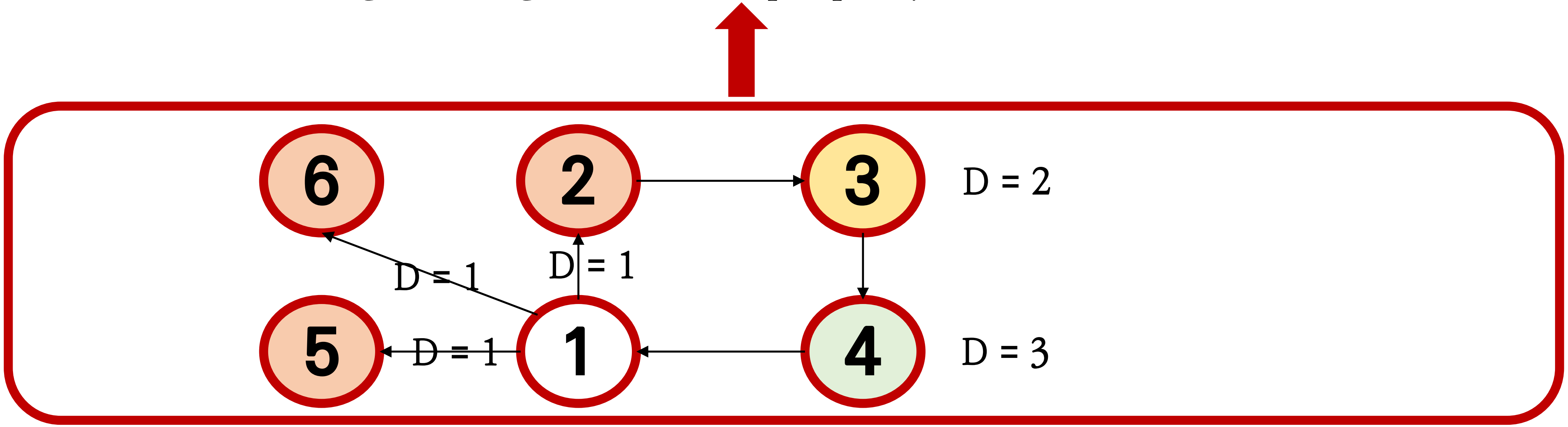
***Theorem* 4**: $\boldsymbol{M}$ has period d
iff its digraph $\boldsymbol{G}$ can be partitioned into d sets $C_0, C_1, \dots, C_{d-1}$ such that
(a) if $i \in C_k$ and $(i, j) \in E$ then $j \in C_{(k+1) \bmod d}$
(b) d is the largest integer with this property.

For every pair of nodes i and j in $\boldsymbol{G}$,
all **CYCLE** including $\boldsymbol{i}$ and $\boldsymbol{j}$ have the same length modulo d.

Problem 2: Periodicity

- Algorithm Design

***Theorem* 4**: $\boldsymbol{M}$ has period d
iff its digraph $\boldsymbol{G}$ can be partitioned into d sets $C_0, C_1, \ldots, C_{d-1}$ such that
(a) if $i \in C_k$ and $(i,j) \in E$ then $j \in C_{(k+1)\ mod\ d}$
(b) d is the largest integer with this property.



Problem 2: Periodicity

- Algorithm Design

***Theorem* 4**: $\boldsymbol{M}$ has period d
iff its digraph $\boldsymbol{G}$ can be partitioned into d sets $C_0, C_1, \dots, C_{d-1}$ such that
(a) if $i \in C_k$ and $(i, j) \in E$ then $j \in C_{(k+1) \bmod d}$
(b) d is the largest integer with this property.

Let Q_1 and Q_2 be (i, j_1) and (i, j_2) – paths in $\boldsymbol{G}$, respectively.
If $l(Q_1) \bmod d = l(Q_2) \bmod d$, then j1 and j2 are elements of the same equivalence class

Problem 2: Periodicity

- Algorithm Design

***Theorem* 4**: $\boldsymbol{M}$ has period d
iff its digraph $\boldsymbol{G}$ can be partitioned into d sets $C_0, C_1, \dots, C_{d-1}$ such that

(a) if $i \in C_k$ and $(i, j) \in E$ then $j \in C_{k+1 \bmod d}$

(b) $\boldsymbol{d}$ is the largest integer with this property.

Let Q_1 and Q_2 be (i, j_1) and (i, j_2) – paths in $\boldsymbol{G}$, respectively.
If $l(Q_1) \bmod d = l(Q_2) \bmod d$, then j1 and j2 are elements of the same equivalence class

Problem 2: Periodicity

- **Algorithm Design**
  - By collecting nodes having the length of path $x \bmod d$, the set of distinct equivalence classes forms a partition of $\boldsymbol{V}$.
  - In the irreducible Markov chain $\boldsymbol{M}$, every transition moves from a node in C_k to a node in $C_{k+1 \bmod d}$ by Theorem 4.
  - Using this property, we can find the sets of nodes C_k in $\boldsymbol{G}$ by examining the nodes of $\boldsymbol{G}$ in order of non-decreasing path length from an arbitrary starting node.
  - So, use *Breadth First Search!!*

Problem 2: Periodicity

- **Problem Setting**
  - Suppose that $\boldsymbol{M}$ is finite-state Markov chain and it is irreducible
    * The set of State $\boldsymbol{S} = \{1,2,3, \dots, m\}$
    * The transitions probability matrix $\boldsymbol{M} = [p_{ij}] \in \boldsymbol{R}^{\boldsymbol{m} \times \boldsymbol{m}}$,
      $where\ p_{ij}\ is\ the\ transition\ probability\ from\ "state\ i"\ to\ "state\ j“$
      $and\ for\ evert\ p_{ij} > 0$
  - We can generate the directed graph $\boldsymbol{G} = \boldsymbol{G}_{\boldsymbol{M}}$
    having the set of nodes $\boldsymbol{N} = \{1,2,3, \dots, m\}$ and the set of edges $\boldsymbol{E}$
    * Each node corresponds to a state of $\boldsymbol{M}$
    * $\boldsymbol{G}$ contains edge $\boldsymbol{e} = (i, j) \in \boldsymbol{E}$ if $\boldsymbol{p}_{\boldsymbol{ij}} > 0$

Real task: Find the period of strong component

- Pseudocode

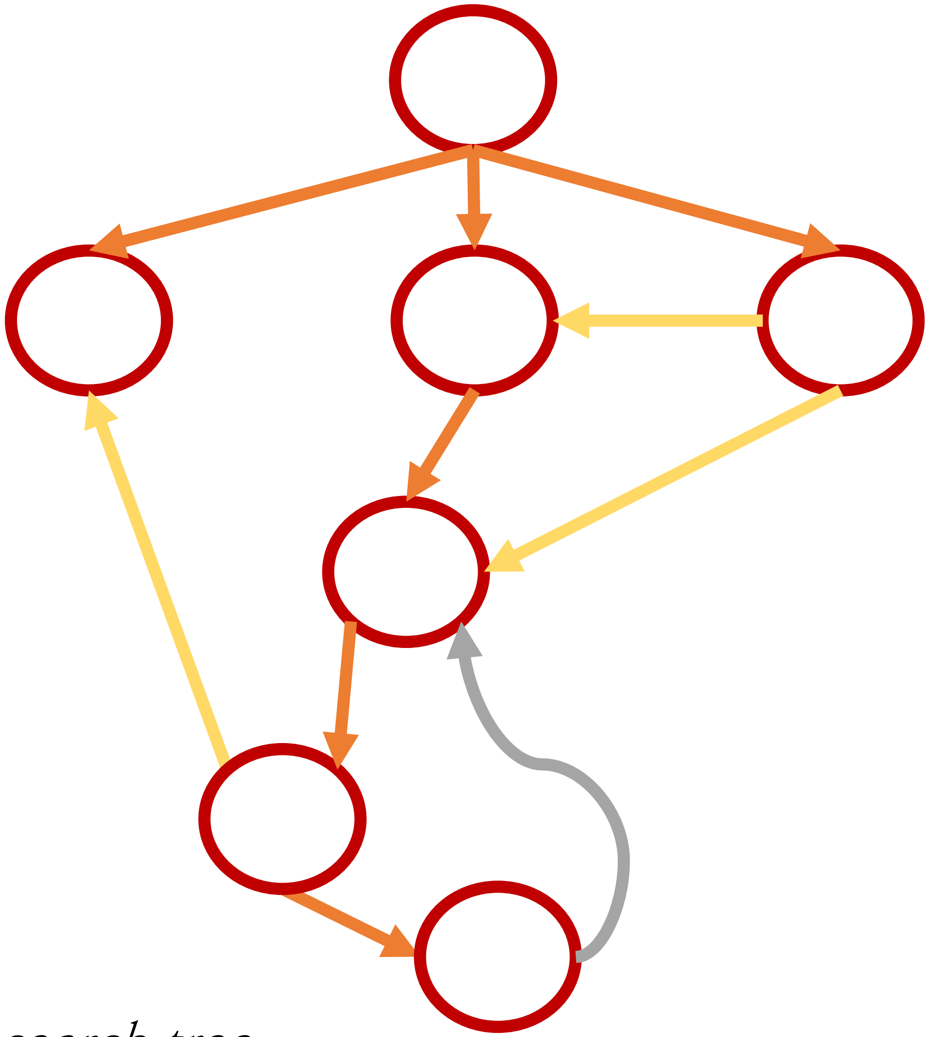
```
procedure BFS (G, v):
    level(v) = 0
    create a queue Q
    enqueue source on to Q
    mark v
    while Q is not empty:
        v ← dequeue an item from Q
        r = level(v)
        for each edge e incident on v in G do
            let w be the other end of e
            if w is not yet labeled as discovered then
                level(w) = r+1          # mark w
                add e in tree edge set
                enqueue w on to Q
            else
                s = level(w)
                If s < r then          # j is an ancestor of i in the search tree
                    add e in back edge
                else # (s ≤ r + 1)
                    add e in cross edge
```

Algorithm 1: Finding the period $\boldsymbol{d}$ of $\boldsymbol{M}$

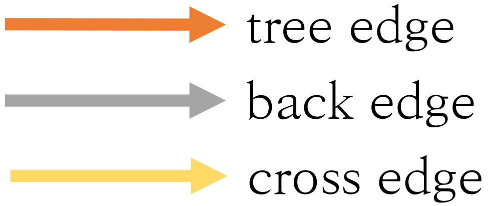
1. From an arbitrary root node, perform a breadth-first search of $\boldsymbol{G}$ producing the rooted tree $\boldsymbol{T}$.
2. The period $\boldsymbol{g}$ is given by $\gcd\{val(e) \mid e \notin T\}$

BFS algorithm
***input*:** $\boldsymbol{G}$ *is graph,* $\boldsymbol{v}$ *is a start node*
***Output*:** $\boldsymbol{T}$ *is a Spanning Tree*

- Toy Problem



```
procedure BFS (G, v):
    level(v) = 0
    create a queue Q
    enqueue source on to Q
    mark v
    while Q is not empty:
        v ← dequeue an item from Q
        r = level(v)
        for each edge e incident on v in G do
            let w be the other end of e
            if w is not yet labeled as discovered then
                level(w) = r+1          # mark w
                add e in tree edge set
                enqueue w on to Q
            else
                s = level(w)
                If s < r then           # j is an ancestor of i in the search tree
                    add e in back edge
                else # (s ≤ r + 1)
                    add e in cross edge
```



Problem 2: Periodicity

- Toy Problem

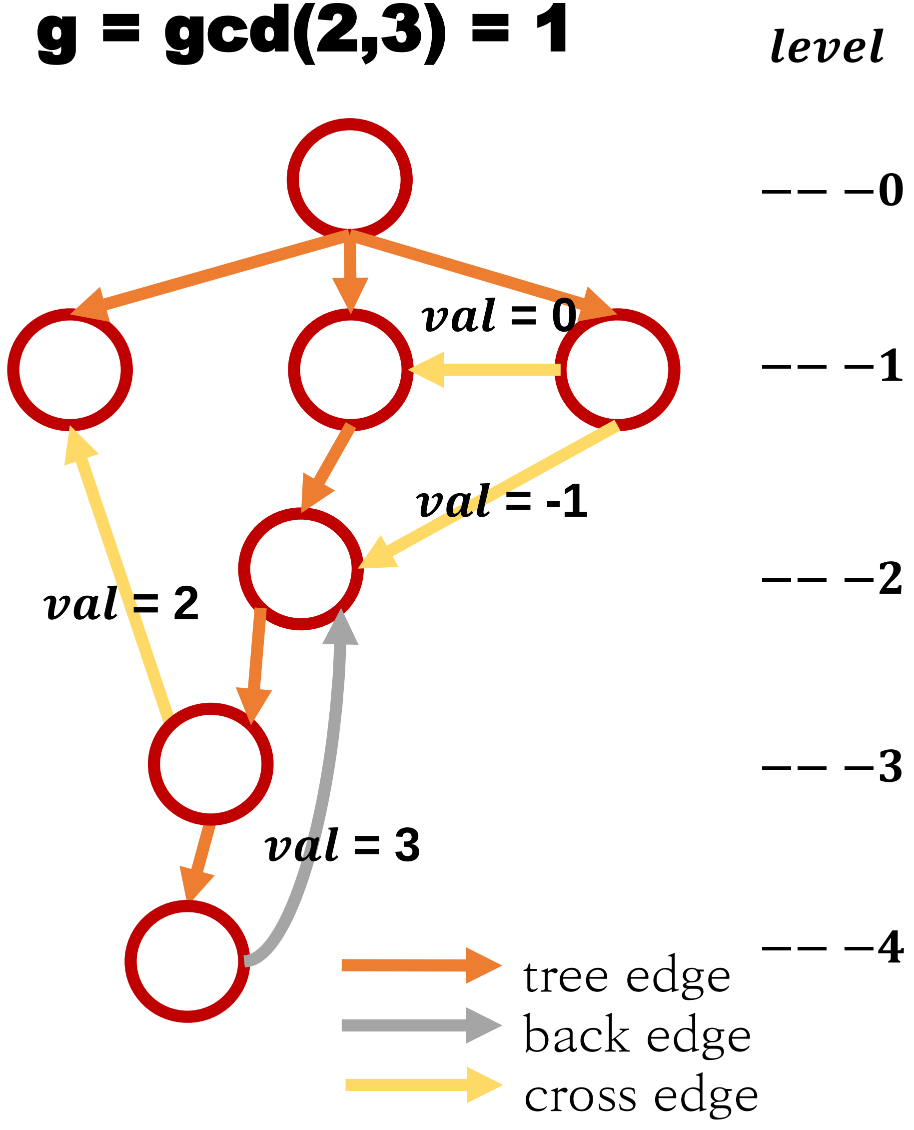
Algorithm 1: Finding the period $\boldsymbol{d}$ of $\boldsymbol{M}$

1. From an arbitrary root node, perform a breadth-first search of $\boldsymbol{G}$ producing the rooted tree $\boldsymbol{T}$.
2. The period $\boldsymbol{g}$ is given by $\mathbf{gcd}\{\boldsymbol{val}(\boldsymbol{e}) > \mathbf{0} \mid \boldsymbol{e} \notin \boldsymbol{T}\}$

Second line!!
– ***Input*** : $\boldsymbol{T}$ *is Spanning Tree*
– ***Output***: *a period* $\boldsymbol{g}$ *of* $\boldsymbol{M}$

Let $\boldsymbol{e} = (\boldsymbol{u}, \boldsymbol{v})$.
Define $\boldsymbol{val}(e)$ **by** $\boldsymbol{val}(e) = \boldsymbol{val}(u, v)$
$= \boldsymbol{level}(u) - \boldsymbol{level}(v) + \mathbf{1}$
Therefore, if $e \notin T$, $\boldsymbol{val}(e) > \mathbf{0}$, else $\boldsymbol{val}(e) = \mathbf{0}$

g = gcd(2,3) = 1



Problem 2: Periodicity

- **Convergence behavior to an Optimal solution**
  - To establish the correctness of Algorithm 1, it is sufficient to show that $g = gcd\{val(e) > 0 | \, e \notin T\}$ divides the length of an arbitrary cycle W in G.
    - 1) $\boldsymbol{g|d}$: because d is period and g is the gcd of all cycle.
    - 2) since we have already seen $\boldsymbol{d|g}$, then we must has g = d
  - Algorithm 1: two parts
    - executing the breadth-first search → Time Complexity: $\boldsymbol{O}(|\boldsymbol{E}|)$
    - finding the greatest common divisor → Time Complexity: $\boldsymbol{O}(|\boldsymbol{E}|)$

Reference

- [1] Levin, David Asher, Yuval Peres, and Elizabeth Lee Wilmer. Markov chains and mixing times. American Mathematical Soc., 2009.
- [2] J. P. Jarvis, D. R. Shier. Graph-Theoretic Analysis of Finite Markov Chains.