Computer Science and Information Engineering
National Chi Nan University

Combinatorial Optimization

Dr. Justie Su-Tzu Juan

Lecture 3. Domination Problem in Tree

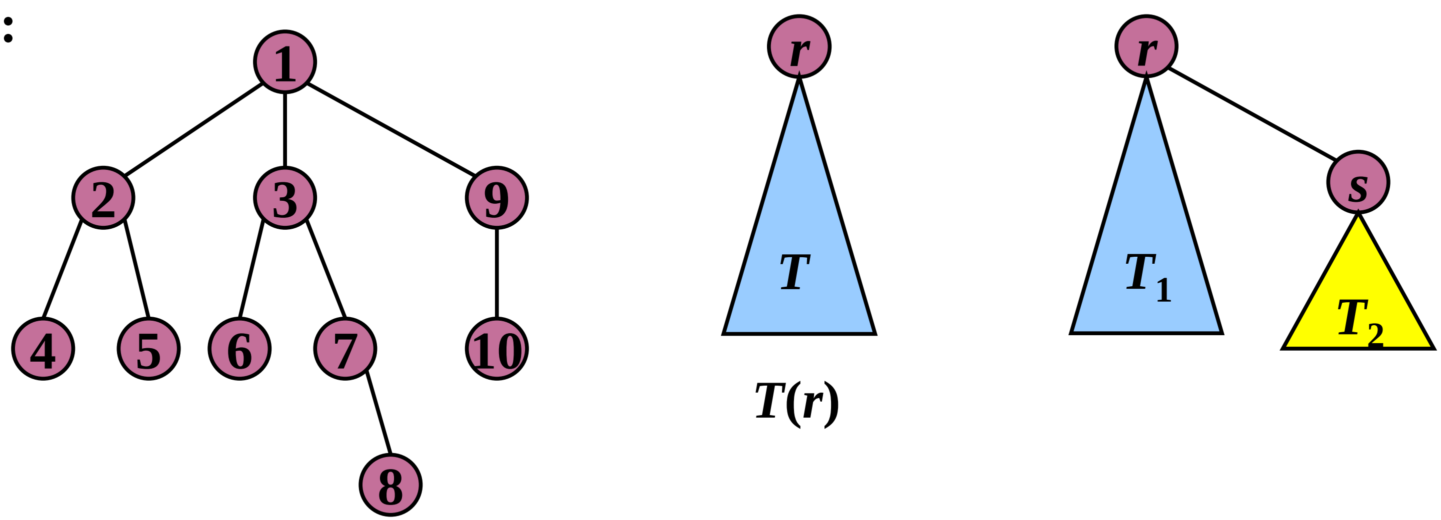
§3.3 Method 3 : Dynamic Programming

3.3 Method 3 : Dynamic Programming

- **Def:**
  - **A** **rooted tree** ***T*** **rooted at** ***r*** **is denoted by** ***T*(*r*).**
  - **Given two rooted trees $T_1(r)$ and $T_2(s)$, compose them into a rooted tree $T(r)$ by adding an edge rs into $T_1 \cup T_2$, denoted by**

$$T(r) = T_1(r) \otimes T_2(s).$$

- **Ex:**

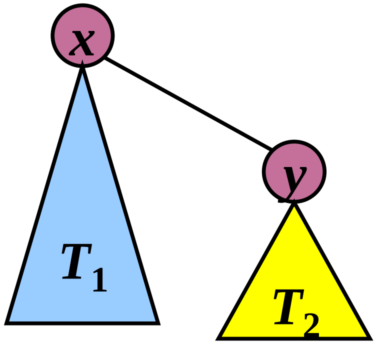


3.3 Method 3 : Dynamic Programming

- **Def**: Given a rooted tree $T(x)$,
  - ① $\gamma_1(T, x) = \min\{|D|: x \in D \text{ is a dominating set of } T\}$
  - ② $\gamma_2(T, x) = \min\{|D|: x \notin D \text{ is a dominating set of } T\}$
  - ③ $\gamma_3(T, x) = \min\{|D|: D \text{ is a dominating set of } T-x\}$

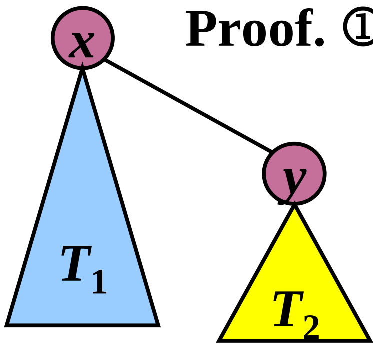
- **Note**: $\gamma(T) = \min\{\gamma_1(T, x), \gamma_2(T, x)\}$

- **Thm**: For any rooted tree $T(x) = T_1(x) \otimes T_2(y)$:
  - ① $\gamma_1(T, x) = \gamma_1(T_1, x) + \min\{\gamma_1(T_2, y), \gamma_3(T_2, y)\}$;
  - ② $\gamma_2(T, x) = \min\{\gamma_3(T_1, x) + \gamma_1(T_2, y), \gamma_2(T_1, x) + \gamma_2(T_2, y)\}$;
  - ③ $\gamma_3(T, x) = \gamma_3(T_1, x) + \gamma(T_2, y)$.



3.3 Method 3 : Dynamic Programming

- **Thm:** ① $\gamma_1(T, x) = \gamma_1(T_1, x) + \min\{\gamma_1(T_2, y), \gamma_3(T_2, y)\}$.
  ② $\gamma_2(T, x) = \min\{\gamma_3(T_1, x) + \gamma_1(T_2, y), \gamma_2(T_1, x) + \gamma_2(T_2, y)\}$.



Proof. ①

D is a dominating set of T with $x \in D \Leftrightarrow D = D_1 \cup D_2$
where D_1 is a dominating set of T_1 with $x \in D_1$,
D_2 is either a dominating set of T_2 with $y \in D_2$
or a dominating set of $T_2 - y$.
Hence $\gamma_1(T, x) = \gamma_1(T_1, x) + \min\{\gamma_1(T_2, y), \gamma_3(T_2, y)\}$.

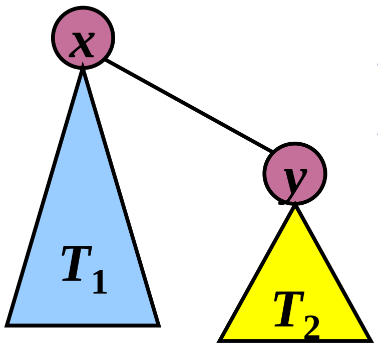
Proof. ②

D is a dominating set of T with $x \notin D \Leftrightarrow D = D_1 \cup D_2$ where
either $y \in D_2$ is dominating set of T_2
and D_1 is a dominating set of $T_1 - x$,
or $y \notin D_2$ is dominating set of T_2
and D_1 is a dominating set of T_1 with $x \notin D_1$.
$\therefore \gamma_2(T, x) = \min\{\gamma_3(T_1, x) + \gamma_1(T_2, y), \gamma_2(T_1, x) + \gamma_2(T_2, y)\}$.

3.3 Method 3 : Dynamic Programming

- **Thm:** ③ $\gamma_3(T, x) = \gamma_3(T_1, x) + \gamma(T_2, y)$.

Proof. ③



D is a dominating set of $T-x \Leftrightarrow D = D_1 \dot\cup D_2$ such that

D_1 is a dominating set of T_1-x, D_2 is a dominating set of T_2

$\therefore\ \gamma_3(T, x) = \gamma_3(T_1, x) + \gamma(T_2, y)$.

3.3 Method 3 : Dynamic Programming

- **Algorithm 3.2:**

Given tree ordering $[v_1, v_2, \ldots, v_n]$ of T

for $i = 1$ to n do

$\gamma_1(v_i) = 1$

$\gamma_2(v_i) = \infty$

$\gamma_3(v_i) = 0$

for $i = 1$ to $n-1$ do

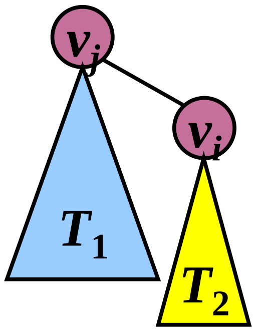
choose $j > i$ which $v_iv_j \in E$;

$\gamma_1(v_j) = \gamma_1(v_j) + \min\{\gamma_1(v_i), \gamma_3(v_i)\}$

$\gamma_2(v_j) = \min\{\gamma_3(v_j) + \gamma_1(v_i), \gamma_2(v_j) + \gamma_2(v_i)\}$

$\gamma_3(v_j) = \gamma_3(v_j) + \min\{\gamma_1(v_i), \gamma_2(v_i)\}$

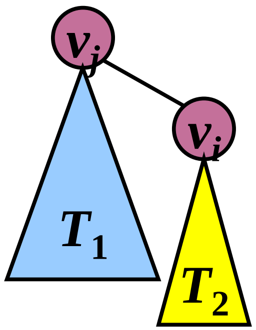
Output $\min\{\gamma_1(v_n), \gamma_2(v_n)\}$

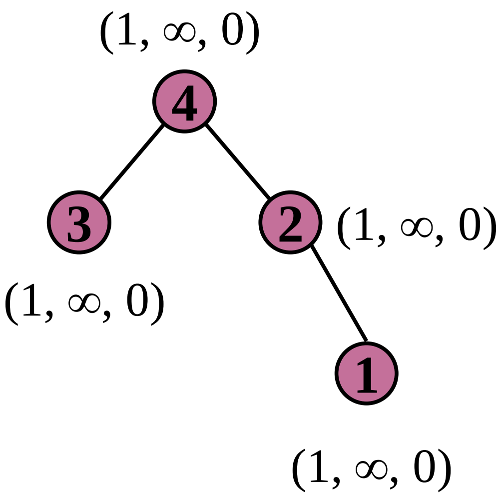


- **Time complexity = $\mathcal{O}(n)$.**

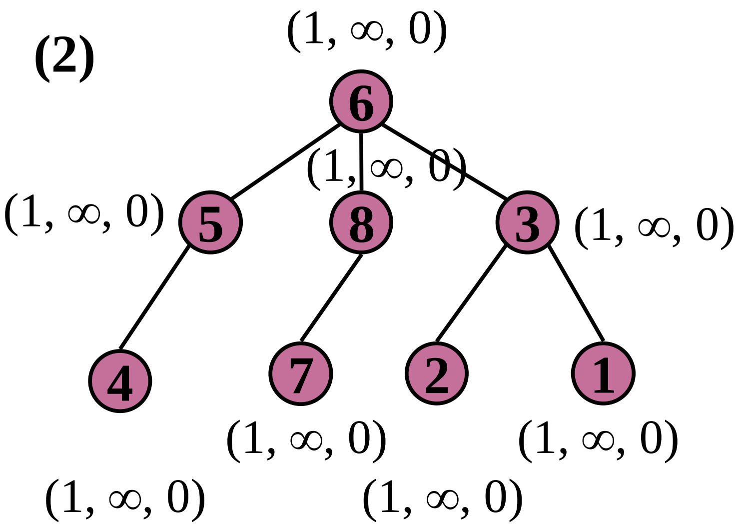
3.3 Method 3 : Dynamic Programming

- **Ex:**
- **(1)**





(2)



$$\gamma_1(v_j) = \gamma_1(v_j) + \min\{\gamma_1(v_i), \gamma_3(v_i)\}$$
$$\gamma_2(v_j) = \min\{\gamma_3(v_j) + \gamma_1(v_i), \gamma_2(v_j) + \gamma_2(v_i)\}$$
$$\gamma_3(v_j) = \gamma_3(v_j) + \min\{\gamma_1(v_i), \gamma_2(v_i)\}$$

3.3 Method 3 : Dynamic Programming

- **Exercise 2 (3/23): Use dynamic programming to solve the vertex covering problem.**

Computer Science and Information Engineering
National Chi Nan University

Combinatorial Optimization

Dr. Justie Su-Tzu Juan

Lecture 4. The Domination Problems on Interval Graphs

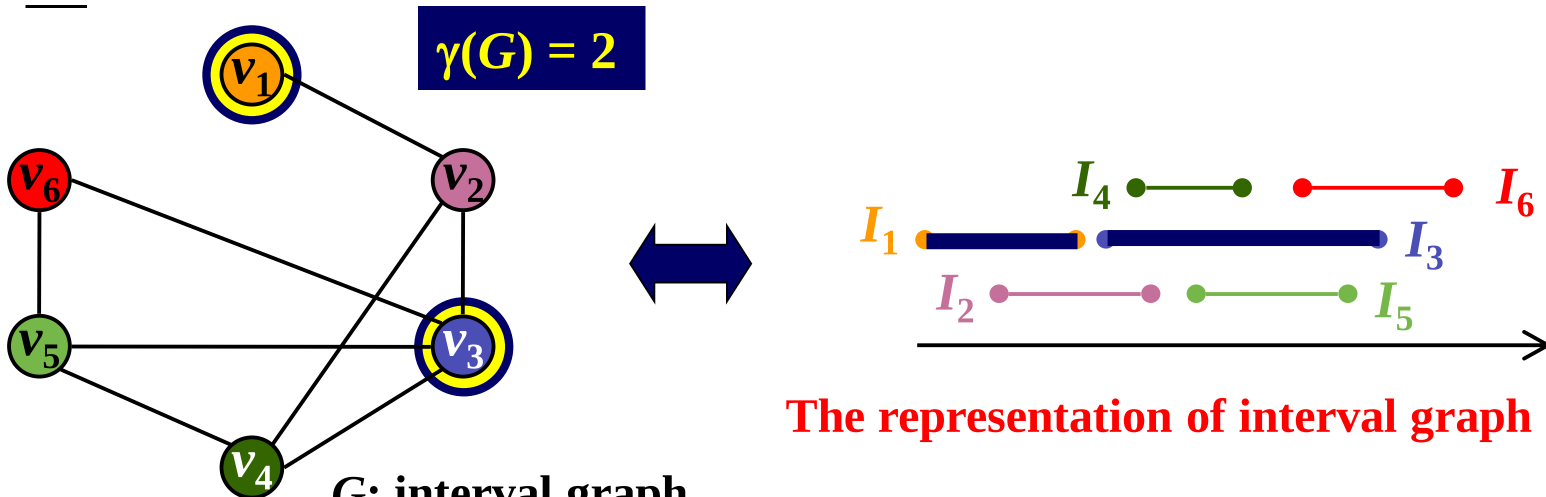
§4.1 Introduction to interval graph

4.1 Introduction to interval graph

- **Def: An interval graph is the intersection graphs of some (closed) intervals in the real lines.**

  i.e. $G = (V, E)$ is an interval graph for $V = \{v_1, v_2, \ldots, v_n\}$ if $\exists\ \mathscr{I} = \{I_1, I_2, \ldots, I_n\}$, each $I_i = [a_i, b_i] \in \mathbb{R}$ such that $E = \{v_iv_j \mid i \neq j$ and $I_i \cap I_j \neq \phi\}$.

- **Ex:**



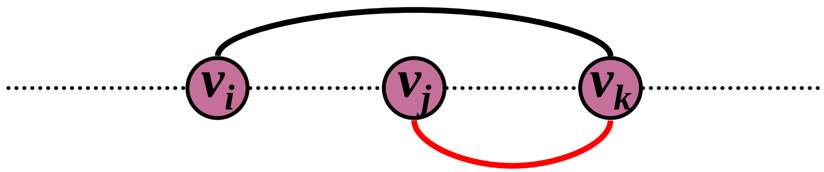
G: interval graph

The representation of interval graph G

4.1 Introduction to interval graph

- **Def**: Given graph $G = (V, E)$, an **interval ordering** of G is an ordering $[v_1, v_2, \ldots, v_n]$ of V, such that

$$i < j < k \text{ and } v_i v_k \in E \Rightarrow v_j v_k \in E \qquad (*)$$



- **Theorem**: G is an interval graph iff $\exists$ an interval ordering of G.

Proof. (1/2)

($\Rightarrow$)Let G be the intersection graph of $\{I_i = [a_i, b_i]: 1 \le i \le n\}$.

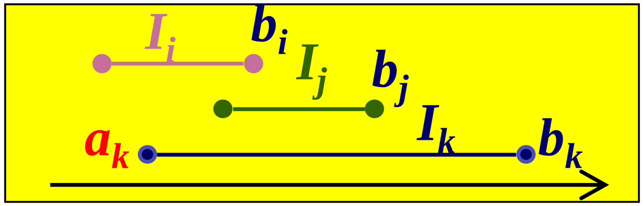
We may assume that $b_1 \le b_2 \le \ldots \le b_n$

If $i < j < k \Rightarrow b_i \le b_j \le b_k$

$v_i v_k \in E \Rightarrow I_i \cap I_k \neq \phi \Rightarrow a_k \le b_i$

$\because a_k \le b_i \le b_j \le b_k \Rightarrow I_j \cap I_k \neq \phi$ (Since $b_j \in [a_k, b_k]$)

$\Rightarrow v_j v_k \in E$



4.1 Introduction to interval graph

- **Theorem: G is an interval graph iff ∃ an interval ordering of G.**

Proof. (2/2)

(⇐)

Let i^* be the smallest index such that $v_{i^*} \in N[v_i]$.

Let $I_i = [i^*, i]$, $\forall\ i = 1, 2, \ldots, n$.

for any $i < j$:

① if $v_i v_j \in E$, then by def., $\therefore\ j^* \leq i < j \Rightarrow I_i \cap I_j \neq \phi$

② if $I_i \cap I_j \neq \phi$, then $j^* \leq i < j$

$\because$ by def, $v_{j^*} v_j \in E$

$\therefore$ by (∗), $v_i v_j \in E$.

$i < j < k$ and $v_i v_k \in E \Rightarrow v_j v_k \in E$ (∗)

Hence G is the intersection graph of $\{I_i \mid 1 \leq i \leq n\}$.

i.e. G is an interval graph.

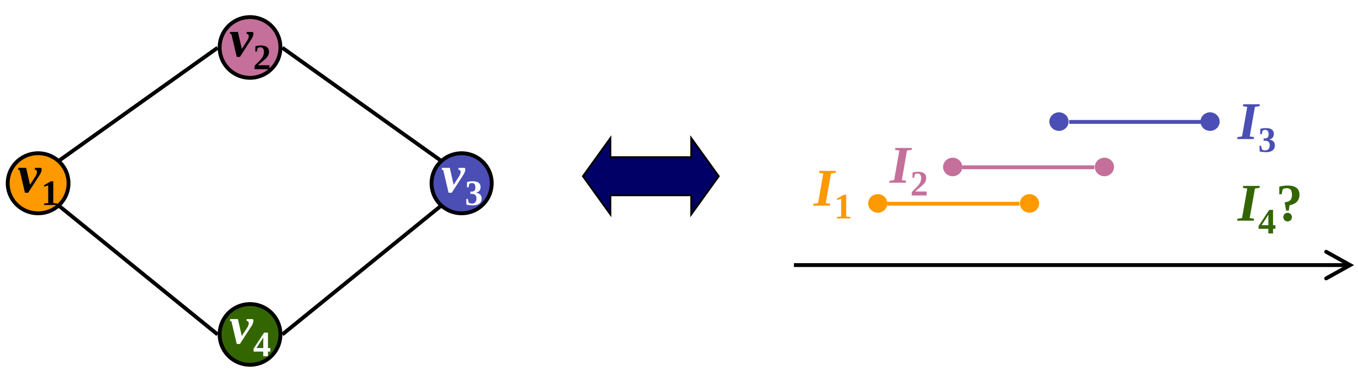
4.1 Introduction to interval graph

- **<u>Remark</u>: Booth and Lneker in 1976 gave an $\mathcal{O}(|V|+|E|)$-time algorithm for recognizing an interval graph and constructing.**

- **<u>Note</u>: For any interval graph G, there is no C_k, $k \geq 4$, be an induced subgraph of G.**

  (i.e. interval graph is chordal graph)

- **<u>Ex</u>:**



Computer Science and Information Engineering
National Chi Nan University

Combinatorial Optimization

Dr. Justie Su-tzu Juan

Lecture 4. The Domination Problems on Interval Graphs

§4.2 Primal-Dual Method

4.2 Primal-Dual Method

- **Algorithm 4.1:**

Given the interval set $\{I_i = [a_i, b_i] \mid 1 \le i \le n\}$, where $b_1 \le b_2 \le \ldots \le b_n$ according to the interval ordering of G.

$D^* \leftarrow \phi$;

$S^* \leftarrow \phi$;

for $i = 1$ **to** n **do**

  if $N[v_i] \cap D^* = \phi$ **then**

    Let $j \ge i$ **such that** $I_i \cap I_j \ne \phi$ **and** b_j **is largest;**

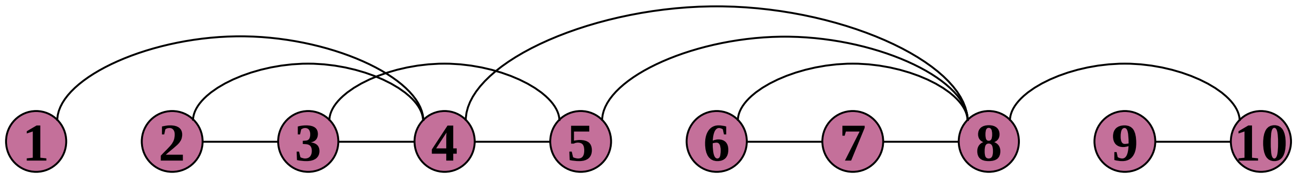
    $D^* \leftarrow D^* \cup \{v_j\}$;

    $S^* \leftarrow S^* \cup \{v_i\}$

- **Time Complexity = ?**

4.2 Primal-Dual Method

- **Ex:**



$D^* = \{ 4, 8, 10 \}$

$S^* = \{ 1, 6, 9 \}$

if $N[v_i] \cap D^* = \phi$ **then**

Let $j \geq i$ **such that** $I_i \cap I_j \neq \phi$ **and** b_j **is largest;**

$D^* \leftarrow D^* \cup \{v_j\}$;

$S^* \leftarrow S^* \cup \{v_i\}$

4.2 Primal-Dual Method

- **<u>Thm</u>: ① D^* is a dominating set.**
  ② $|D^*| \le |S^*|$.
  ③ S^* is a 2-stable set.

- **<u>Note</u>: ❶ D^* is a optimal dominating set.**
  ❷ $\alpha_2(G) = \gamma(G)$.
  ❸ S^* is a optimal 2-stable set.
  $\because$ $|S^*| \le \alpha_2(G) \le \gamma(G) \le |D^*| \le |S^*|$
  $\therefore$ **all "≤" are "=".**

4.2 Primal-Dual Method

- **Thm: ① D^* is a dominating set.**

  ② $|D^*| \le |S^*|$.

Proof.

① $\forall v_i \in V$, if $N[v_i] \cap D^* = \phi$, then

algorithm 會加入v_j到D^*中, where

$\because I_i \cap I_j \neq \phi$, $\therefore v_iv_j \in E$.

i.e. 對最後的D^*而言, $N[v_i] \cap D^* \neq \phi$.

② Algorithm中，每次加入v_j到D^*中時，必加一新點v_i到S^*中

$\therefore |S^*| \ge |D^*|$.

- **Notation:** $x \sim y$表示$x \in N[y]$ (also, $y \in N[x]$)

4.2 Primal-Dual Method

- **<u>Thm</u>: ③ S^* is a 2-stable set.**

Proof.

③ Suppose $\exists\, v_i, v_{i'} \in S^*$, for $i < i'$ and $d(v_i, v_{i'}) \le 2$.

i.e. $\exists\, j'$ such that $v_i \sim v_{j'}$ and $v_{j'} \sim v_{i'}$.

當algorithm執行到 i iteration 時：

會找出v_j放入D^*中，其中j滿足$v_iv_j \in E$, $j \ge i$ is largest.

當algorithm執行到 i' iteration 時：$v_j \in D^*$

Case 1: $i < i' \le j$

By (∗), $v_{i'} \sim v_j$, since $v_i \sim v_j$.

Case 2: $i \le j < i'$

By the choice of j, we have $j' \le j$

$\because j' \le j < i'$ and $v_{i'} \sim v_{j'}$

$\therefore$ By (∗), $v_j \sim v_{i'}$.

In both case, $v_{i'} \sim v_j$ and $v_j \in D^*$

$\therefore$ 不會將 $v_{i'}$放入S^*中. $\rightarrow\leftarrow$