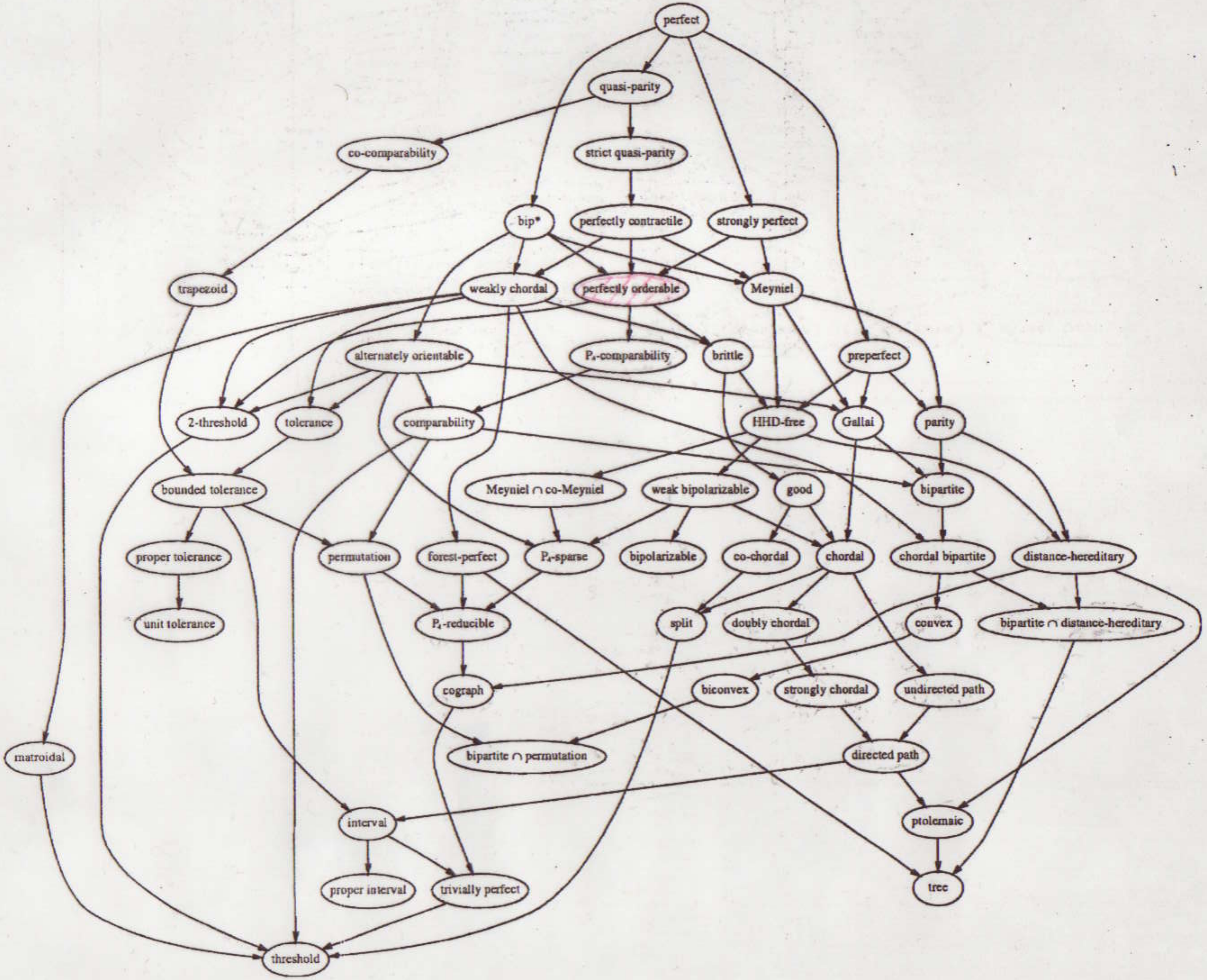
ΑΛΓΟΡΙΘΜΙΚΗ ΘΕΩΡΙΑ ΓΡΑΦΗΜΑΤΩΝ

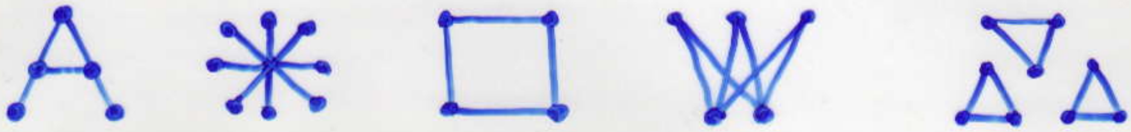
- Βασικοί Αλγόριθμοι Γραφημάτων
- Πολυπλοκότητα: O, Ω
- Τέλεια Γραφήματα
  - Κλάσεις
  - προβλήματα αναγνώρισης
  - προβλήματα βελτιστοποίησης.
- Modular Διάσπαση
- Αλγόριθμοι στα Συμπληρώματα των Γραφημάτων
- Αλγόριθμοι Μέτρησης
- Αλγόριθμοι Ευστάθειας Δικτύων.
- Project
- Τελικές Εξετάσεις



perfect
quasi-parity
co-comparability
strict quasi-parity
bip*
perfectly contractile
strongly perfect
trapezoid
weakly chordal
perfectly orderable
Meyniel
alternately orientable
P_4-comparability
brittle
preperfect
2-threshold
tolerance
comparability
HHD-free
Gallai
parity
bounded tolerance
Meyniel ∩ co-Meyniel
weak bipolarizable
good
bipartite
proper tolerance
permutation
forest-perfect
P_4-sparse
bipolarizable
co-chordal
chordal
chordal bipartite
distance-hereditary
unit tolerance
P_4-reducible
split
doubly chordal
convex
bipartite ∩ distance-hereditary
cograph
biconvex
strongly chordal
undirected path
matroidal
bipartite ∩ permutation
directed path
interval
ptolemaic
proper interval
trivially perfect
tree
threshold

◎ Graph Theoretic Foundations

- Graph $G=(V,E)$

- 

  A graph $K_{1,6}$ C_4 $K_{3,2}$ $3K_3$

- $G=(V,E)$ and $G'=(V',E')$ are isomorphic, denoted $G \cong G'$, if $\exists$ a bijection $f: V \to V'$:

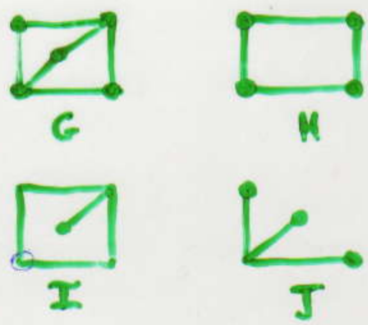
$$(x,y) \in E \Leftrightarrow (f(x), f(y)) \in E'$$

$\forall\, x, y \in V$.

- Let $A \subseteq V$. We define the subgraph induced by A to be $G_A = (A, E_A)$, where

$$E_A = \{ xy \in E \mid x \in A \text{ and } y \in A \}$$

- Not every subgraph of G is an induced subgraph of G.



- clique number $\omega(G)$
  the number of vertices in a maximum clique of G.

- stability number $\alpha(G)$
  the number of vertices in a stable set of max cardin.

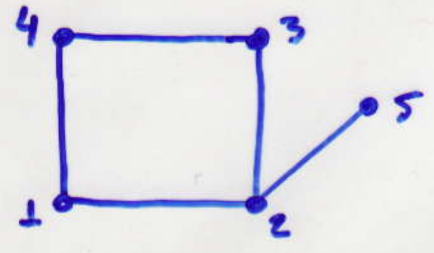
- A clique cover of size k is a partition
$$V = C_1 + C_2 + \cdots + C_k$$
  such that C_i is a clique.

- A proper c-coloring is a partition
$$V = X_1 + X_2 + \cdots + X_c$$
  such that X_i is a stable set.

- clique-cover number $k(G)$
  the size of the smallest possible clique cover of G.

- chromatic number $\chi(G)$
  the smallest possible c for which there exists a proper c-coloring of G.

- Example



G:

$\omega(G) = 2$

$\alpha(G) = 3$

clique cover

$V = \{2,5\} + \{3,4\} + \{1\}$

c-coloring

$V = \{1,3,5\} + \{2,4\}$

$k(G) = 3 \quad \chi(G) = 2$

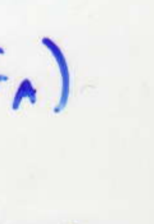
- For any graph G: $\omega(G) \leq \chi(G)$
  $\alpha(G) \leq k(G)$
- Obviously: $\alpha(G) = \omega(\bar{G})$ and $k(G) = \chi(\bar{G})$.

- Let $G = (V, E)$ be an undirected graph:

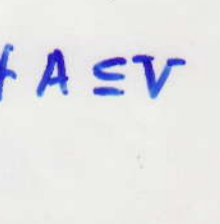
$(P_1) \quad \omega(G_A) = \chi(G_A) \quad \forall A \subseteq V$

$(P_2) \quad \alpha(G_A) = k(G_A) \quad \forall A \subseteq V$

G is called Perfet.



C_5



Grötzsch

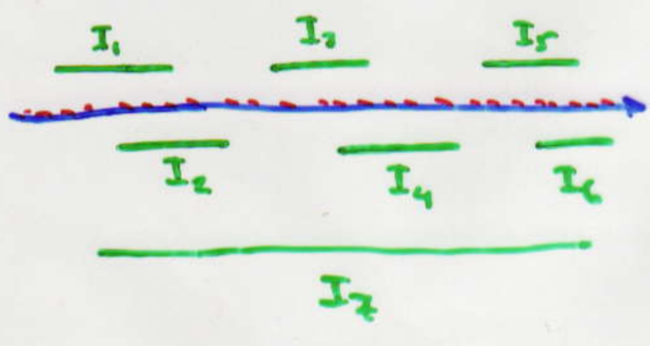
Intersection Graphs

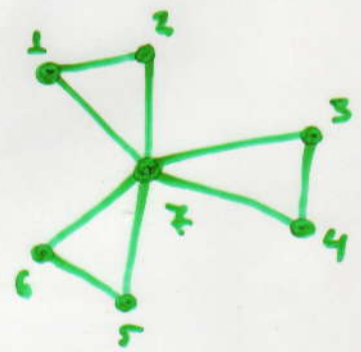
- Let $\mathcal{F}$ be a family of nonempty sets.

- The **intersection graph** of $\mathcal{F}$ is obtained by representing each set in $\mathcal{F}$ by a vertex:

$$x \to y \Leftrightarrow S_x \cap S_y \neq \emptyset$$

- The intersection graph of a family of intervals on a linearly ordered set (like the real line) is called an **interval graph**.



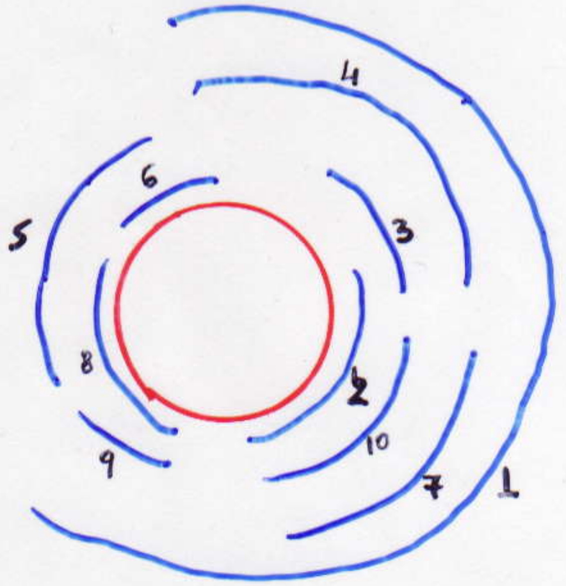


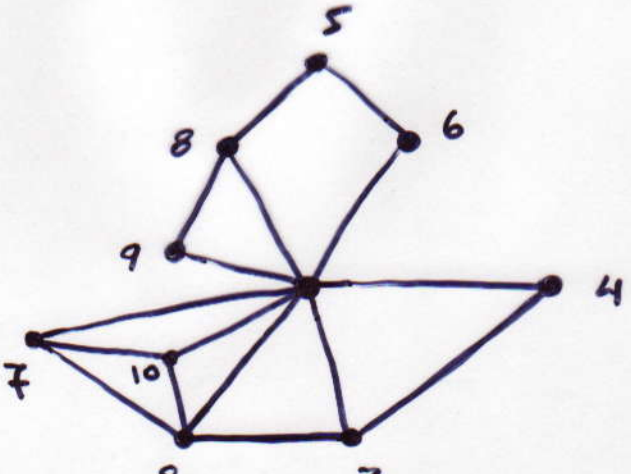
- **unit interval graph**
- **proper interval graph**
  - no interval properly contains another

- Consider the following relaxation:
  if we join the two ends of our line, the intervals will become arcs on the circle.

- Allowing arcs to slip over, we obtain a class of intersection graphs called the circular-arc graphs.

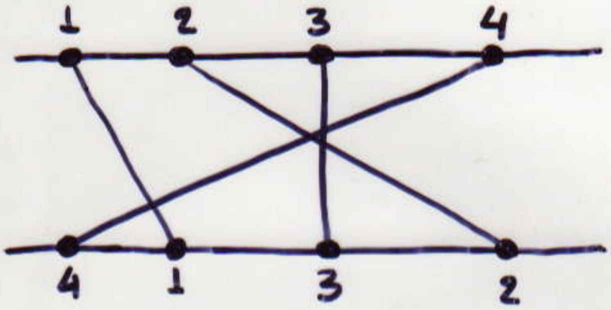
- Circular-arc graphs properly contain the interval graphs.



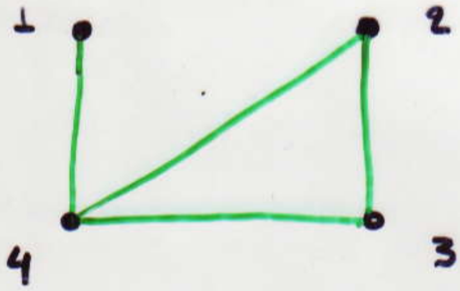


- proper circular-arc graphs

- A permutation diagram consists of n points on each of two parallel lines and n straight line segments matching the points.

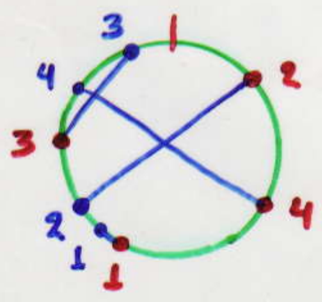
- 

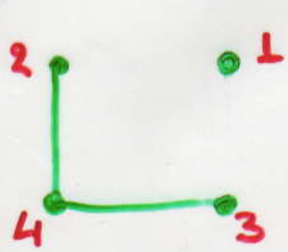
$\pi = [4,1,3,2]$



$G[\pi]$

- Intersecting chords of a circle





• Proposition 1.1. An induced subgraph of an interval graph is an interval graph.

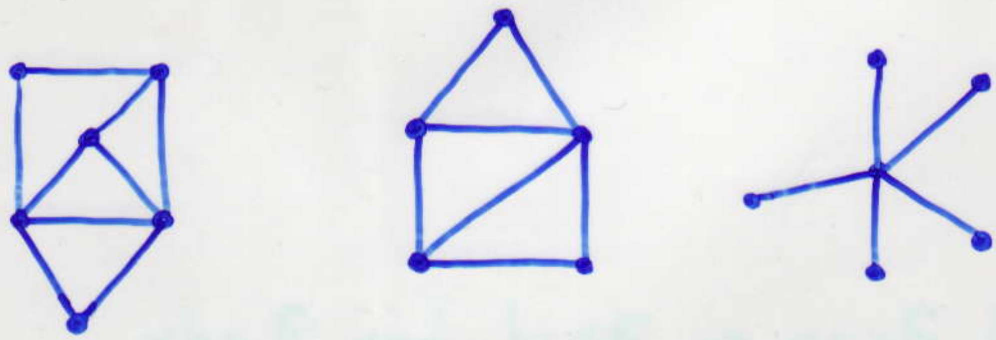
Proof. If $\{I_v\}_{v\in V}$ is an interval representation of a graph $G=(V,E)$. Then $\{I_v\}_{v\in X}$ is an interval representation of the induced subgraph $G_X=(X,E_X)$.

• Triangulated graph property

Every simple cycle of length $\ell > 3$ possesses a chord.

• Triangulated graphs (or chord graphs)

•



- Proposition 1.2. An interval graph satisfies the triangulated graph property.

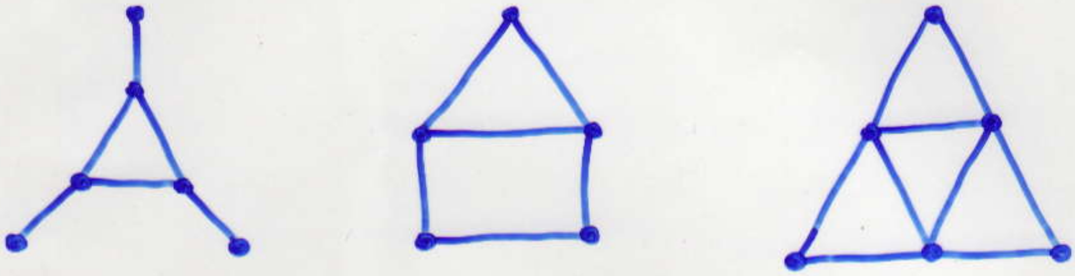
Proof. Suppose G contains $[v_0, v_1, \dots, v_{\ell-1}, v_0]$, with $\ell > 3$. Let $I_k \to v_k$.
For $i = 1, 2, \dots, \ell-1$, choose a point $P_i \in I_{i-1} \cap I_i$.
Since I_{i-1} and I_{i+1} do not overlap, the points P_i constitute a strictly increasing or decreasing sequence.
Therefore, it is impossible for I_0 and $I_{\ell-1}$ to intersect, contradicting the criterion that $v_0 v_{\ell-1}$ is an edge of G.

- Transitive orientation property

Each edge can be assigned a one-way direction in such a way that the resulting oriented graph (V, F) :

$$ab \in F \text{ and } bc \in F \Rightarrow ac \in F \quad (\forall a, b, c \in V)$$

- Graphs which satisfy the transitive orientation property are called comparability graph.



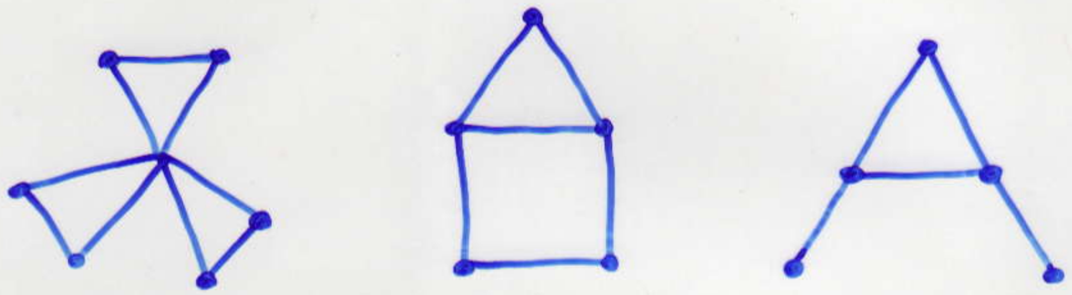
- Proposition 1.3. The complement of an interval graph satisfies the transitive orientation property.

Proof Let $\{I_v\}_{v\in V}$ be the interv. repres. for $G=(V,E)$. Define an orientation F of $\bar{G}=(V,\bar{E})$ as follows:

$$xy\in F \iff I_x < I_y \quad (\forall xy\in \bar{E}).$$

Here, $I_x < I_y$ means that I_x lies entirely to the left of I_y. Clearly the t.o.p is satisfied, since $I_x < I_y < I_z \Rightarrow I_x < I_z$.
Thus, F is a transitive orientation of $\bar{G}$.

• Theorem 1.4. An undirected graph G is an interval graph iff G is triangulated graph and its complement $\bar{G}$ is a comparability graph.

• 

Each of the graphs can be colored using 3 colors and each contains a triangle. Therefore, $\chi = \omega$

• χ-Perfect property. For each induced subg. G_A of G

$$\chi(G_A) = \omega(G_A)$$

• α-Perfect property. For each induced subg. G_A of G

$$\alpha(G_A) = k(G_A)$$

The Design of Efficient Algorithms

- Computability – Computational complexity

- Computability addresses itself mostly to questions of existence: Is there an algorithm which solves problem Π?

- An algorithm for Π is a step-by-step procedure which when applied to any instance of Π produces a solution

- Rewrite an optimization problem as a decision probl.

Graph Coloring
Instance: A graph G
Question: What is the smallest number of colors needed for a proper coloring of G?

Graph Coloring
Instance: G and $k \in \mathbb{Z}_+$
Question: Does there exist a proper k coloring of G?

- Determining the complexity of a problem Π requires a two-sided attack:

  (1) The upper bound – the minimum complexity of all known algorithms solving Π.

  (2) The lower bound – the largest function f for which it has been proved (mathematically) that all possible algorithms solving Π are required to have complexity at least as high as f.

- Gap between (1) – (2) ⇒ research

- Example: matrix multiplication

  - Strassen [1969] $O(n^{2.81})$
  - Pan [1979] $O(n^{2.78})$

    $O(n^{2.6054})$ $n \gg$
  - The lower bound known to date for this problem is only $O(n^2)$ [Aho, Hopcroft, Ullman, 1974, pp 438]

- The biggest open question involving the gap between upper and lower complexity bounds involves the so called NP-complet problems.

- $\Pi \in$ NP-complete $\Rightarrow$ only exponential-time algorithms are known,
yet the best lower bouds proven so far are polynomial functions.

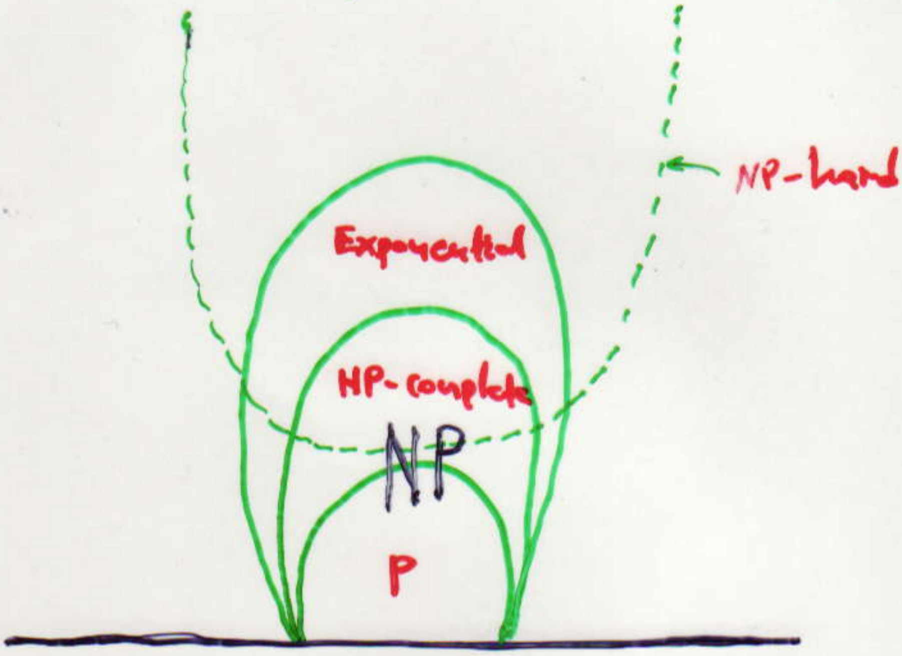
- $\Pi \in P$ if there exists a "deterministic" polynomial-time algorithm which solves Π.

- A nondeterministic algorithm is one for which a state may determine many next states and which follows up on each of the next states simultaneously.

- $\Pi \in NP$ if there exists a "nonderministic" polynomial-time algorithm which solves Π.

- clearly, $P \subseteq NP$.

- Open question is whether the containment of P in NP is proper — is $P \neq NP$?



- $\Pi \in$ NP-complete if $\Pi \in NP$ & $\Pi \in$ NP-hard.

- Repeat the following instructions:
  (1) Find a candidate Π which might be NP-complete
  (2) Select Π' from the bag of NP-complete problems.
  (3) Show that $\Pi \in NP$ and $\Pi' \leq \Pi$.
  (4) Add Π to the bag.

- Theorem (Poljak [1974]):

STABLE SET $\leq$ STABLE SET ON TRIANGLE-FREE GRAPHS

Proof.

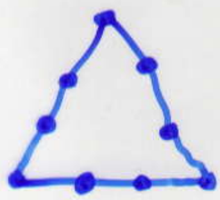
Let G be a graph on n vertices and m edges.
We construct from G
a triangle-free graph H with
the property that:

G

Knowing $\alpha(H)$ will immediately give us $\alpha(G)$.

Subdivide each edge of G into a path of length 3

H is triangle-free with
$n+2m$ vertices, and
$3m$ edges.



H

Also, H can be constructed from G in $O(n+m)$.

Finally, since $\alpha(H) = \alpha(G) + m$, a deterministic polynomial time algorithm which solves for $\alpha(H)$ yields a solution to $\alpha(G)$.

- Since it is well known that STABLE SET is NP-complete, we obtain the following lesser known result.

Corollary: STABLE SET ON TRIAGLE-FREE GRAPHS is NP-complete.

- Theorem (Poljak [1974]):

  STABLE SET $\preceq$ GRAPH COLORING.

- Some NP-complete Problems

- Graph coloring

  instance: G.

  question: What is $\chi(G)$?

- Clique

  instance: G.

  question: what is $\omega(G)$?

- Stable set

  instance: G.

  question: what is $\alpha(G)$?

- Clique Cover

  instance: G.

  question: what is $k(G)$?

- Perfect graphs $\Rightarrow$ Optimization Problem?

$x \leftarrow$ **choice**(S) creates $|S|$ copies of the algorithm, and assigns every member of the set S to the variable x in one of the copies.

failure causes that copy of the algorithm to stop execution.

success causes all copies of the algorithm to stop execution and indicates a "yes" answer to that instance of the problem.

A nondeterministic polynomial-time algorithm for the decision version of the CLIQUE problem is the following: Let $G = (V, E)$ be an undirected graph and let $k \geq 0$.

```
   procedure CLIQUE(G, k):
   begin
1.  A ← ∅;
2.  for all v ∈ V do A ← choice({A + {v}, A});;
3.  if |A| < k then failure;
4.  for all v, w ∈ A, v ≠ w do
5.     if vw ∉ E then failure;;
6.  success;
   end
```

The loop in line 2 nondeterministically selects a subset of vertices $A \subseteq V$; lines 4–6 decide if A is a complete set. If **success** is reached in one of the copies, then the final value of A in that copy is a clique of size at least k. Using the above procedure we obtain a nondeterministic polynomial-time algorithm for the optimization version of the CLIQUE problem as follows: Let G be an undirected graph with n vertices.

```
procedure MAXCLIQUE(G):
begin
  for k ← n to 1 step −1 do
    if CLIQUE(G, k) then return k;
end
```

• Analysis of Parallel Algorithms

- A **parallel computer** is simply a collection of processors, typically of the same type, interconnected in a certain fashion to allow the coordination of their activities and the exchange of data.

- Our main goal is to present algorithms that are suitable for implementation on parallel computers.

- The **running time** $t(n)$ or $T(n)$ of a parallel algorithm is defined as the time required by the algorithm to solve a computational problem.

- For a problem of size n, if the **number of processors** required by a parallel algorithm is a function of n, then it is denoted by $p(n)$ or $P(n)$.

- Measuring the performance of a parallel algorithm

- Cost: $c(n) = t(n) \cdot p(n)$

- Assume tha a lower bound of $\Omega(f(n))$ is known on the number of steps required in the worst-case to solve one problem of size n.

- If the cost of a parallel algorithm is $O(f(n))$, then the algorithm is said to be asymptotically cost optimal.

- Speedup: $S(1,p) = \dfrac{t_1(n)}{t_p(n)}$

  - A good parallel algorithm is one for which this ratio is large.

* Efficiency : $E(1,p) = \dfrac{t_1(n)}{C(n)}$

(1) If $E(1,p) < 1$, then the parallel algorithm is not cost optimal.

(2) If $E(1,p) = 1$, then the parallel algorithm is cost optimal, provided that the sequential algorithm is time optimal.

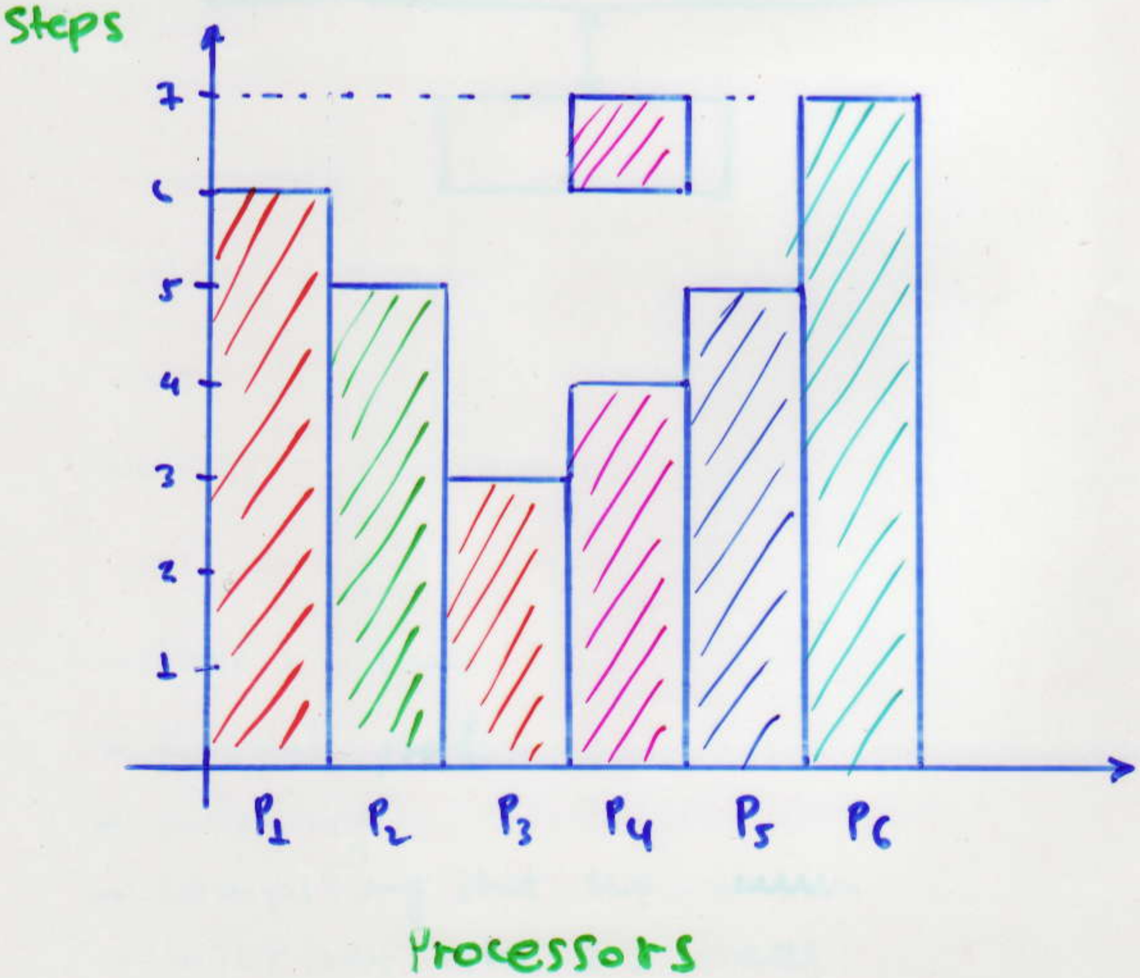
(3) If $E(1,p) > 1$, then a faster sequential algorithm can be obtained by simulating the parallel one.

* Work : $W(n)$ measures the total number of operations used by the algorithm.

- The work $W(n)$ has nothing to do with the number of processors avaibable.

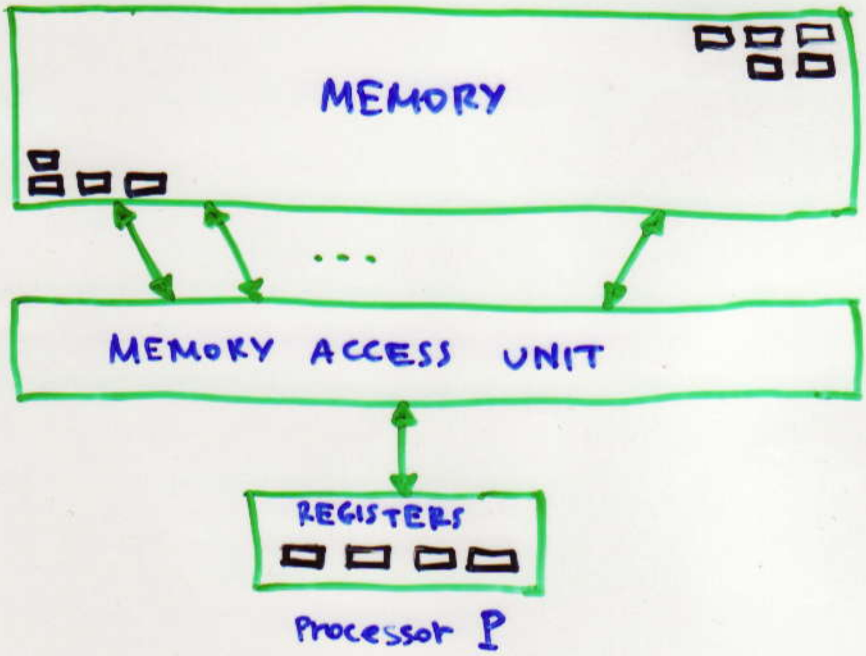
- The cost $C(n)$ measures the cost of the algorithm relative to the number $p(n)$ of processors available.

- Work versus Cost.



• Models of Computation

• Random Access Machine (RAM)



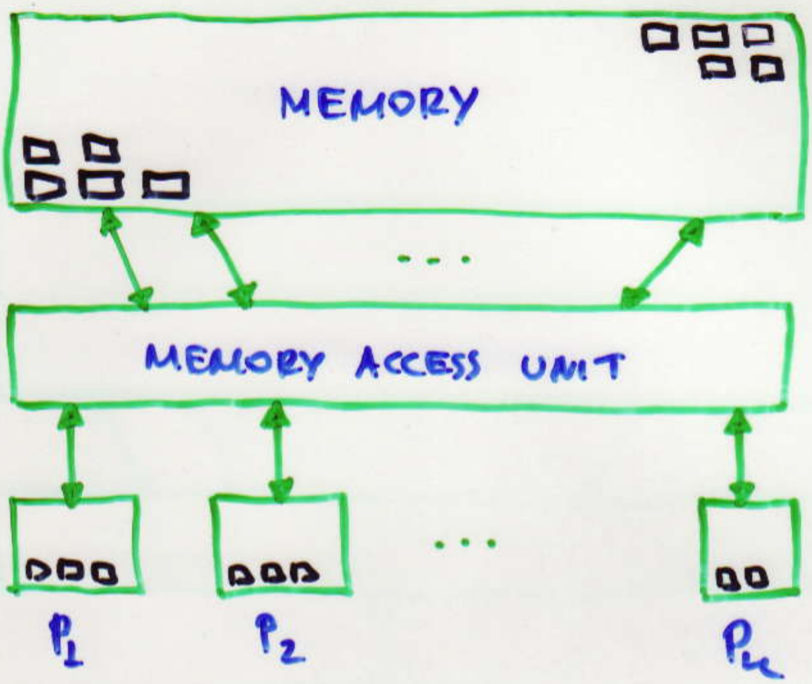
• Each step of the algorithm consists of (up to) 3 phases:

(1) READ phase

(2) COMPUTE phase

(3) WRITE phase

- Parallel Random Access Machine (PRAM)



- READ phase: processors (up to n) read simultaneously from (up to n) memory locations.

- COMPUTE phase: processors (up to n) perform basic arithmetic or logical operations on their local data.

- WRITE phase: processors (up to n) write simult. into (up to n) memory locations.

- Memory Access

  - EREW
  - CREW
  - ERCW
  - CRCW

- The CW instruction

  (1) PRIORITY CW

  (2) COMMON CW

  (3) ARBITRARY CW

- Basic Techniques
  - Prefix sums
  - Parallel prefix
  - Merging
  - Computing the maximum
  - Insertion into 2-3 trees
  - Convex hull
  - Coloring the vertices of a dag

Basic Techniques

- We introduce some basic techniques and apply them to a selected set of combinatorial problems, which are interesting on their own and often appear as sub-problems in numerous computations.

(I) ALS Partition

- Given a connected graph $G=(V,E)$ and a vertex $v\in V$, we define a partition $\mathcal{L}(G,v)$ of the set V (we shall use the term partition of the graph G), with respect to the vertex v as follows:

$$\mathcal{L}(G,v)=\{\, N_i(v) \mid v\in V,\ 0\le i\le L_v,\ 1\le L_v\le |V| \,\}$$

where $N_i(v)$, are the adjacency-level sets, and L_v is the length of the partition $\mathcal{L}(G,v)$.

- The adjacency-level sets of the partition $\mathcal{L}(G,v)$ of the graph $G=(V,E)$, are formally defined as follows:

$$N_i(v)=\{u\in V \mid d(v,u)=i\}$$

where $d(v,u)$ denotes the distance v-u in G.

- $d(v,u)\geq 0$, and $d(v,u)=0$ where $v=u$.

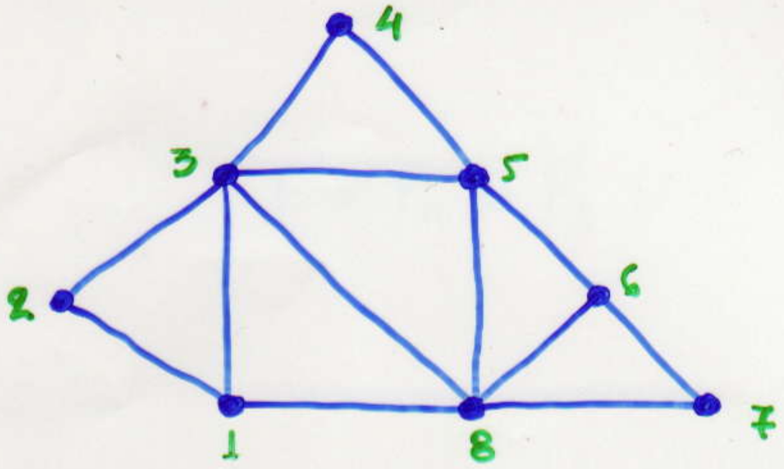
- If G is a disconnected graph $\Rightarrow$
$d(v,u)=\infty$ where $x\in CC_i$ and $y\in CC_j$, $i\neq j$.

- Obviously, $L_v=\max\{d(v,u) \mid u\in V\}$
and

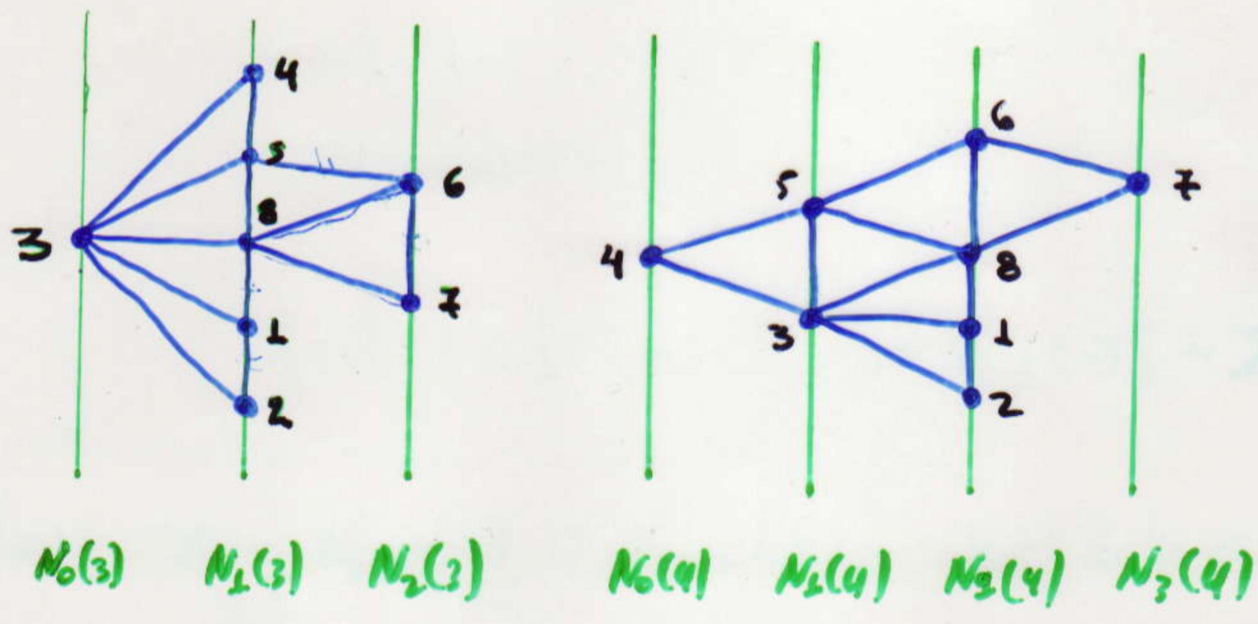
$$N_0(v)=\{v\} \quad \text{and} \quad N_1(v)=N(v)$$

- Note that: $N(v)=adj(v)$
$N[v]=\{v\}\cup N(v)$.

- Let G be the following graph:



- The ALS of G are:



- The ALS have the following properties:

$$N_i(v) \cap N_j(v) = \phi \qquad \forall\, i \neq j$$
$$N(x) \cap N_{i-1}(v) \neq \phi \qquad \forall\, x \in N_i(v)$$
$$N(x) \cap N_{i-2}(v) = \phi \qquad \forall\, x \in N_i(v)$$

and

$$V = N_0(v) + N_1(v) + \cdots + N_{L_v}(v)$$

- The ALS of $\mathcal{L}(G,v)$, can be computed recursively as follows:

$$N_0(v) = \{v\}, \quad v \in V$$
$$N_1(v) = \mathrm{adj}(v)$$

and

$$N_i(v) = \{ u \mid (x,u) \in E,\ x \in N_{i-1}(v) \} - X$$

where $X = N_{i-1}(v) \cup N_{i-2}(v)$, $\quad 2 \leq i \leq L_v < n$.

- ALS can also be computed by the distance matrix of the graph G.

(II) FSA Decomposition

- Given a graph G, an edge $(x,y) = (y,x)$ of G is classified as follows according to relationship of closed neighbourhoods:

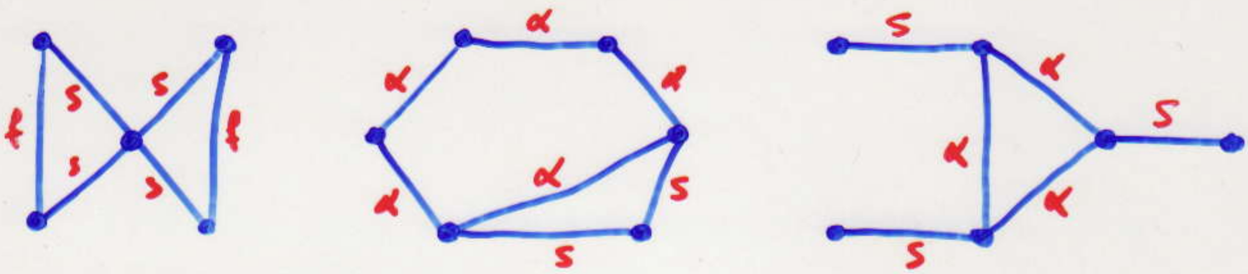
(x,y) is free if $N[x] = N[y]$
(x,y) is semi-free if $N[x] \subset N[y]$
(x,y) is actual if $N[x] \setminus N[y] \neq \phi$ and $N[y] \setminus N[x] \neq \phi$.

- We denote: FE the set of free edges
  SE " semi-free "
  AE " actual "

- Then,

$$E(G) = FE + SE + AE$$

•



• A graph G is called an A-free if every edge of G is either free or semi-free.

• We define the cent of a graph G as follows:

$$\mathrm{cent}(G) = \{ x \in V(G) \mid N[x] = V(G) \}$$

• Theorem 1. Let G be a simple graph. The the following statements are equivalent.

(i) G is a A-free graph;

(ii) G has no induced subgraphs isomorphic to P_4 or C_4;

(iii) Every connected induced subgraph $G[S]$, $S \subseteq V(G)$, satisfies $\mathrm{cent}(G[S]) \neq \emptyset$.

(III) G-Decomposition

- We define the binary relation Γ on the edges of an undirected graph $G=(V,E)$ as follows:

$$\alpha b \,\Gamma\, \alpha' b' \text{ iff } \begin{cases} \text{either} & \alpha=\alpha' \text{ and } bb' \notin E \\ \text{or} & b=b' \text{ and } \alpha\alpha' \notin E. \end{cases}$$

- We say that αb directly forces $\alpha' b'$ whenever $\alpha b \,\Gamma\, \alpha' b'$.

- Since E is irreflexive ($bb \notin E$), $\alpha b \,\Gamma\, \alpha b$; ($\alpha = \alpha$ and $bb \notin E$) (irreflexive $x \notin R(x)$) however $\alpha b \not\Gamma\, b\alpha$.

- The reflexive ($x \in R(x)$, $x \in X$), transitive closure Γ^* of Γ partitions E into what we call the implication classes of G.

- Thus edges αb and cd are in the same implication class iff there exists a sequence of edges

$$\alpha b = \alpha_0 b_0 \,\Gamma\, \alpha_1 b_1 \,\Gamma \cdots \Gamma\, \alpha_k b_k = cd, \quad \text{with } k \geq 0$$

- Such a sequence is called a Γ-chain from ab to cd, and we say that αb forces cd whenever $\alpha b \,\Gamma^* cd$.

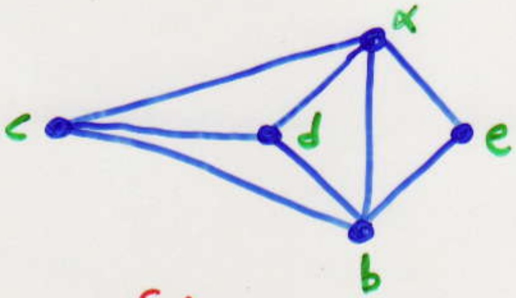
- Let $\mathcal{I}(G)$ denote the collection of implication classes of G. We define

$$\hat{\mathcal{I}}(G) = \{ \hat{A} \mid A \in \mathcal{I}(G) \}$$

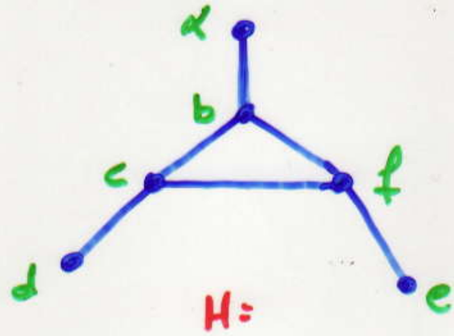
where $\hat{A} = A \cup A^{-1}$

- The members of $\hat{\mathcal{I}}(G)$ are called the color classes of G.

- **Examples:** Let G and H be the following graphs:



G:



H:

- The graph G has eight implication classes:

$A_1 = \{\alpha b\}$ $\quad A_2 = \{cd\}$, $\quad A_3 = \{\alpha c, \alpha d, \alpha e\}$, $\quad A_4 = \{bc, bd, be\}$

$A_1^{-1} = \{b\alpha\}$ $\quad A_2^{-1} = \{dc\}$ $\quad A_3^{-1} = \{c\alpha, d\alpha, e\alpha\}$ $\quad A_4^{-1} = \{cb, db, eb\}$

So we have $\hat{\mathcal{I}}(G) = \{\hat{A}_1, \hat{A}_2, \hat{A}_3, \hat{A}_4\}$.

- On the other hand, H has only one implication class:

$$A = \{\alpha b, cb, cd, cf, ef, bf, b\alpha, bc, dc, fc, fe, fb\}$$

and $A = \hat{A}$.

- Let $G=(V,E)$ be an undirected graph. A partition of the edge set

$$E(G)=\hat{B}_1+\hat{B}_2+\cdots+\hat{B}_k$$

is called a G-decomposition of $E(G)$ if B_i is an implication class of

$$\hat{B}_i+\hat{B}_{i+1}+\cdots+\hat{B}_k$$

for all $i=1,2,\ldots,k$.

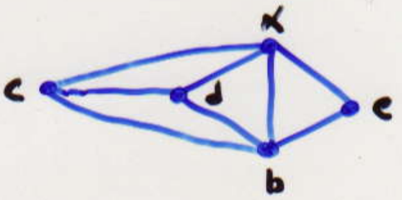
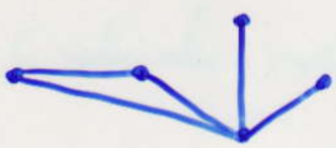
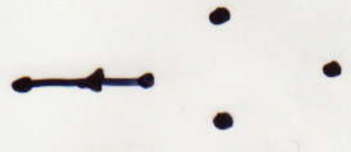
- A sequence of edges $[x_1y_1, x_2y_2, \ldots, x_ky_k]$ is called a decomposition scheme for G if there exists a G-decomposition $E(G)=\hat{B}_1+\hat{B}_2+\cdots+\hat{B}_k$ satisfying $x_iy_i\in B_i$ for all $i=1,2,\ldots,k$.

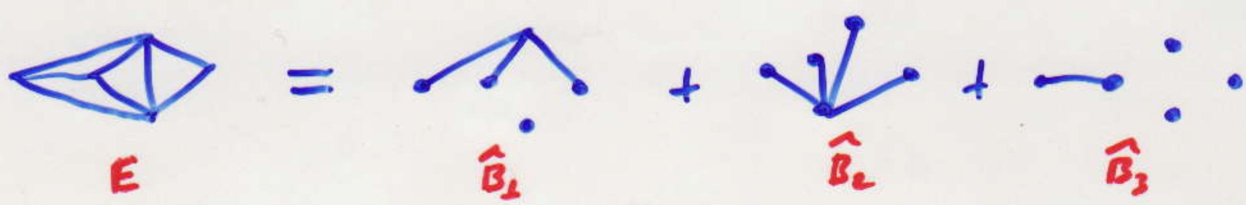
- Algorithm G-decomposition

Input: $G=(V,E)$

Initially: $i=1$, $E_1=E$.

1. Arbitrarily pick an edge $e_i = x_i y_i \in E_i$;
2. Enumerate the impl. class B_i of E_i containing $x_i y_i$
3. Define $E_{i+1} = E_i - \hat{B}_i$
4. If $E_{i+1} = \emptyset$ then let $k=i$ and Stop;
   otherwise, $i = i+1$ and goto Step 1.

(V, E_i)	$x_i y_i$	(V, B_i)
	ac	
	bc	
	dc	



(IV) Ear Decomposition

- BFS and DFS are two graph-traversal methods that have been found to be effective in handling many graph-theoretic problems.

- However, no efficient parallel implementations of these two methods are known at this point.

- We introduce the technique of ear decomposition which does have an efficient parallel implementation.

- An ear decomposition is essentially an ordered partition of the set $E(G)$ into simple paths (which include simple cycles).

- Let $G = (V, E)$ be an undirected graph, with $|V| = n$ and $|E| = m$.

- An ear decomposition of G starting with P_0 is an ordered partition of the set $E(G)$

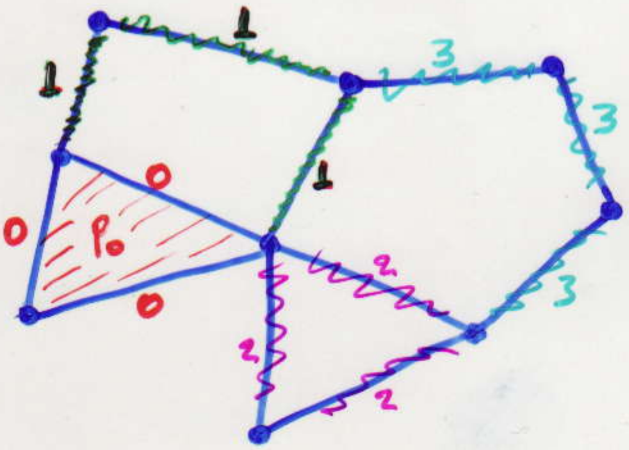
$$E = P_0 \cup P_1 \cup \cdots \cup P_k$$

such that, for each $1 \le i \le k$,

P_i is a simple path whose endpoints belong to $P_0 \cup P_1 \cup \cdots \cup P_{i-1}$, but none of whose internal vertices does.

- Each simple path P_i is called ear.

- If, for each $i > 0$, P_i is not a cycle, the decomposition is called an open ear decomposition.

- 

- **Theorem**: An undirected graph $G=(V,E)$ has an ear decomposition iff it is bridgeless. The graph G has an open ear decomposition iff it is biconnected.

cutpoint

- Let G have an open ear decomposition

$$E = P_0 \cup P_1 \cup \cdots \cup P_k$$

- Hence, P_0 is a simple cycle
  P_1 is a simple path; endpoints $\in P_0$
  P_2 is a simple path; endpoints $\in P_0 \cup P_1$
  $\vdots$

- Actually, removing an arbitrary edge from each ear $\Rightarrow$ spanning tree of G.

- Therefore, the number of ears is equal to the number $(m-n)+1$ of nontree edges.

- On the other hand, we know that a cycle basis can be generated from an arbitrary spanning tree of G by the nontree edges.

- This observation suggests the following method to obtain an ear decomposition.

— We label each nontree edge $e = (u, v)$ as follows:

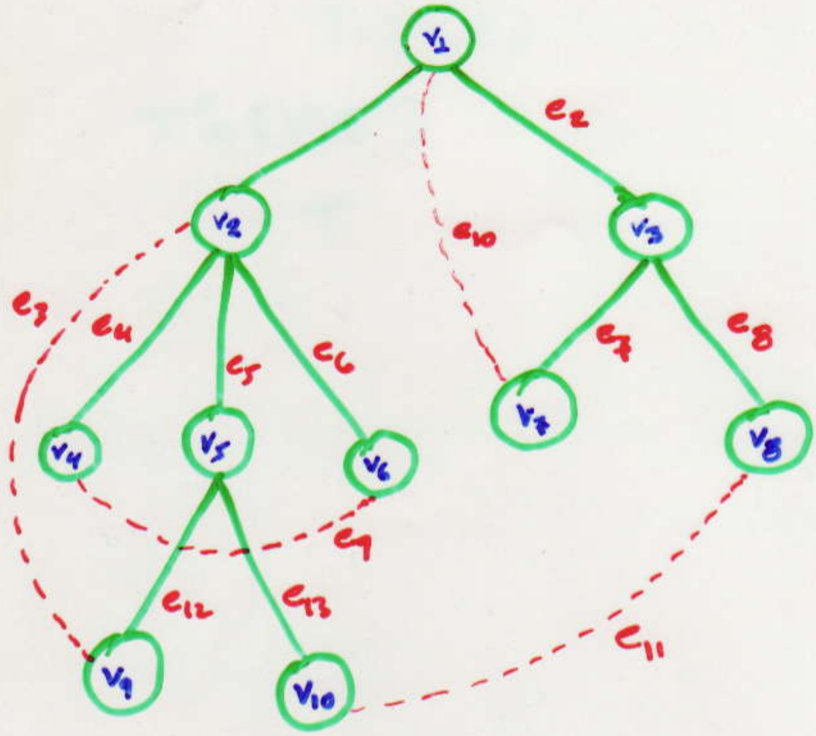
- $\text{level}(e)$ = level of the lowest common ancestor of u and v.

- $\text{label}(e) = (\text{level}(e), s(e))$

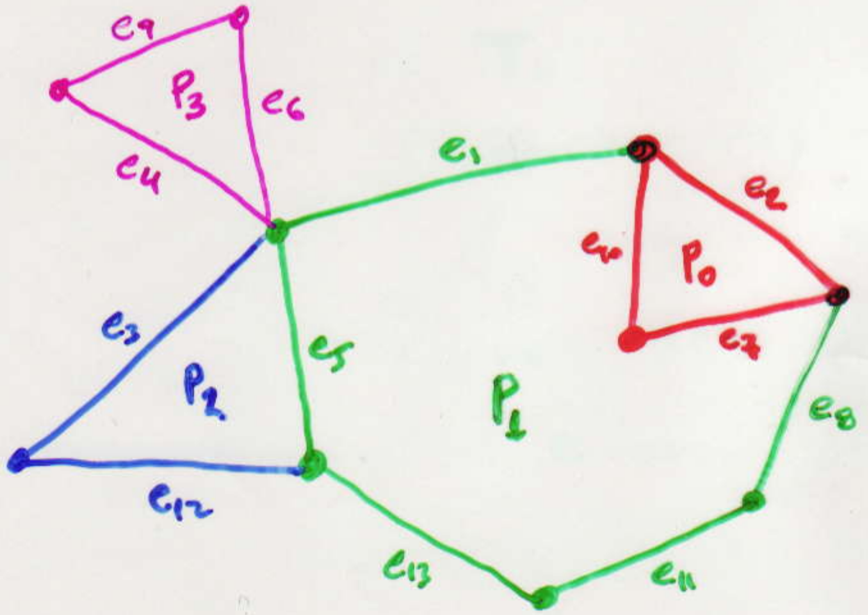
  where $s(e)$ is the serial number of e and $1 \leq s(e) \leq m$.

- For each tree edge g, let $\text{label}(g)$ be the smallest label of any nontree edge whose cycle contains g.

- Example



Edges	Label
e_1	(0,11)
e_2	(0,10)
e_3	(1,3)
e_4	(1,9)
e_5	(0,11)
e_6	(1,9)
e_7	(0,10)
e_8	(0,11)
e_9	(1,9)
e_{10}	(0,10)
e_{11}	(0,11)
e_{12}	(1,3)
e_{13}	(0,11)



- $P_e = \{e\} \cup \{g \in T \mid label(g) = label(e)\}$, where e a non tree edge.

(V) Euler-Tour Technique

- Let $T=(V,E)$ be a given tree and let $T'=(V,E')$ be the directed graph obtained from T when each $(u,v)\in E$ is replaced by two arcs $\langle u,v\rangle$ and $\langle v,u\rangle$.

- Since $\text{indegree}(u) = \text{outdegree}(u)$ $\forall u \in T' \Rightarrow T'$ is an Eulerian graph.

- Euler circuit of T' can be used for the optimal parallel computation of many functions of T.

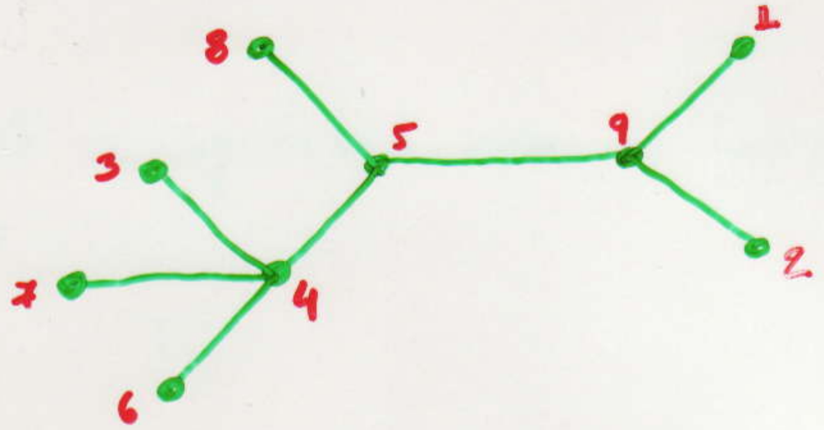
- Let $adj(v) = \langle u_0, u_1, \dots, u_{d-1} \rangle$

- We define the successor of each arc $e=\langle u_i, v\rangle$ as follows:

$$s(\langle u_i, v\rangle) = \langle v, u_{(i+1) \bmod d} \rangle$$

for $0 \le i \le d-1$.

- Example



v	$adj(v)$
1	9
2	9
3	4
4	5, 3, 7, 6
5	8, 4, 9
6	4
7	4
8	5
9	5, 2, 1

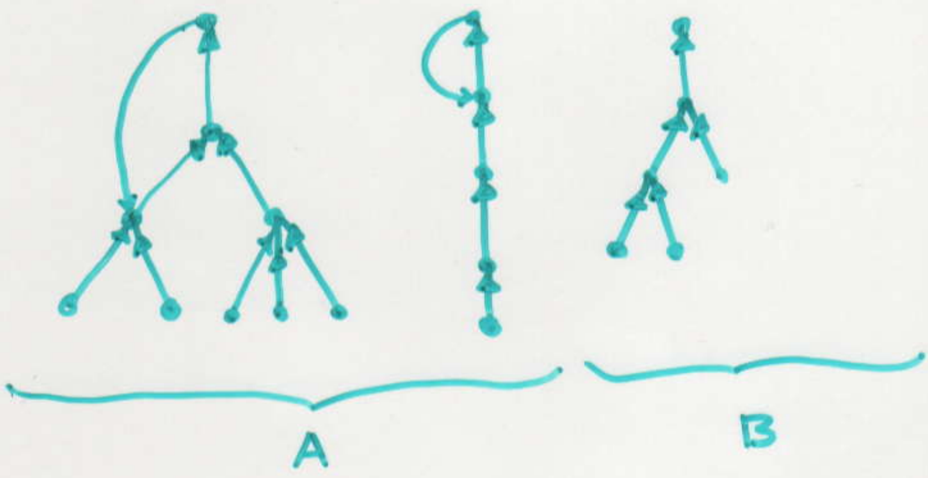
Arc	Succ
$\langle 9,1\rangle$	$\langle 1,9\rangle$
$\langle 9,2\rangle$	$\langle 2,9\rangle$
$\langle 4,3\rangle$	$\langle 3,4\rangle$
$\langle 5,4\rangle$	$\langle 4,3\rangle$
$\langle 3,4\rangle$	$\langle 4,7\rangle$
$\langle 7,4\rangle$	$\langle 4,6\rangle$
$\langle 6,4\rangle$	$\langle 4,5\rangle$
$\langle 8,5\rangle$	$\langle 5,4\rangle$
$\langle 4,5\rangle$	$\langle 5,9\rangle$
$\langle 9,5\rangle$	$\langle 5,8\rangle$
$\langle 4,6\rangle$	$\langle 6,4\rangle$
$\langle 4,7\rangle$	$\langle 7,4\rangle$
$\langle 5,8\rangle$	$\langle 8,5\rangle$
$\langle 5,9\rangle$	$\langle 9,2\rangle$
$\langle 2,9\rangle$	$\langle 9,1\rangle$
$\langle 1,9\rangle$	$\langle 9,5\rangle$

$\langle 9,1\rangle \to \langle 1,9\rangle \to \langle 9,5\rangle \to \langle 5,8\rangle \to$
$\langle 8,5\rangle \to \langle 5,4\rangle \to \langle 4,3\rangle \to \langle 3,4\rangle \to$
$\langle 4,7\rangle \to \langle 7,4\rangle \to \langle 4,6\rangle \to \langle 6,4\rangle \to$
$\langle 4,5\rangle \to \langle 5,9\rangle \to \langle 9,2\rangle \to \langle 2,9\rangle \to$
$\langle 9,1\rangle$

NC : Connected Components, Minimum Spanning trees, Biconnected components.

- Connected Components

- A pseudoforest is a directed graph in which each vertex has an outdegree less than or equal to 1.



- An arbitrary function $D: V \rightarrow V$ defines a pseudoforest (V, F), where $F = \{ \langle v, D(v) \rangle \mid v \in V \}$, but the converse is not necessarily true.

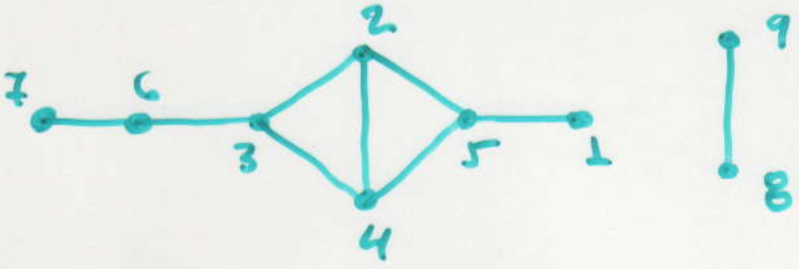
- Let A be the adj matrix of $G=(V,E)$

- $c: V \to V$: $c(v) = \min\{u \mid A(u,v)=1\}$, and

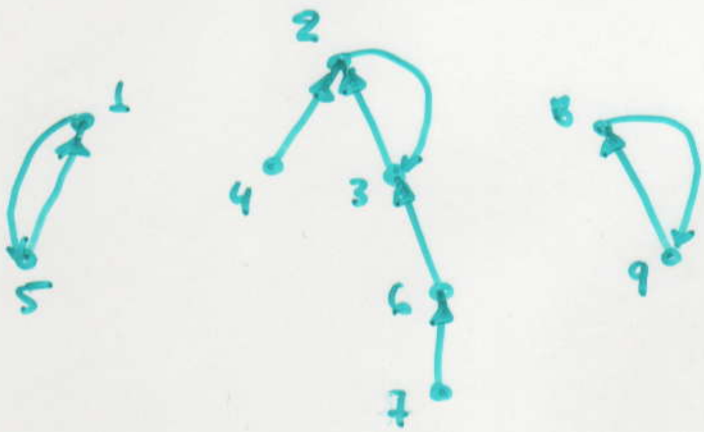
  if v is an isolated vertex, then $c(v) = v$.

- Lemma: Let $G=(V,E)$ be a graph and c be the function defined previously. Then, c defines a pseudoforest $\mathcal{F}$ that partitions $V = V_1 + V_2 + \cdots + V_s$ where V_i is the set of vertices in T_i of $\mathcal{F}$.
  The following claims hold:
  1. All vertices of V_i belong to the same con. comp.
  2. Each cycle in $\mathcal{F}$ either is a self-loop or contains exactly two arcs.
  3. The cycle of each tree T_i in $\mathcal{F}$ contains the smallest vertex in T_i.

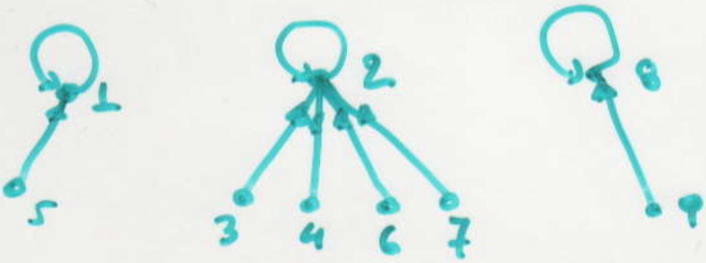
- $G=(V,E)$:

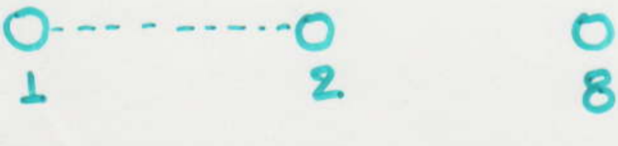


- The pseudoforest:



- The shrinking of the directed trees:



- 

(in 3)

- $O(\log^2 n)$
$O(n^2/\log^2 n)$

- Minimum Spanning Trees

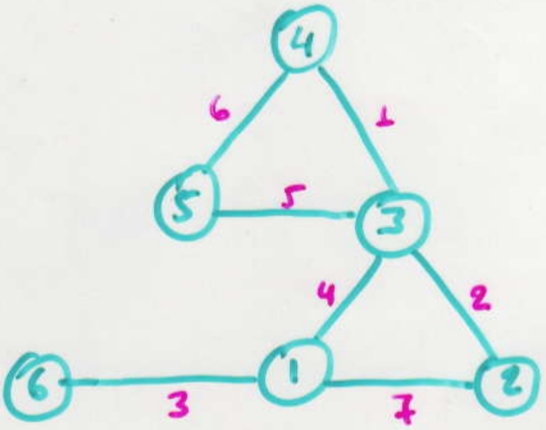
- Lemma: Let $G=(V,E)$ be a weighted graph. For each $u \in V$, let $G(u) \in V$ be such that $(u, G(u))$ is the minimum-weight edge incident on u. Then,

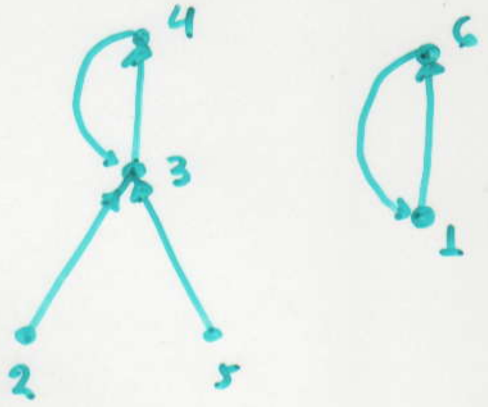
1. All the edges $(u, G(u))$ belong to the MST.

2. The function G defines a pseudoforest $\mathcal{F}$ such that each directed tree has a cycle containing exactly two arcs.

3. Let $V_1, V_2, \ldots, V_t$ be the vertex set of the trees $T_1, T_2, \ldots, T_t \in \mathcal{F}$.
   For each i, let e_i be the minimum-weight edge connecting a vertex in V_i to a vertex in $V - V_i$, $1 \le i \le t$.
   Then, all the edges e_i belongs to an MST of the graph G.

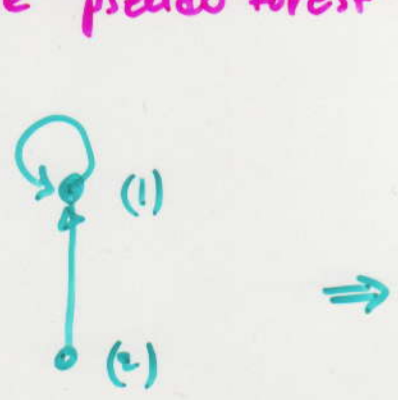
- The input graph $G=(V,E)$; and $\mathcal{F}$

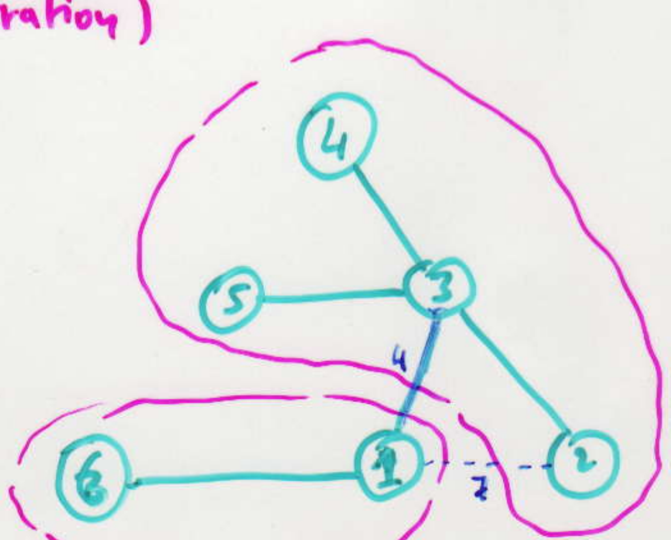




- The stars

- The pseudo forest (2ed iteration)



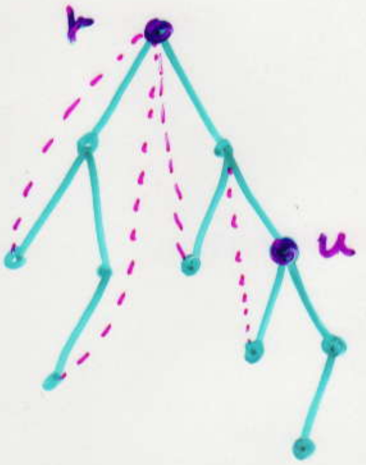


- $O(\log^2 n)$

  $O(n^2/\log^2 n)$

- Biconnected Components

- cutpoints



DFS tree of G

- root r
- internal node u

- Let $T = (V, E_T)$ be a span. tree of G.

- Each edge $e \in E - E_T$ creates a unique cycle C_e when added to T.

- The collection $\{ C_e \mid e \in E - E_T \}$ forms a cycle basis for G.

- Each C_e is called a basis cycle.

Paralle co-connecciuty Algorithm

- connected compounts

  - $O(\log^2 n)$ $\quad O\left(\frac{n^2}{\log^2 n}\right)$ CREW

  - $O(\log n)$ $\quad O\left(\frac{n^2}{\log n}\right)$ EREW

  - $O(\log n)$ $\quad O(n+m)$ CRCW

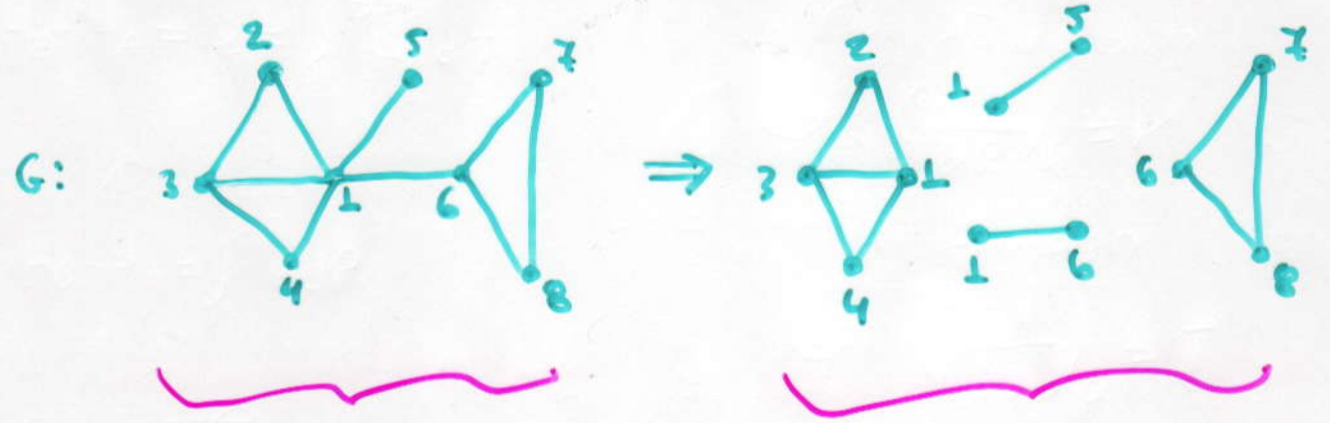
- co-connected componets

  - Naive algoritum : $G \Rightarrow \overline{G} = G'$

    connected componets in G'

  - поju проширу ?

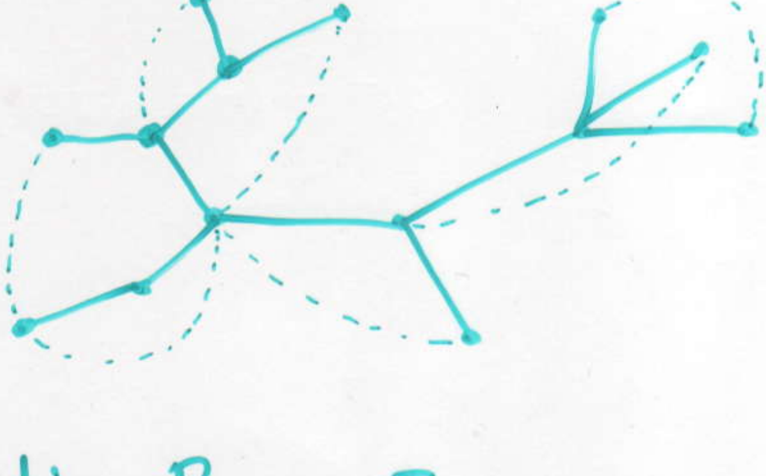
- Cutpoints - blocks



G

4 blocks of G

- Spanning tree: Tree and non-tree edges

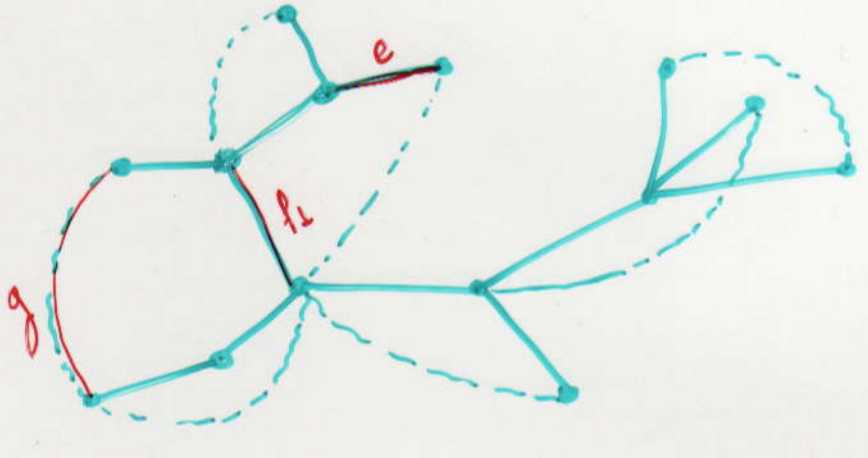


- Relation R_c on E

For any pair of edges (e,g)

$e \, R_c \, g \iff e, g$ belong to a common cycle determined by a nontree edge.

- The reflexive transitive closure R_c^* of R_c consists of all pairs of edges (e, g) for which either
  (i) $e = g$, or
  (ii) there exist edges $f_1, f_2, \dots, f_t$ such that $e\, R_c\, f_1$, $f_1\, R_c\, f_2$, $\dots$, $f_{t-1}\, R_c\, f_t$ and $f_t\, R_c\, g$.



- Lemma: Let T be a sp. tree of G, and let R_c be the relation defined prev.ly. Then, $R_b = R_c^*$, where R_b is the equivalence relation defining the blocks of G.

- equivalence $\equiv$ reflexive, symmetric and transitive.

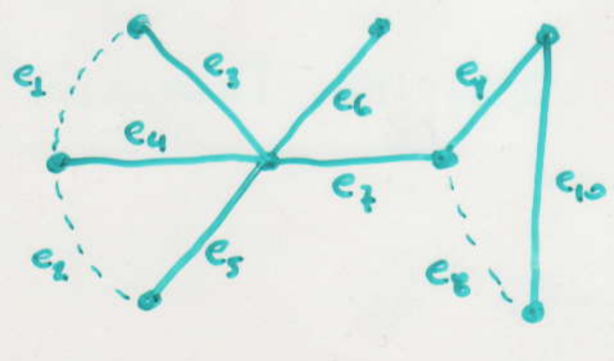
- The blocks of G can be determined as follows:

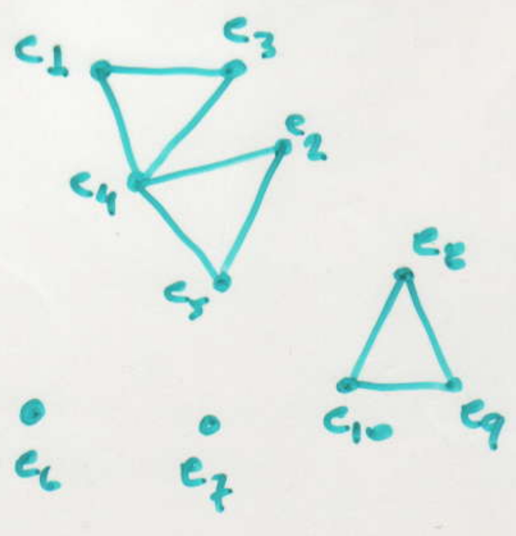
  - Let $G' = (V', E')$, where

    V' = the set of E of edges of G

    $(e,g) \in E' \iff e \, R_c \, g$.

  - The connected components of G' correspond to the equivalence classes of R_c^*, and hence uniquely identify the blocks of G.

- 

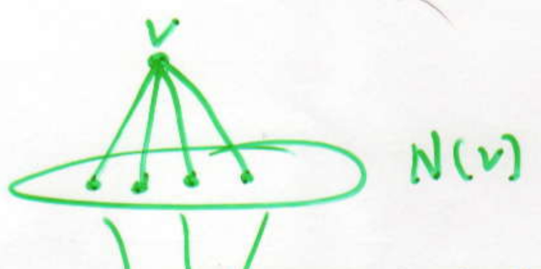


An Optimal Parallel Co-Connectivity Algorithm.

– connected components in $\overline{G}$;

co-connected comp.
or co-components

• Lemma 1: Let G an (n,m) graph.
If v is the vertex of min degree, then
$G[N(v)]$ has fewer than $\sqrt{2m}$ vertices.



Since $d_G(v)$ min $\Rightarrow$

$\sum_x d_G(x) \geq n \cdot d_G(v) \Rightarrow$

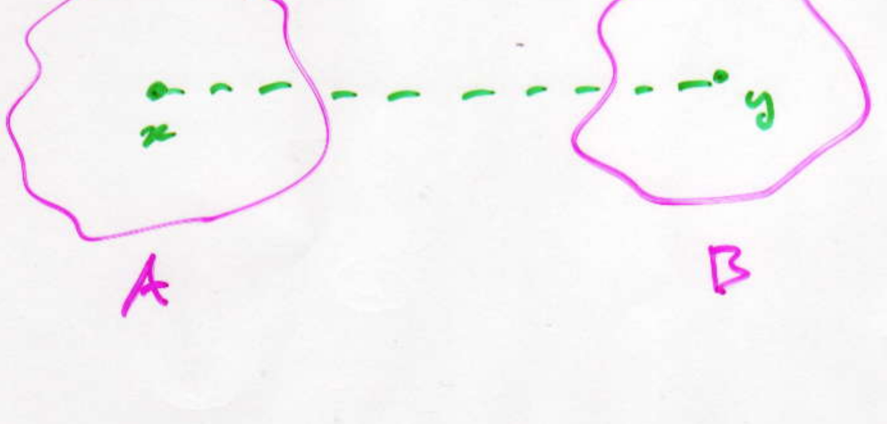
$d_G(v) \leq \dfrac{\sum d_G(x)}{n} = \dfrac{2m}{n}$ (1) $|N(v)| \leq \sqrt{2m}$

Since $m \leq \dfrac{n(n-1)}{2} < \dfrac{n^2}{2} \Rightarrow n > \sqrt{2m}$ (2)

(1) + (2) $\Rightarrow$ $\boxed{d_G(v) < \sqrt{2m}}$ ✓

- Lemma 2: Let $A \subseteq V(G)$ and $B \subseteq V(G)$ and $A \cap B = \emptyset$ (disjoint) and the vertices of A belong to the same co-component so do the vertices of B.

If the number of edges of G with one endpoint in A and the other in B is less than $|A| \cdot |B|$ then, the vertices of $A \cup B$ all belong to the same co-component.



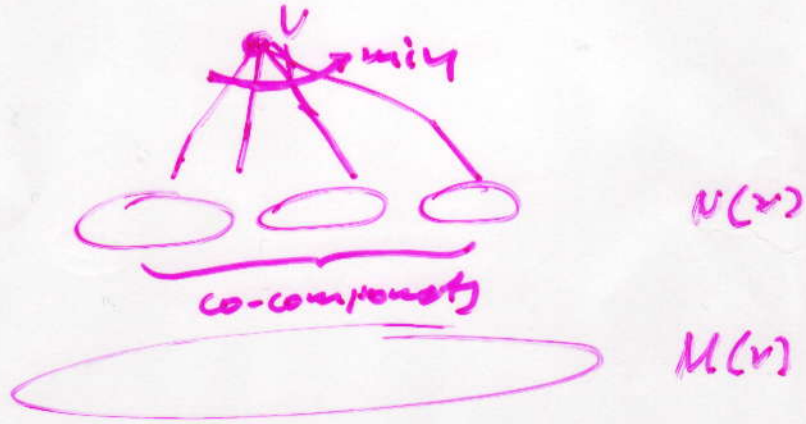
---- antiedge

- Algorithm Par_Co-components

Step 1: Comput v with min degree in G;

Step 2: If $m < n-1$ or $d_G(v) = 0$
then
for each $u \in V(G)$, do in parallel
co-comp[u] $\leftarrow v$;
Stop.

Step 3: Compute $\overline{G}[N(v)]$
Compute $d_{\overline{G}[N(v)]}(x)$, $\forall x \in \overline{G}[N(v)]$
Comput connected componts of $\overline{G}[N(v)]$



Step 4: For each $u \in N(v)$ in G, do in parallel

co-comp[u] $\leftarrow$ representative of the co-component of $G[N(v)]$ to which u belongs;

If $d_G[u] + d_{\bar{G}[N(v)]}(u) < n-1$

then

mark the representative of u;

Step 5: For each $u \in G$, do in parallel

If $uv \notin E(G)$, i.e. $u \in M(v)$

then

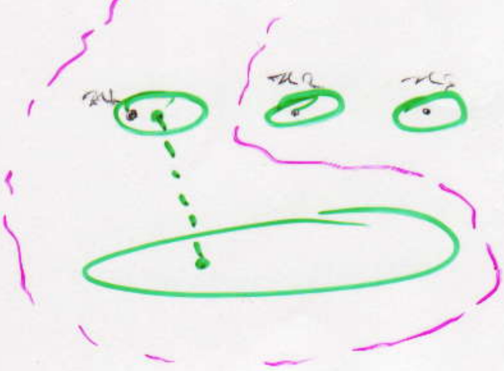
co-comp[u] $\leftarrow v$;

else $\{u \in N(v)\}$

if the representative of u is marked

then

co-comp[u] $\leftarrow v$;



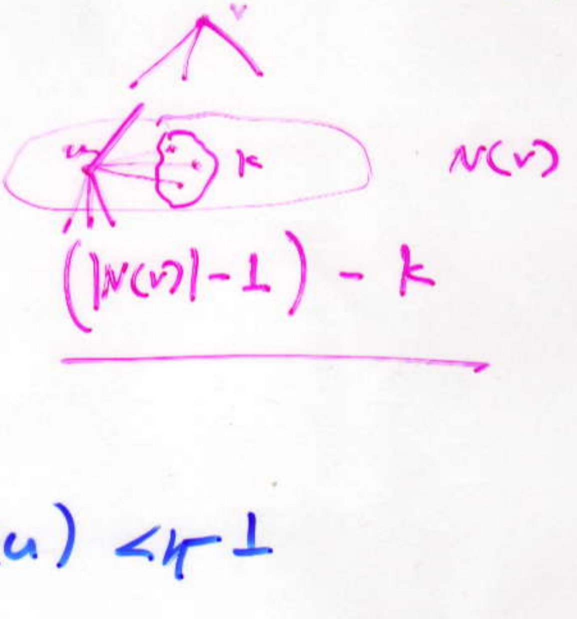
in the same co-component

Lemma 3: A vertex $u \in N(v)$ is not-adjacent to at least one vertex in $V(G) - N[v] = M(v)$ iff $d_G[u] + d_{\bar{G}[N(v)]}(u) < n-1$.

Proof: Let $k = \cancel{|N(v)|}$ number of neighbors of u which belong to $N(v)$ and $M(v)$ in G
$\ell = \cancel{|M(v)|}$

Then, clearly, $d_{\bar{G}[N(v)]}(u) = |N(v)| - k - 1$ (1)

$$d_G(u) = k + \ell + 1 \qquad (2)$$



Then, the condition

$$d_G(u) + d_{\bar{G}[N(v)]}(u) < n-1$$

is equivalent to

$$|N(v)| + \ell < n - 1 \Rightarrow$$

$$\boxed{\ell < n - 1 - |N(v)|}$$

The lemma follows, since: $n - 1 - |N(v)| = |M(v)|$