UE 7 : Mobility and Smart Cities

Philippe Canalda

Philippe.Canalda@univ-fcomte.fr

Remember …

- 2 main objectives, a first one which is real-time organization of mobility (optimization, prediction), a second one which is continuity of positionning and mobility (indoor <-> outdoor) if it remains enough time (now we know we do not have this time. However the MSC project has consisted in the co-specification and implementation of « a smart application of a indoor-outdoor instantaneous group's rendez-vous navigation system ».

Part 1, 22/11/2020, was :

- 1h of presentation of Louage problem : definition, specification and solving.
- 1h of course introducing orders and combinatorial algorithm by enumeration : greedy algorithm, enumeration with or without orders, enumeration with cut and price (when the solution computed will not improve the best results already computed), a recursive algoritm also known as dynamic programming algorithm with or without cut and price.
- 1h of practical work

Part 2, 02/12/2020, was :

- 1h of presentation of multimodal itinerary generation problem : definition, specification and solving.
- 1h of course introducing orders (partial, total), graph properties among symetry, transitivity.
- 1h of practical work to pursue exact solving by enumeration and price and cut and share + transitive closure
- 1h of MSC project (Ph Canalda part 2)

Part 3, 09/12/2020, was :

- 0.5h of directed work to assist the realisation of exercices given the previous 2 parts.
- 1h of practical work to pursue exact solving by enumeration and price and cut and share + transitive closure
- 1h of MSC project (Ph Canalda part 3)

Part 4, 16/12/2020, is :

- 1h of course pursuing introducing orders (partial, total), graph properties among symetry, transitivity.
- 1h of practical work to pursue exact solving by enumeration and price and cut and share + transitive closure
- 1h of MSC project (Ph Canalda part 2)

Part 5, 4/1/2021, is :

- 1h of course composed of :
  - a presentation of carpooling with transhipments finaly totally solved
  - the final part of course with specific elements in a – graph+sub-graph decomposition, graph properties, and the walk definition in depth and in width of a graph.
- 0h30 of practical work to evaluate the making changes algorithms + transitive closure and to finalize the exercices about special items
- 0h30 of MSC project (Ph Canalda part 4)

Dynamic Carpooling Intra-Modal with Transhipment(s).

- Illustrated and presented by Isham and Gautham

Subject of practical work n ° 1&2 (or homework 24/11/20)

- Prg1 = return the ‘making change’ greedy approach, taken from the course.
  The initialization of the data is carried out "hard", that is to say by using int L [7] = {5, 2, 1, 0.5, 0.2, 0.1, 0.05}
  The preferred language is JAVA. It could be C, or C ++.
  - » As support for your work, you must provide, even in a draft version: the class diagram, the factory fab (int) which updates the instantiation of the test, the compilation command, the command or the script execution (s), the results files, an analysis of these results and their reviews.
- Prg2 = exhaustive calculation of all solutions without recursivity. The calculation is long. Mostly printing to the screen or to a file. The solutions are displayed on the screen with an overload of the toString (...) function. The result can be redirected to a solutions_n ° test_output.txt file

Subject of practical work n ° 1&2 (or homework 24/11/20)

- Work to be done in practical work continued: part giving change continued
  - Prg3 = exhaustive calculation by recursion (also called dynamic programming algorithm) where all the solutions are generated in an in-depth walk. The solution of the current calculation is stored in an array of the same length as L and which contains the number of units of the available values. The combinations are displayed on the screen as they happen, when a solution is valid. This approach makes it possible to avoid the consumption of allocated memory (also known as RAM). However, I / O are special files called buffers which consume resources (slowness).

Suite TP1 - change part continued

- Continuation Prog 3: The data displayed on the screen will be redirected to a file AllTheSolutionsTest1.txt When this file is validated you suffix it with CORRECT_ You can restart the java execution Prg1> AllTheSolutionsTest1.txt You will compare the 2 files with the command : diff CORRECT_AllSolutionsTest1.txt AllSolutionsTest1.txt
- Prg3 bis: from prg 3 write prg3bis which reverses the order of the generation of solutions by favoring the maximum number of units of the value being processed.
- Prg4 = You calculate all the solutions that you do not display, as you go. If you store the solutions in a 2D array of solutions, in the order in which the valid combinations are found, then a subsequent display following that order can be compared with the contents of the CORRECT_ ... file and will show no difference.

Continuation of suiteTP1

- Prg5 = modify program 3 so as to evaluate the cost of a solution according to the number of units of the values. Calculate the best solution according to this cost by displaying successive values that improve. Display the trace of this execution in a file CORRECT_SolutionsWhichImproveIncrementally.txt
  Calculate the number of solutions displayed, and the display economy, using the wc –l command. Calculate with the time, or date command, the execution time programs 3 and 5.
- Prg6 = modify program 5 or program 4 so that the best calculated solution is stored in a solution table. Evaluate the computation time saved and the display savings.
- Prg7 = from program 6, implement the cut. To do this, the solution cost value being calculated must be calculated as you go. When the “partial” cost is greater than the best_cost, you stop recursive processing. This amounts to adding an end (if then else additional to the recursive processing).

Continuation of suiteTP1

- program 7 bis: How effective is the cut, and/or the gain, when the order implemented in prg3bis is implemented in prg 7?
- Program 8 that you could do but that I don't ask: is equivalent to calculating a sorted subset of the 5 best solutions.
- Program 8' which you could do but which I don't ask: is equivalent to calculating the best solution as well as the 2 less worst solutions. The first of the less worse solutions will have a better cost but for an amount to be made positive. The second less worse will have a better cost but for an amount to be made negative. Would you be able to realize this program !!!!!

– Relationship part

- The Boolean function is Transitive (int Relation (int [] [])) which evaluates whether a relation is transitive or not. The relation passed in parameter is the most simplistic with t [i] [j] taking the values -1 if no relation, 0 on the diagonal, and a positive value of cost to characterize the trsnision of the node i towards the node j.

Continuation of MSC project

– Project activity

- 2 functions to be developed here f1 and f2. the incomplete displacement graph can be obtained by interrogating a GIS. The matrix is somewhat sparse because the paths obtained by transitivity are not initialized => a saving of time, access to the GIS and therefore a saving of money when the number of calls is counted. An example of Graph g not complete thus :

$$g = \begin{matrix} 0 & 1 & 8 & -1 \\ -1 & 0 & 5 & 2 \\ 1 & -1 & 0 & -1 \\ 3 & 2 & -1 & 0 \end{matrix}$$

f1 consists in making the transitive closure. We traverse all the arcs of g: t [i] [j]. If t [i] [j]> 1 I go looking for all the t [j] [k]> 0. For such a couple t [i] [j] + t [j] [k] if it is > t [i] [k] existing, I replace this t [i] [k].

Continuation of MSC project

– f1 pursued

  - It is Dijkstra's algorithm that calculates the shortest paths from any point to any point.
    Warning ! There is a degree of difficulty between isTransitive and this Transitive closure. During a complete walk over the matrix roll, if a modification occurs, and therefore an addition of edge is issued from the transitivity computation, it is necessary to redo a complete walk, in search of an improvement of the transitive closure. We stop when there has been no improvement or modification during the complete walk over the matrix roll.

Continuation of MSC project

- f2 :
  - The f2 function uses a list of people positions. One then determines the point within the list of all the (discrete) positions of the problem by calculating the total value of all the displacements of the people to converge towards this point.

    We keep the point that minimizes the overall movements of the people of the group of students using the application.
    In order to, then, be able to generate the routes of each one, it is sufficient to modify the time matrix into a matrix of cells. If the cell is null, it matches -1. Otherwise, the cell will be made up of a cost in time, or in km, or a more complex table depending on the mode (bike, foot, car, etc.) and the time of day for taking congestion into account.

Continuation of MSC project

- f2 pursued:
  - … This cell is also made up of the route in text format which explains how to go from i to j. Going from i to k through j will correspond to the concatenation of the route t [i] [j] with that t [j] [k]. And that"s all folk!

    You have the innovative elements that many mobility solutions have not yet implemented;)

Subject of practical work n ° 4 (or homework 9/12/20)

- The challenge of the TD work to follow this presentation session will be :
  - for a given example with L={5, 2, 1, .5, .2, .1, .05} and m=12.35
  - greedy algorithm
    - » how much execution time ? How much print screen memory is used ?
  - exhaustive and enumerated version algorithm
    - » how much execution time ? How much print screen memory is used ? How much dynamic memory is used to keep in mind the best solution(s) ?
  - cut and price and share version
    - » how much execution time ? How much print screen memory is used ? How much dynamic memory is used to keep in mind the best solution(s) ? How much we have shared dynamic memory ? How much cuts are done ?

Exam question

- For a relation, a diagram or a graph, determining the properties of reflexivity, symmetry, transitivity, equivalence classes, this allows to reduce the starting data. It also helps to reduce the set of good solutions (targets). For each characteristic, it is necessary to be able to estimate the saving in quantity (size of the initial data, complexity saved by the algorithm, savings on the calculated solutions).

- For a given R relation, you need to formulate the new problems and evaluate the savings. We will have to deduct from these savings, possibly the additional calculation of the elements of an equivalence class, the calculation of the solutions resulting from a symmetry that we have taken into account to reduce the diagram of the starting data, etc.

The particular elements of an ordered set Example (3/3)

- Let X = {1, 2, 3, 5, 10, 20, 30}, be a set partially ordered by the relation x | y (x divides y).
  For the following two parts,
  - P_1 = {2,3,5,10} and P_2 = {2,5,10}
    - qualify them
      - the upper bound of P,
      - the lower bound of P,
      - the maximum elements
      - and the minimum elements,
      - the largest element of P,
      - the smallest element of P,
      - the u.b., the l.b.,
      - the universal element
      - and the null element.

- 1/ TD solve exercise slide 20,
- 2/ the complexity evaluation of making change
- 3/ adapt the transitive closure, or is-transitive or the best Mo-d which minimizes the path between 2 points of a graph,
  to
  remove the transitive edges of a relation that means that, for any t[x][z], having t[x][y] and t[y][z], remove t[x][z] from the relation

Part5

The particular elements of an ordered set Example (3/3) first result

- Let X = {1, 2, 3, 5, 10, 20, 30}, be a set partially ordered by the relation x | y (x divides y).
  For the following two parts,
  - P_1 = {2,3,5,10}
    - qualify them
      - the upper bound of P, {30}
      - the lower bound of P, {1}
      - the maximum elements, {3, 10}
      - and the minimum elements, {2, 3, 5}
      - the largest element of P, inexistant
      - the smallest element of P, inexistant
      - the u.b., {30}
      - the l.b., {1}
      - the universal element, inexistant
      - and the null element, {1}.

The particular elements of an ordered set Example (3/3) second result

- Let X = {1, 2, 3, 5, 10, 20, 30}, be a set partially ordered by the relation x | y (x divides y).
  For the following two parts,
  - P_2 = {2, 5,10}
    - qualify them
      - the upper bound of P, {10, 20, 30}
      - the lower bound of P, {1}
      - the maximum elements, {10}
      - and the minimum elements, {2, 5}
      - the largest element of P, {10}
      - the smallest element of P, inexistant
      - the u.b., {10}
      - the l.b., {1}
      - the universal element, inexistant
      - and the null element, {1}.

Use cases of these particular items 1/2

- The **a priori** knowledge of the data on which a combinatorial calculation (or optimizations) must be carried out.
  - case where the use of calculation clearly follows the availability of data over time ...
    - => so-called static optimizations where the result can be stored in the form of arrays, sagittal diagram / graph / automaton, Hass diagram, etc.
      This static part can sometimes correspond to the generation of data and / or code which, once compiled and added to a dynamic optimization kernel, will produce the expected results after compilation and then execution (see below).
  - ... and where sometimes the **a posteriori** use limits the scope of the combinatorial calculus on the initial data => so-called dynamic optimizations which will perhaps exploit the static processing prior to, and not necessarily exclusively,
      - Compilation, filtering, automata browsing or graph browsing or ...

Use cases of these particular items 2/2a

- On a given network or territory, we have the distances separating nodes 2 to 2.
  First, and for the sake of simplification, we will count the number of segments / edges separating different points / nodes (of a graph)

  G = (N = {0,1,2,3,4,5,6,7,8}, E = {(0.5), (0.7), (5.7), (5.3) , (3.6), (3.2), (7.3), (7.8), (7.1), (7.4), (4.8), (5.1)})

  We want to write a program which, given a subset of these nodes and edges, computes the list of nodes ordered in ascending order of distances to a given particular node.

  » For node 0 and the subset N0 = {1,3,4,5,7} and the associated edges E0 = {(0,5), (0,7), (5,7), (7, 3), (7,1), (7,4), (5,1)}
    and the expected result is L0 = {0,5,7,1,3,4}

Use cases of these particular items 2/2b

- G = (N = {0,1,2,3,4,5,6,7,8}, E = {(0.5), (0.7), (5.7), (5.3) , (3.6), (3.2), (7.3), (7.8), (7.1), (7.4), (4.8), (5.1)})
  - » For node 0 and the subset N0 = {1,3,4,5,7} and the associated edges E0 = {(0,5), (0,7), (5,7), (7, 3), (7,1), (7,4), (5,1)} and the expected result is L0 = {0,5,7,1,3,4}
  - This involves calculating the shortest paths from 0 to all points (by extension from any point to any point),
  - to arrange in L the nodes according to the increasing distance (but also in alphanumeric order in the event of equality _ because we have a partial order)
  - then during dynamic optimization, one have to browse the initial list L by showing the indexes of the nodes present in N0.

Use cases of these particular items 2/2c

- G = (N = {0,1,2,3,4,5,6,7,8}, E = {(0.5), (0.7), (5.7), (5.3) , (3.6), (3.2), (7.3), (7.8), (7.1), (7.4), (4.8), (5.1)})
      - » For node 0 and the subset N0 = {1,3,4,5,7} and the associated edges E0 = {(0,5), (0,7), (5,7), (7, 3), (7,1), (7,4), (5,1)} and the expected result is L0 = {0,5,7,1,3,4}
  - This involves calculating the shortest paths from 0 to all points (by extension from any point to any point),
  - to arrange in L the nodes according to the increasing distance (but also in alphanumeric order in the event of equality _ because we have a partial order)
  - then during dynamic optimization, one have to browse the initial list L by showing the indexes of the nodes present in N0.

Exemples d'usages de ces éléments particuliers 2/2 c

- It's about
  - calculate the shortest paths from 0 to all points (by extension from any point to any point),
  - to arrange in L the nodes according to the increasing distance (but also in alphanumeric order in the event of equality _ because we have a partial order)
  - then during dynamic optimization, it suffices to browse the initial list L by showing the indexes of the nodes present in N0.
- Give the matrix representation,
- Characterize the relationship (properties)
- Give the sagittal representation
- Does the representation of the Hasse diagram make sense in the present case and why
- Design the algorithm i) which calculates the shortest paths from any point to 0
- Extend this algo into an algo ii) which returns the list ordered totally according to the distances then, in case of equality, the list ordered according to alphanumeric order
- Design the algorithm iii) which calculates the sublist L0 corresponding to the subgraph G0 = (N0, EO) = G \ {2,4,6,8}
- Design the algorithm which extends i) to iv) which calculates the shortest paths from any point to any point
- Discussion VERY INTERESTING EXERCISE TO PERFORM

Trellises - definition 1/2

- When the part P of X is reduced to 2 elements ("pair") {x, y}, there may exist a ub. and / or a l.b.
  - The u.b. when it exists, denoted by $x \vee y$
  - The l.b. when it exists, denoted by $x \wedge y$

- Definition: We call a lattice a partially ordered set in which, for any pair of elements, there exists a u.b. and a l.b.

Trellises - definition 2/2

- Let T be a set, whose elements are endowed with 2 laws of composition, V (OR) and Λ (AND), verifying, whatever is x, y and z Є T, the properties:
  - 1'- x OR y = y OR x (commutativity) 1- x AND y = y AND x
  - 2'- x OR (y OR z) = (x OR y) OR z (associativity) 2- x AND (y AND z) = (x AND y) AND z
  - 3'- x OR x = x (idempotence) 3- x AND x = x
  - 4'- x OR (x AND y) = x (absorption) 4- x AND (x OR y) = x
- Then T constitutes a set ordered by the relation ≤ such that:
  - 5- [x ≤ y] ⇔ [x AND y] = x ⇔ [x OR y] = y
- T is a treillis

Trellises - example

- Consider the set T of the divisors of 30: 1, 2, 3, 5, 6, 10, 15, 30, ordered by the relation x | y.
- If we consider the associated Hasse diagram (and which you will construct),
- and if we consider that the u.b. of 2 elements is their scm and that the l.b. of 2 elements is their gcd,
- Then T, associated with l.b. and u.b. form a lattice

- The universal element is 30, the null element is 1.
- In general, N * ordered by '|' presents an infinite lattice structure whose the null element is always 1 and not having a universal element
- Develop the 2 Hass diagrams, the one for x | y, then the new relation

Trellises - other definitions 1/2

- A trellis T with zero element ***n*** and universal element ***U*** is complemented if, whatever x of T, there exists a possible association of an element x denoted by x-bar such that:
    - 6- x OR x-bar = U and 6'- x AND x-bar = n
- Exercise
    - check if the trellis of the previous example is complemented
    - Show that the system of axioms (1-4), (1’-4 ’) is not minimal, especially you will be able to show that absorption leads to idem-potence.
      Thus the system of axioms (1, 2, 4, 1 ’, 2’, 4 ’) is minimal or generator.

Trellises - other definitions 2/2

- A lattice is distributive if, to axioms 1 to 4 and 1 'to 4' are added, some x, y and z of T:
    - 7- x OR (y AND z) = (x OR y) AND (x OR z)

- Exercise
  - These axioms lead to :
    - 7'- x AND (y OR z) = (x AND y) OR (x AND z)
  - Conversely 7 'results in 7
  - In a distributive lattice the complementation is unique (demonstration by the absurd)

- A lattice that is both distributive and complemented is called a Boolean lattice. It is isomorphic to a Boolean algebra which is:
  - A partially ordered set, endowed with a null element and another universal, whose elements satisfy 1 to 4, 1 'to 4', 6, 6 ', 7
    The order relation is defined by one of the following 4 relations :
    - 5*- [x ≤ y] ⇔ [x AND y] = x ⇔ [x OR y] = y ⇔ x AND y-bar = 0 ⇔ x OR y-bar =1, with n=0 and U=1

A Boolean algebra

- A Boolean algebra with one generator, other than 0 and 1, has 4 elements. (Hass diagram)
- With 2 elements, it has 16 elements.

- Stone's theorem: Any Boolean algebra is isomorphic to a set field.

- Example: from Boolean algebra with 2 generators a and b such that a! = B and a AND b! = 0,
  we make it appear more clearly ...

Body of a set (corps d'ensemble)

- Example: from Boolean algebra with 2 generators a and b such that a! = B and a AND b! = 0,
  we make it appear more clearly…

- In a parallelepipedal shape (Euler-Venn diagram) representing universe 1: A, B, A ∩ B, A ∩ B-bar, A-bar ∩ B, A-bar ∩ B-bar which are "regions" corresponding to the respective atoms a, b, a AND b, a OR b-bar, a-bar AND b, a-bar AND b-bar

- We call a set body a family of parts of a set, endowed with classical set operations
  (union, intersection, complementation).

Reminder on graphs

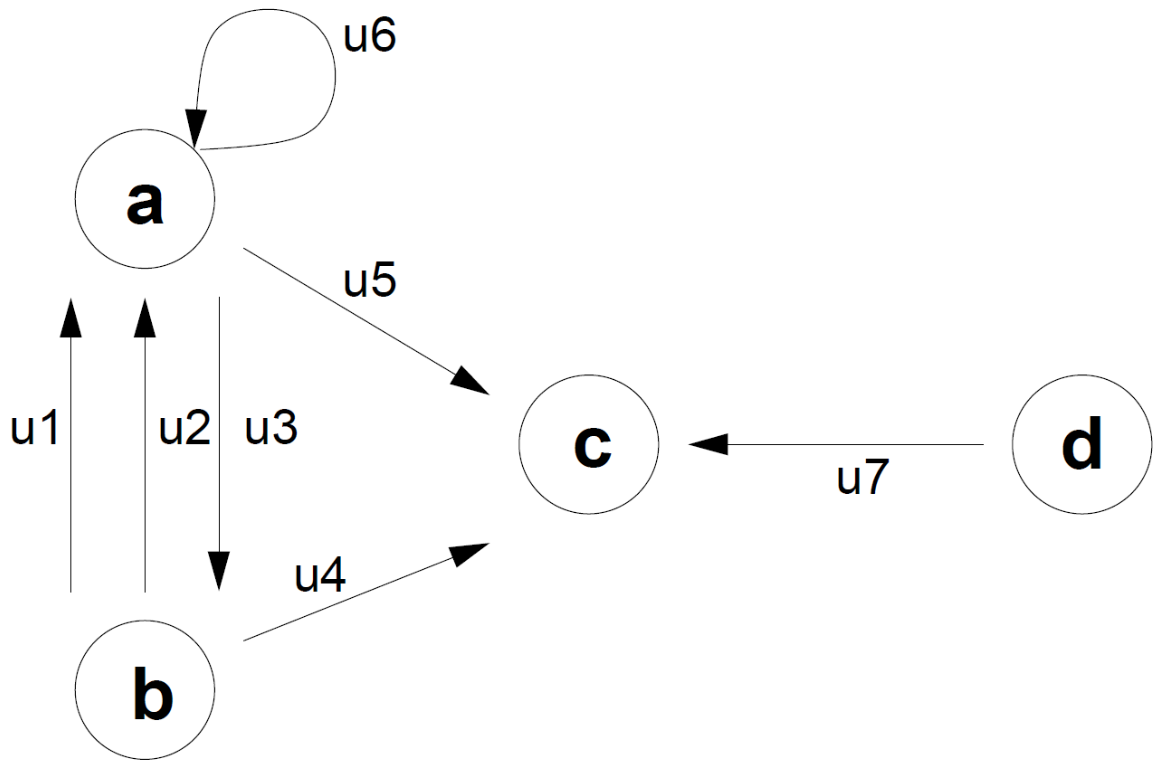
- Oriented graph G
  - A finite set V of vertices x_1, … x_n
  - A family E of items from the cartesian product N x N, called edges

    an edge can appear several time

- An oriented edge (x,y) and a non oriented one (symmetrical) xy

slidesGraphes_in_french_PhC-part5.pdf	• Éléments de graphes en français téléchargeable depuis discord
MIT6_042JF10_chap05-part5.pdf	• Long version english support established by MIT

Reminder on graphs

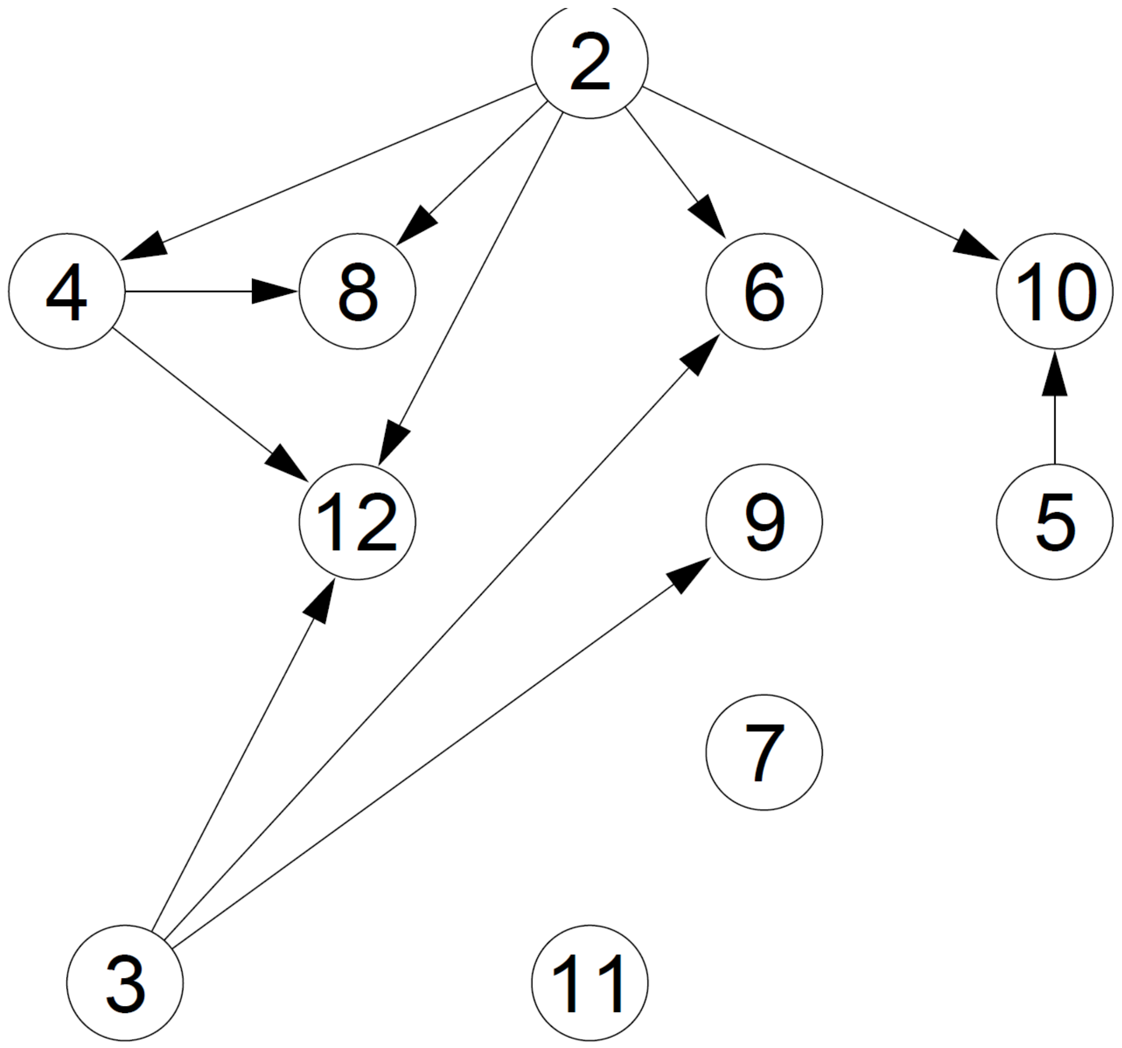


- A 2-graph, a p-graph

Reminder on graphs

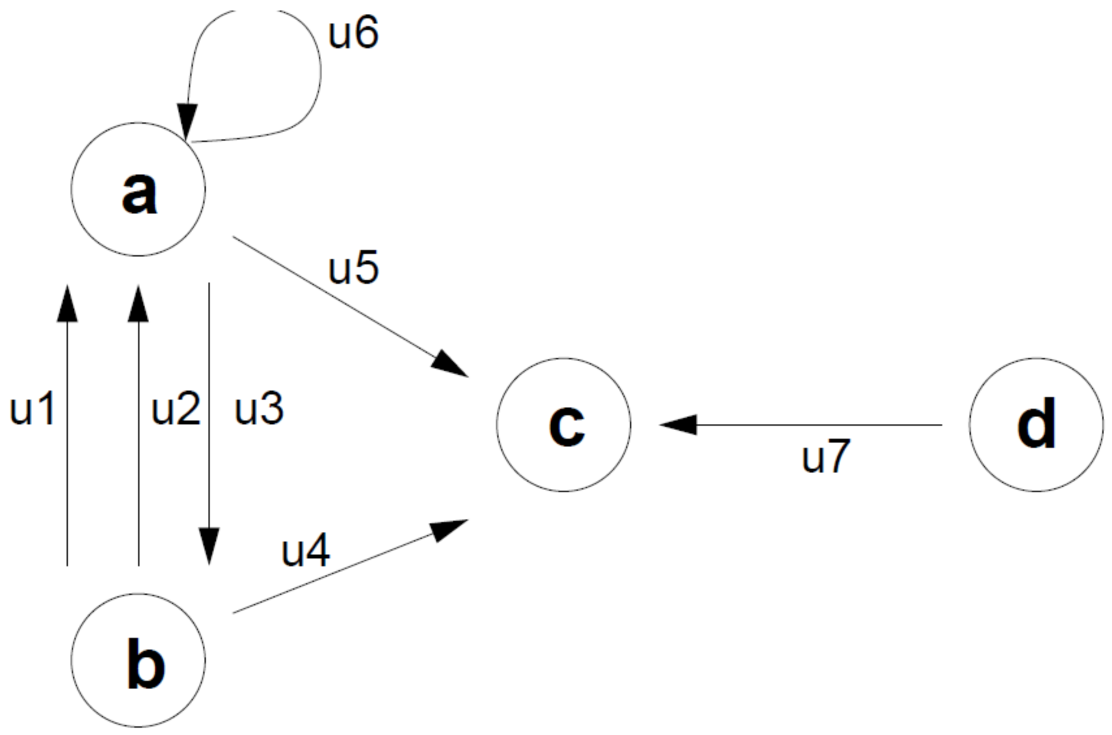
- Relation between nodes
  - Symmetrical, not symmetrical, transitive, etc.

Reminder on graphs



An example :
the relationship x divides exactly y

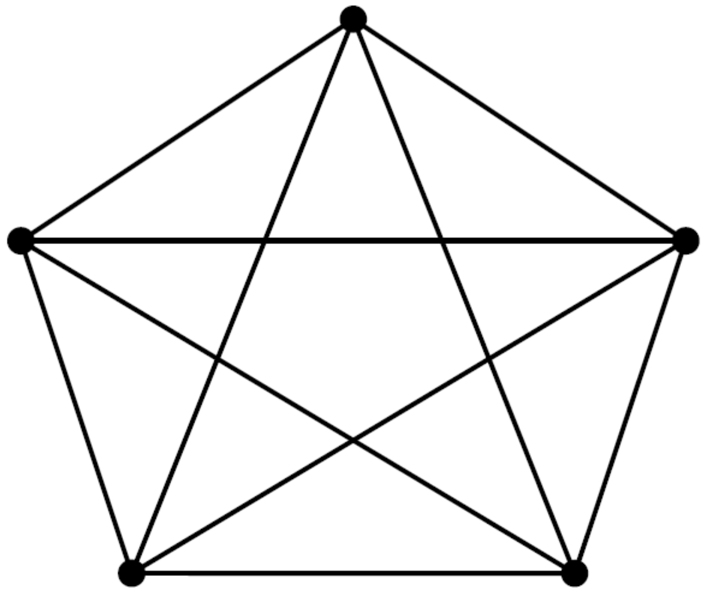
Reminder on graphs



- saggital diagram and weighted graph
- Incident edge, joining a vertex / node
- Self-loop and Loop
- Degree, subgraphs, adjacency matrix,

Reminder on graphs

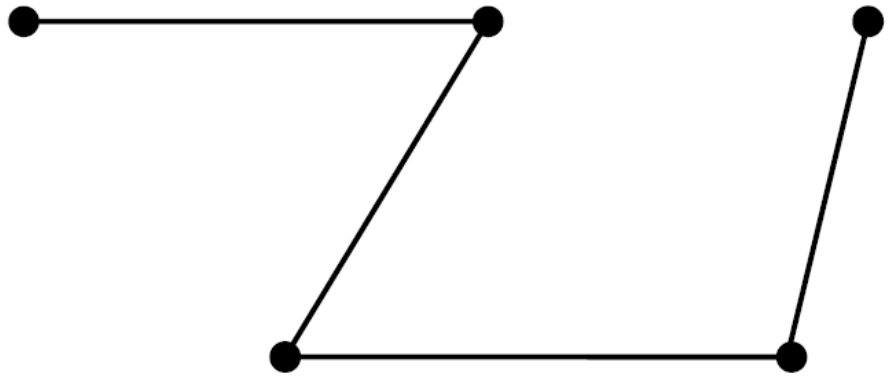
- Common graphes, and simple one’s



- The complete 1-graph on n vertices, denoted K_n has an edge between every 2 vertices. n (n-1) / E edges

Reminder on graphs

- Common graph is for example, the 5-node line graph L5



- An 1-graph with n-1 edges in sequence
- Complete graph
- Hypercube Hn
- Cube Connected Cycle CCCn
- Grid, toric grid, buterfly graph, shuffle exhange, star fly, …

Reminder on graphs

- Simplest chain, shortest path, a loop, a graph without loop, a directed acyclic graph (DAG), incident edge(s), neighbor(s), adjacent vertex/node(s), degree (od a node, of a graph), density of a graph, …

- Particular graphs :
  - regular graph,

    $$\delta(G) = \Delta(G)$$

  - irregular's one,
  - Symmetrical graph, anti-symetrical one
  - Complementary graph
  - Inverse graph
- Diameter and distances : d(x,y), d(xy,uv), d(L1,L2), …

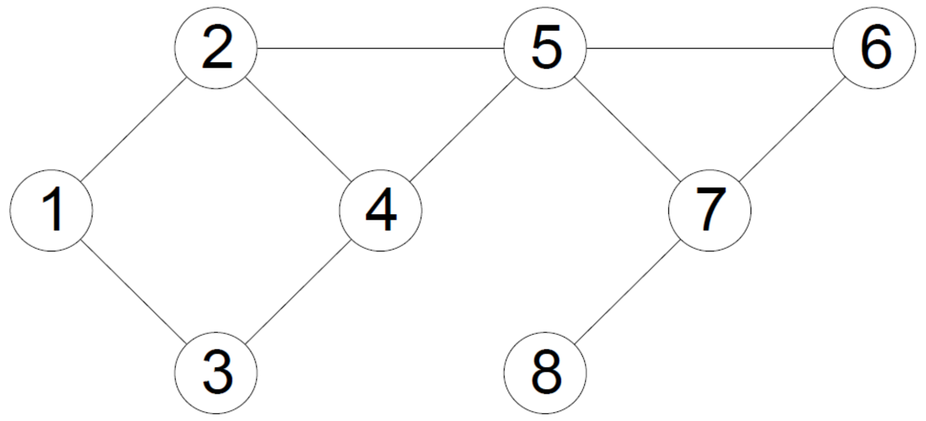
Reminder on graphs

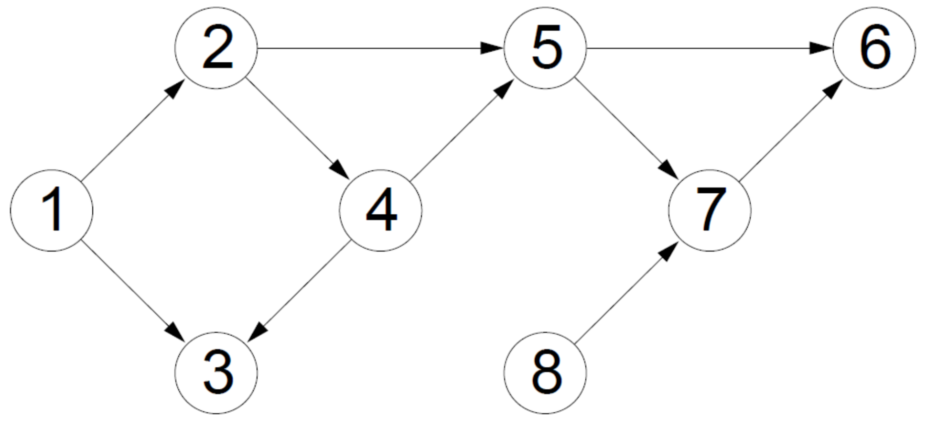
- Operations on graphs:
  - union
  - intersection
- Complete graph, isomorphism and bipartite graph
- Searching a complete sub-graph in a graph is a NP-complete problem

Reminder on graphs

- Path, elementary path, circuit,

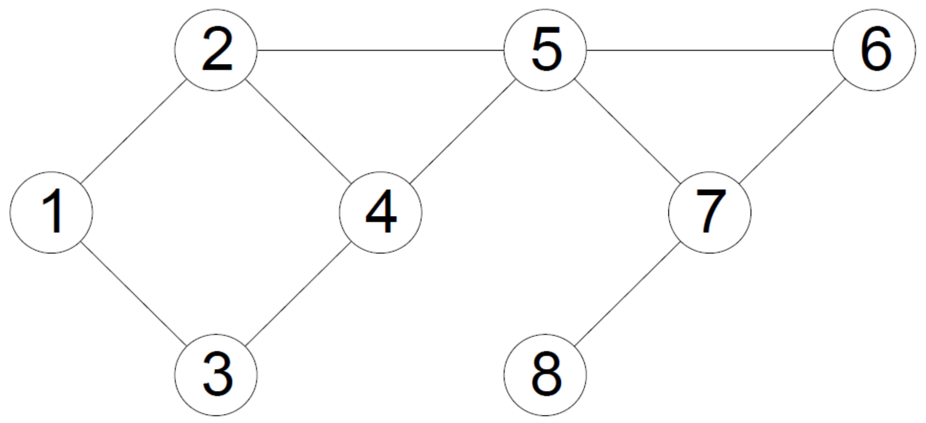
Generic browse : border

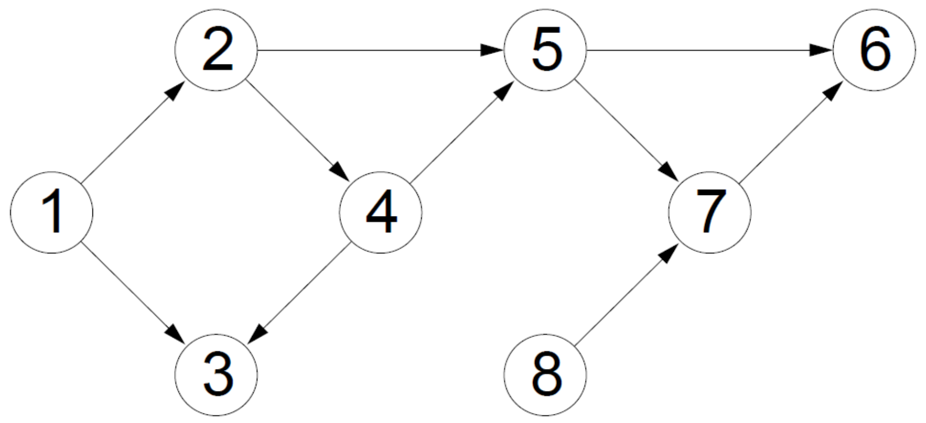




- G(V,E) with |V|=n and |E|=m, and a set T $\subset$ V
- Border of T, B(T,G), is a set of node(s) of V-T, node which is at least a neighbour of one node of T

Generic browse : permutation



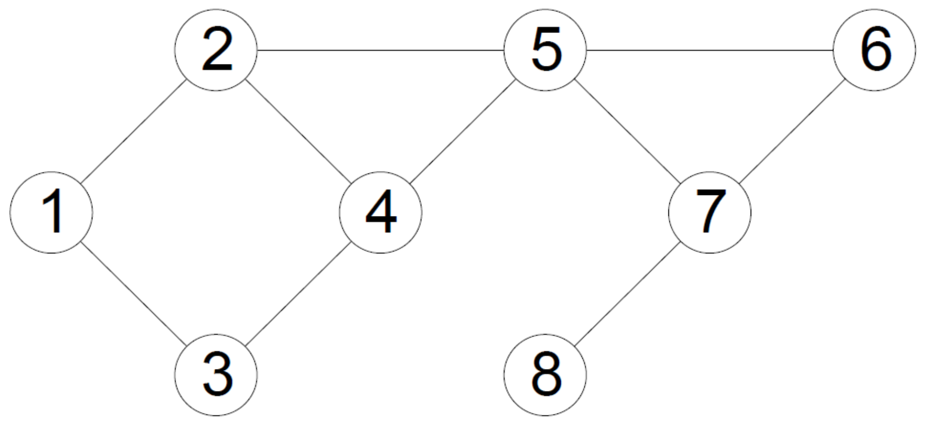


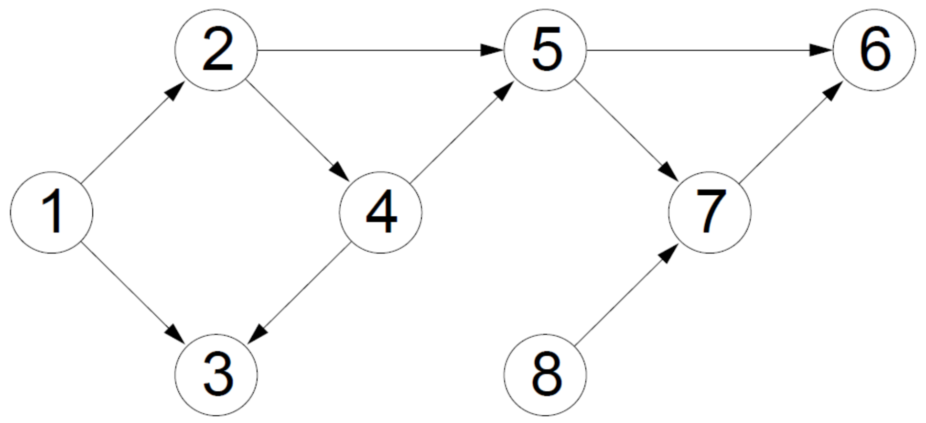
- A permutation L=(S1, S2 .., Sn) of nodes from a graph G a generic browse from the node i if and only if:

$$\forall j \in \{2, \ldots, n\},\ B(L[1, j-1], G) \neq \emptyset \Rightarrow s_j \in B(L[1, j-1], G)$$

with L[1,j-1] a sub-list of nodes S_1, .. S_{j-1}

Generic browse : permutation





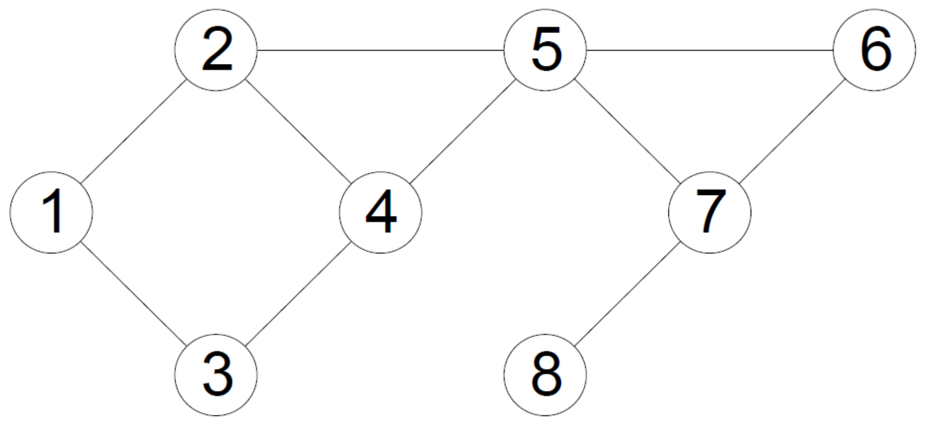
- A permutation L=(S1, S2 .., Sn) of nodes from a graph G a generic browse from the node i if and only if:

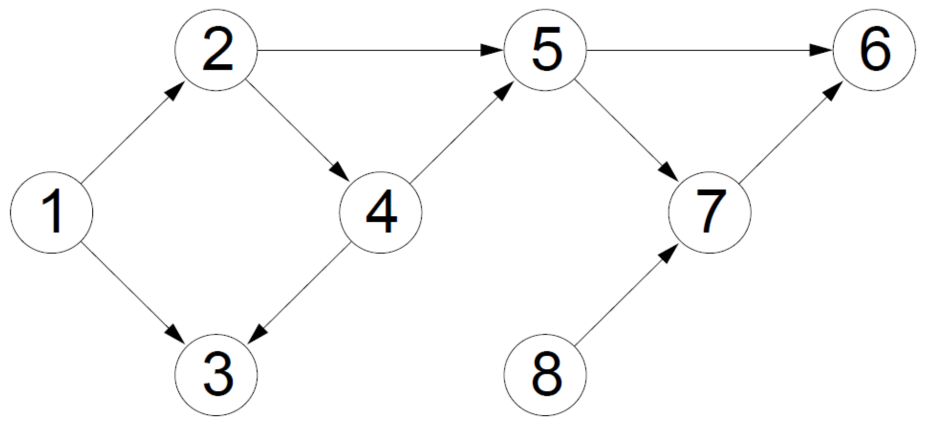
$$\forall j \in \{2, \ldots, n\},\ B(L[1, j-1], G) \neq \emptyset \Rightarrow s_j \in B(L[1, j-1], G)$$

with L[1,j-1] a sub-list of nodes S_1, .. S_{j-1}

? L1 = (1, 3, 4, 5, 7, 8, 6) ?

Generic browse : permutation





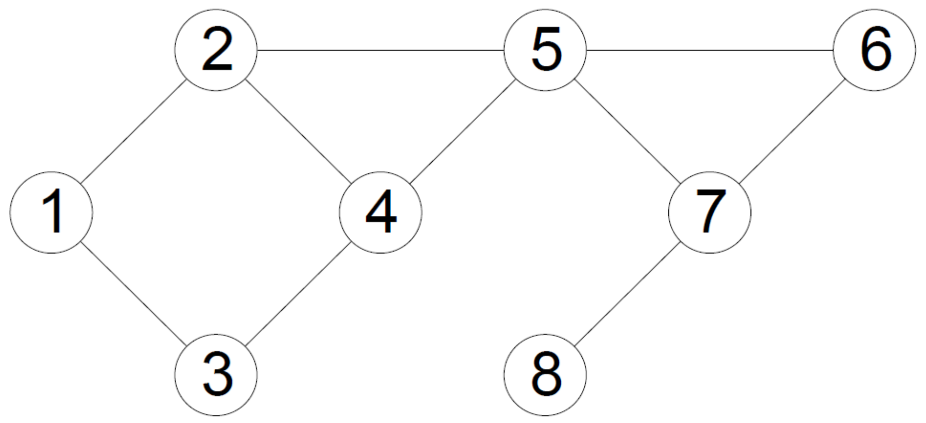
- A permutation L=(S1, S2 .., Sn) of nodes from a graph G a generic browse from the node i if and only if:

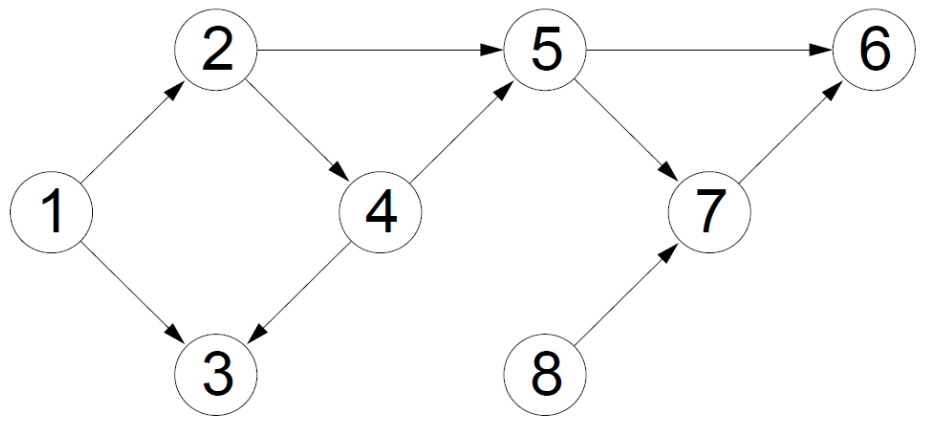
  $$\forall j \in \{2, \ldots, n\},\ B(L[1, j-1], G) \neq \emptyset \Rightarrow s_j \in B(L[1, j-1], G)$$

  with L[1,j-1] a sub-list of nodes S_1, .. S_{j-1}

  ? L1 = (1, 3, 4, 5, 7, 8, 6) ? Is a possible browse

Generic browse : permutation





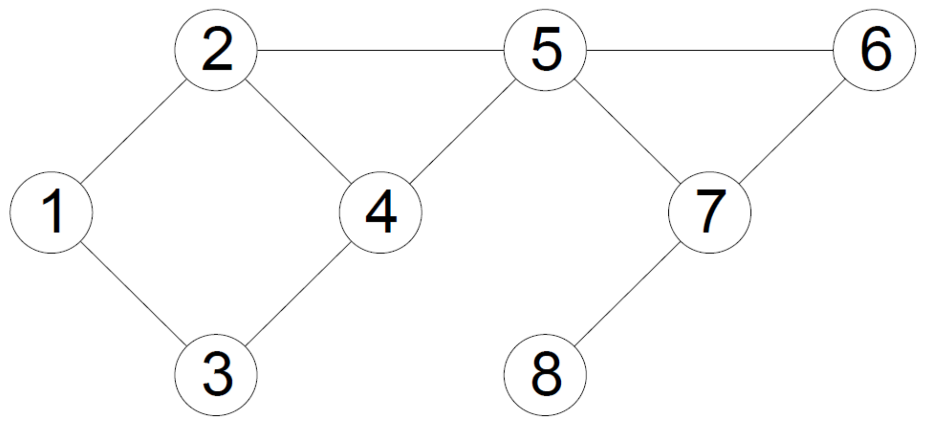
- A permutation L=(S1, S2 .., Sn) of nodes from a graph G a generic browse from the node i if and only if:

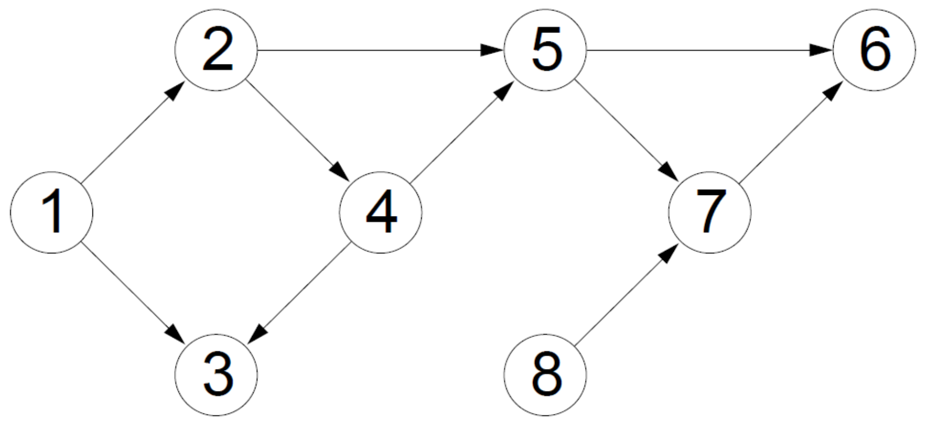
  $$\forall j \in \{2, \ldots, n\},\ B(L[1, j-1], G) \neq \emptyset \Rightarrow s_j \in B(L[1, j-1], G)$$

  with L[1,j-1] a sub-list of nodes S_1, .. S_{j-1}

  ? L2 = (5, 6, 2, 7, 8, 1, 3, 4) ?

Generic browse : permutation





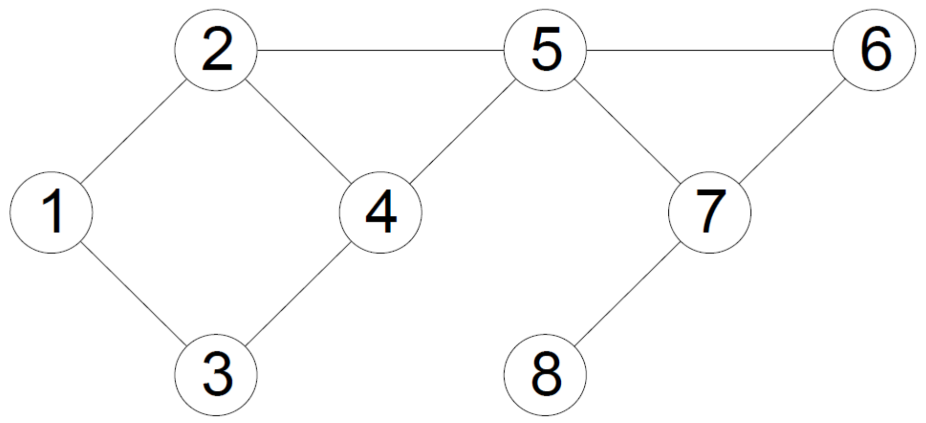
- A permutation L=(S1, S2 .., Sn) of nodes from a graph G a generic browse from the node i if and only if:

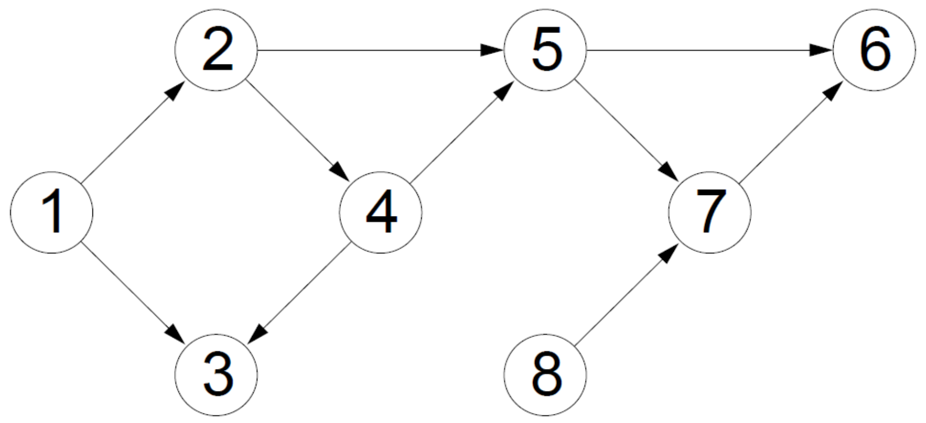
  $$\forall j \in \{2, \ldots, n\},\ B(L[1, j-1], G) \neq \emptyset \Rightarrow s_j \in B(L[1, j-1], G)$$

  with L[1,j-1] a sub-list of nodes S_1, .. S_{j-1}

  ? L2 = (5, 6, 2, 7, 8, 1, 3, 4) ? Is a possible browse

Generic browse : permutation





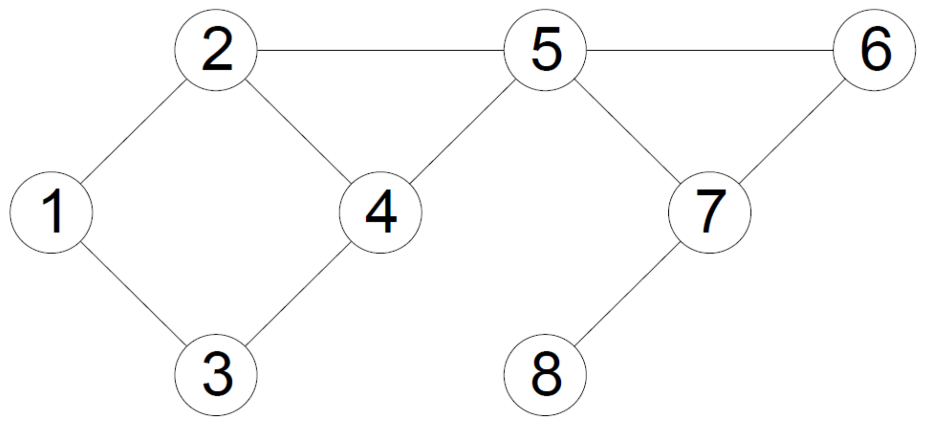
- A permutation L=(S1, S2 .., Sn) of nodes from a graph G a generic browse from the node i if and only if:

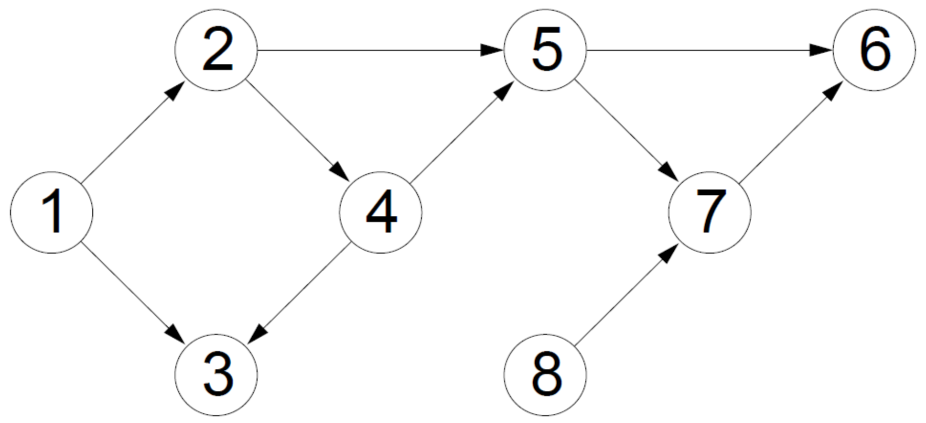
$$\forall j \in \{2,\ldots,n\},\ B(L[1,j-1],G) \neq \emptyset \Rightarrow s_j \in B(L[1,j-1],G)$$

with L[1,j-1] a sub-list of nodes S_1, .. S_{j-1}

? L3 = (5, 6, 3, 2, 1, 4, 8, 7) ?

Generic browse : permutation





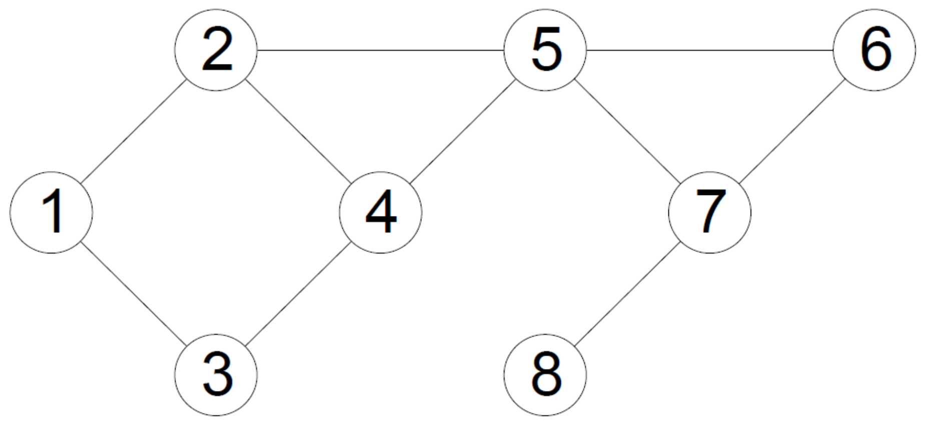
- A permutation L=(S1, S2 .., Sn) of nodes from a graph G a generic browse from the node i if and only if:

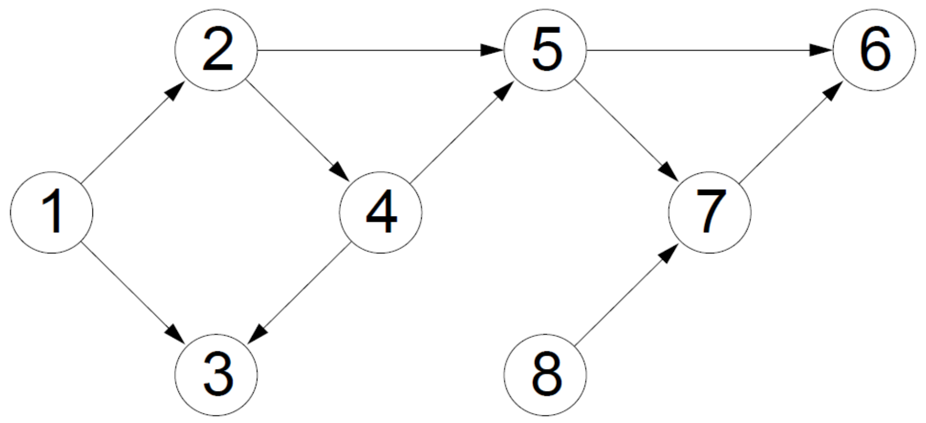
$$\forall j \in \{2, \ldots, n\},\ B(L[1, j-1], G) \neq \emptyset \Rightarrow s_j \in B(L[1, j-1], G)$$

with L[1,j-1] a sub-list of nodes S_1, .. S_{j-1}

? L3 = (5, 6, 3, 2, 1, 4, 8, 7) ? No $s_3 \notin B(L_3[1, 2])$.

Generic browse : regeneration point





- Given L of G, a regeneration point of L is a node Sk such that k=1
  or B (L[1, k-1],G) = ∅ with k > 1

  these points are initiators, or null points, helpfull in multiple wave browse of a graph.

Choice graph, covering forest

- Given L of G, foreach node Sj not a regeneration node, we bind it to one (Si,Sj) of G such that $s_i \in L[1, j-1]$.

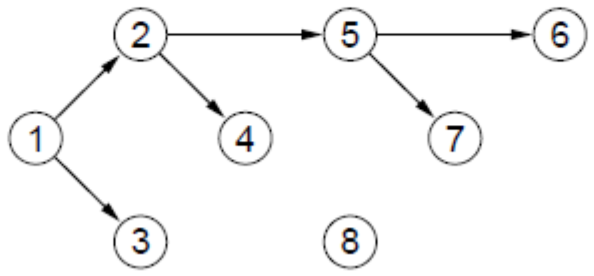
  The obtained graph G’ is called choice graph or covering forest

Choice graph, covering forest

- Given L of G, foreach node Sj not a regeneration node, we bind it to one (Si,Sj) of G such that $s_i \in L[1, j-1]$.

  The obtained graph G' is called choice graph or covering forest

  L4=(1, 2, 3, 4, 5, 6, 7, 8) is a generic browse of G



is one choice graph of G and it is not unic ((4,5) instead of (2,5))

The node 8 is a regeneration node

Opened node, closed node

- A node x of a permutation L, which is a generic browse L[i,1] = (s1, .. Si), is opened in L[1, i] for G if it accepts at least one neighbour in G in L[i+1, n]. Otherwise, this node is closed in L[1,i] for G.

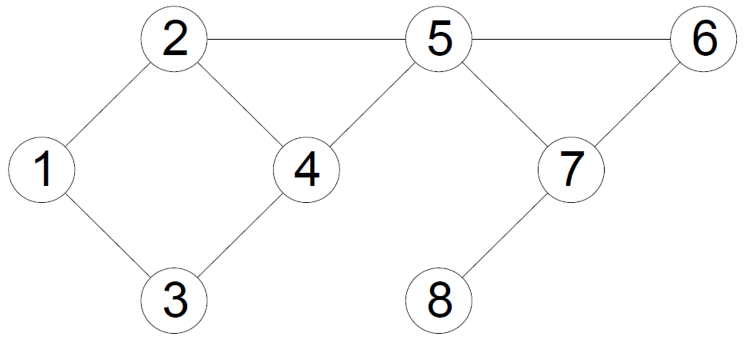
In width walk/browse

- A generic browse L = (s1, …, Sn) is in width
  if
  foreach j in {2, …, n} such that B(L[1, j-1],G)≠∅
  the node sj is a neighbour of the first opened node for L[1, j-1] in G.

In width walk/browse

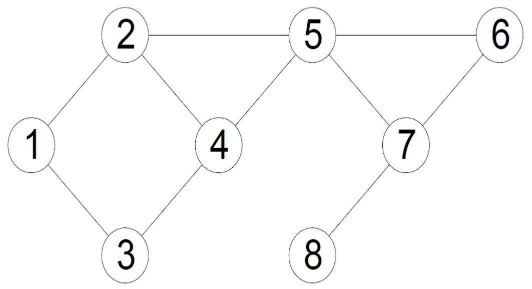
- A generic browse L = (s1, …, Sn) is in width
  if
  foreach j in {2, …, n} such that B(L[1, j-1],G)≠∅
  the node sj is a neighbour of the first opened node
  for L[1, j-1] in G.

  ? L5 = (1, 2, 3, 4, 5, 6, 7, 8)
  and L6 = (6, 7, 5, 8, 4, 2, 3, 1)
  and L6bis = (6, 5, 7, 8, 4, 2, 3, 1) ?



In width walk/browse

- A generic browse L = (s1, …, Sn) is in width
  if
  foreach j in {2, …, n} such that B(L[1, j-1],G)≠∅
  the node sj is a neighbour of the first opened node
  for L[1, j-1] in G.



  ? L5 = (1, 2, 3, 4, 5, 6, 7, 8) in with
  and L6 = (6, 7, 5, 8, 4, 2, 3, 1) in width
  and L6bis = (6, 5, 7, 8, 4, 2, 3, 1) ? not in width
  because s4 = 8 is not adjacent to the first opened
  node for L[1, 3] in G (node 5)

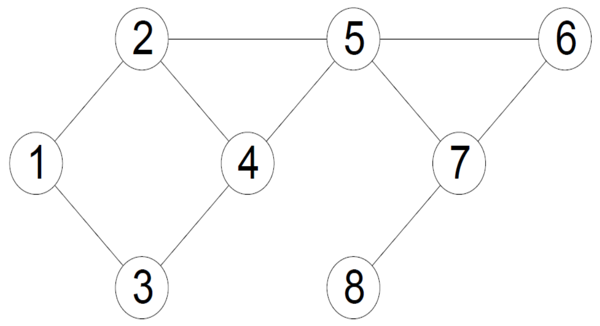
In depth walk/browse

- A generic browse L = (s1, …, Sn) is in depth
  if
  foreach j in {2, …, n} such that B(L[1, j-1],G)≠∅
  the node sj is a neighbour of the last opened node for L[1, j-1] in G.

In depth walk/browse

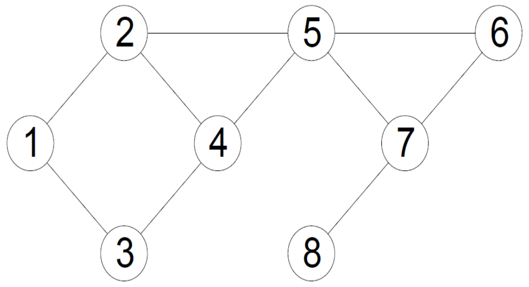
- A generic browse L = (s1, …, Sn) is in depth
  if
  foreach j in {2, …, n} such that B(L[1, j-1],G)≠∅
  the node sj is a neighbour of the last opened node for L[1, j-1] in G.

  ? L7 = (2, 5, 4, 3, 1, 6, 7, 8)
  and L8 = (5, 7, 8, 2, 4, 1, 3, 6) ?



In depth walk/browse

- A generic browse L = (s1, …, Sn) is in depth
  if
  foreach j in {2, …, n} such that B(L[1, j-1],G)≠∅
  the node sj is a neighbour of the last opened node
  for L[1, j-1] in G.



  ? L7 = (2, 5, 4, 3, 1, 6, 7, 8) in depth
    and L8 = (5, 7, 8, 2, 4, 1, 3, 6) not in depth
    because s4 = 2 is not adjacent to the last opened
    node for L[1, 3] in G (node 7)

Reminder on graphs

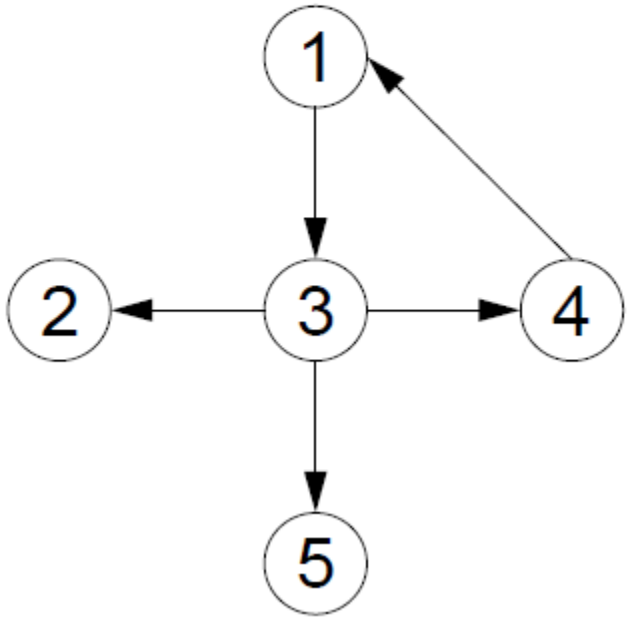
- Transitive closure

$$\Gamma^{+}(x) = \{x\} \cup \Gamma(x) \cup \Gamma(\Gamma(x)) \ldots$$

Reminder on graphs

- Transitive closure

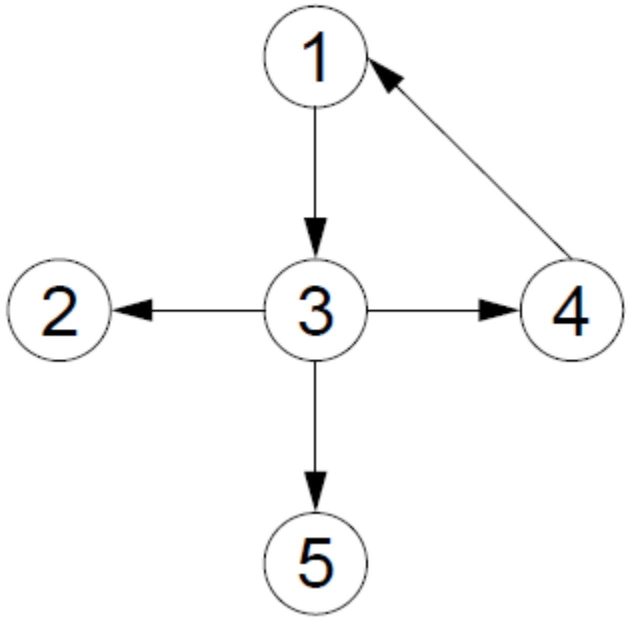
$$\Gamma^+(x) = \{x\} \cup \Gamma(x) \cup \Gamma(\Gamma(x)) \ldots$$

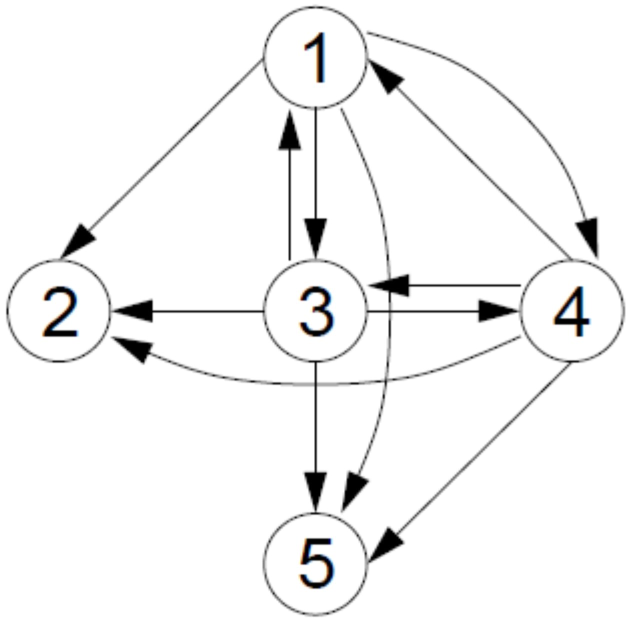


Reminder on graphs

- Transitive closure

$$\Gamma^{+}(x) = \{x\} \cup \Gamma(x) \cup \Gamma(\Gamma(x)) \ldots$$





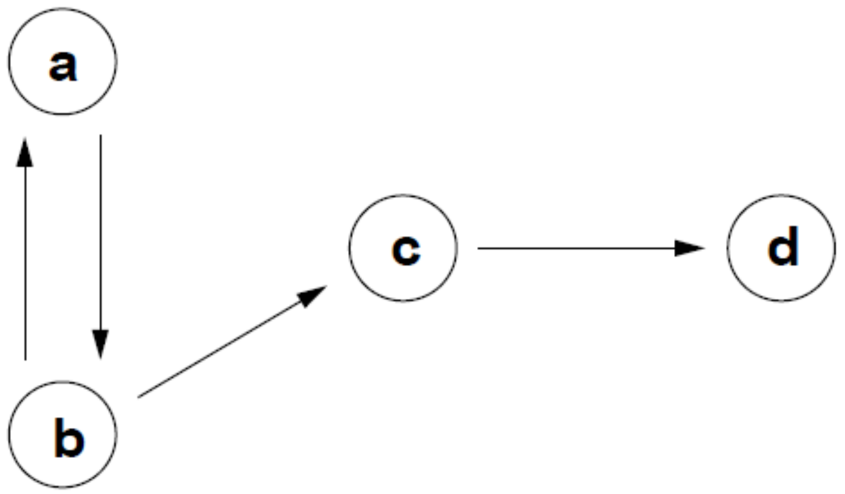
Reminder on graphs

- Computing of transitive closure of a graphe G
  - Given M the adjacency matrix of G

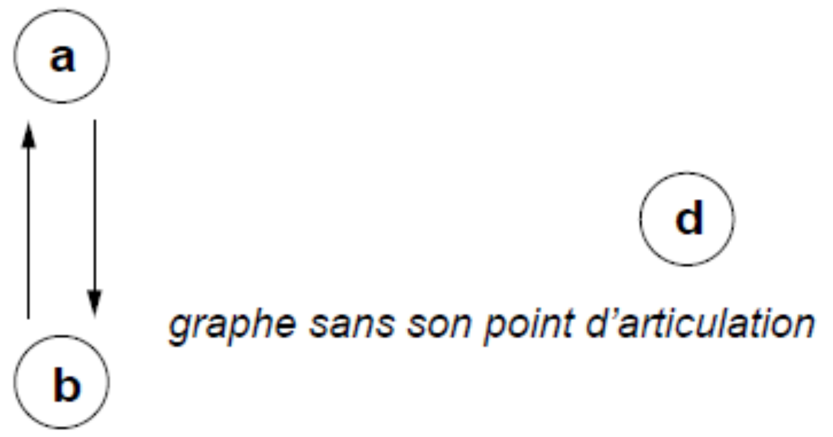
    the matrix M^+ of G^+, transitive closure of G is computed according to :
    M^+ = M + M^1 + … M^{n-1}

Reminder on graphs

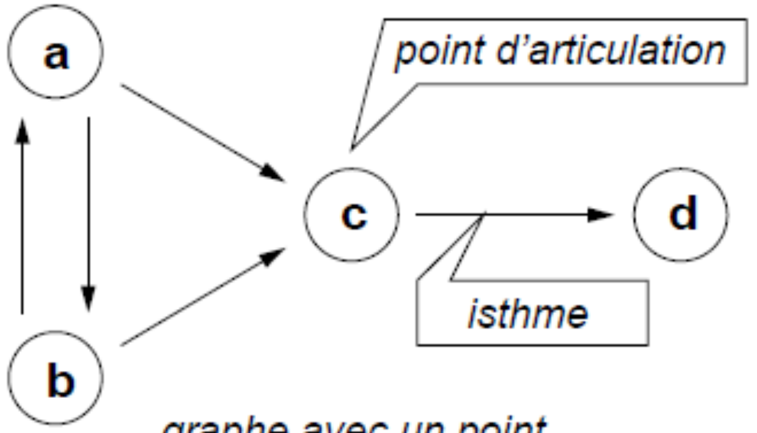
- Eulerian path, Hamiltonian path, …
- Connexity and strong connexity
- isthmus and point of articulation


$$\begin{tikzcd}
a \arrow[d, shift left] & & \\
b \arrow[u, shift left] \arrow[r] & c \arrow[r] & d
\end{tikzcd}$$

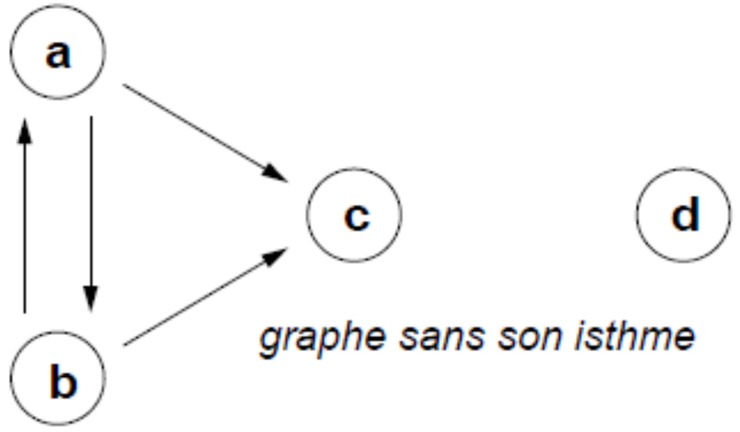
Reminder on graphs



graphe sans son point d'articulation



graphe avec un point articulation et un isthme



graphe sans son isthme

To come …

- Next week :
  - 11th of january from 14h p.m. to 16h exam 50 % of the mark
  - Delivery of practical works until the 15th of january
  - Delivery of the project until the 16th of january