Introduction to combinatorial optimization, modeling and complexity theory

I. Introduction to combinatorial optimization and graph theory

1. Basics of Operations Research; Graph theory: basic notions, simple graph algorithms, examples Connectivity,
2. Shortest path problems, Ford algorithm,Dijkstra, Bellman algorithms, scheduling
3. Flows in networks
4. Introduction to linear programming (LP), Modeling combinatorial problems through LP, solving PL through Excel solver, exercises

History…

- **Léonard EULER**: 1707-1783
  - Seven Bridges of Königsberg
- **Charles BABBAGE**: 1791-1871(**Ada LOVELACE** : 1815-1852**)**
    - **Design of computers:** Babbage sought a method by which mathematical tables could be calculated mechanically, removing the high rate of human error.
- **Alan TURING:** 1912-1954
  - The Turing machine
  - Decrypting the Enigma code (**Combinatorial**),
  - COLOSSUS: one of the first computers

History…

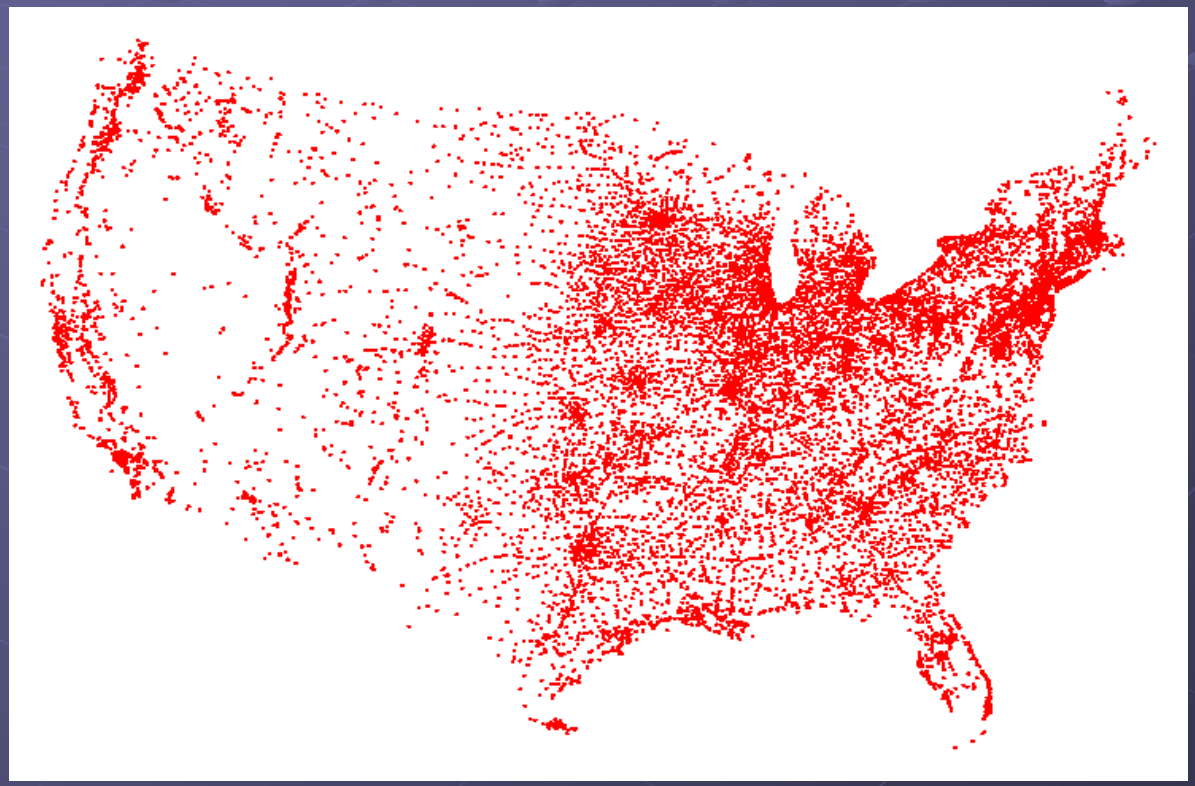
- **Léonid KANTOROVITCH : 1912- 1986**
  - a pioneer of linear programming… transport program,
  - Nobel price in economics (1975)
- **Georges Bernard DANTZIG : 1914- 2005**
  - Linear programming
- **Paul ERDOS and Alfred RENYI**
- **Albert-Lazlo BARABASI, Claude BERGE, Ken APPEL and Wolfgang HAKEN,**
- **Jack EDMONS, Bernard Roy, Paul ROBERTSON and Neil SEYMOUR, Robert TARJAN**

Some problems of Operations Research

- *Discrete combinatorial problems*
  - *Travelling salesman problem,*
  - *Minimum spanning tree*
- *Continuous combinatorial problems*
  - *Linear programming,*
- *Random problems*
  - *Queuing theory*
  - *Equipment replacement*
- *Competitive situations*
  - Game theory

Discrete combinatorial problems
Travelling salesman problem

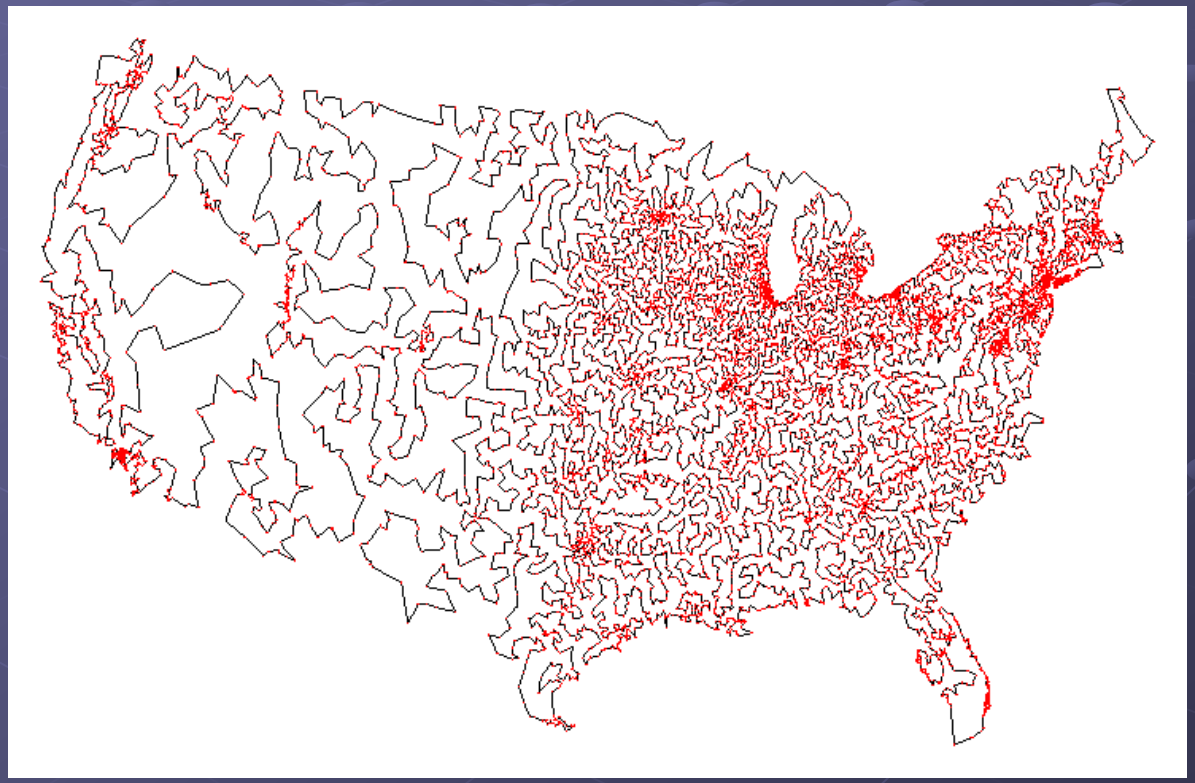
- TSP. Given a set of n cities and a pairwise distance function d(u, v), is there a tour of length $\leq$ D?



All 13,509 cities in US with a population of at least 500
Reference: http://www.tsp.gatech.edu

Discrete combinatorial problems
Travelling salesman problem

- TSP. Given a set of n cities and a pairwise distance function d(u, v), is there a tour of length $\leq$ D?



Optimal TSP tour
Reference: http://www.tsp.gatech.edu

Continuous combinatorial problems
Linear programming

Example. Suppose that a farmer has a piece of farm land, say *A* square kilometers large, to be planted with either wheat or cereals or some combination of the two.

The farmer has a limited permissible amount *F* of fertilizer and *P* of insecticide which can be used, each of which is required in different amounts per unit area for wheat (*F*1, *P*1) and cereals (*F*2, *P*2).

Let *S*1 be the selling price of wheat, and *S*2 the price of cereals produced per unity of planted area (one square kilometer). If we denote the area planted with wheat and cereals by *x*1 and *x*2 respectively, then the optimal number of square kilometers to plant with wheat and cereals can be expressed as a linear programming problem:

maximize $S_1x_1 + S_2x_2$ (maximize the revenue)

subject to:

$x_1 + x_2 \leq A$ limit on total area)

$F_1x_1 + F_2x_2 \leq F$ (limit on fertilizer)

$P_1x_1 + P_2x_2 \leq P$ (limit on insecticide)

$x_1 \geq 0,\ x_2 \geq 0$ (cannot plant a negative area)

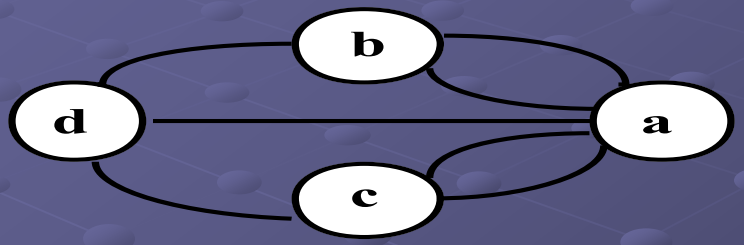
Random problems

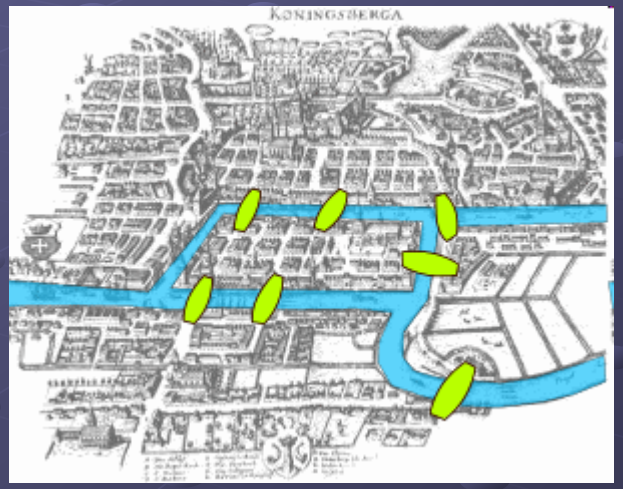
- *Queuing theory*
  - *applications to (internet) network congestion;*
  - *ordering the take-off of aircraft.*
- *Equipment replacement*
  - *Deciding the replacement date for equipments with given failure probability.*

Game theory

- Game theory
  - In mathematics, **game theory** models strategic situations, or *games*, in which an individual's success in making choices depends on the choices of others (Myerson, 1991).

Why using graphs?





Seven bridges of Königsberg

Given the above graph, is it possible to construct a path (or a cycle, i.e. a path starting and ending on the same vertex) which visits each edge exactly once?

Basic definitions

- Directed Graphs, undirected graphs
- Walks/chains, cycles, paths, directed paths, cocycles,
- Particular graphs
- Independent set, clique…

Basic definitions

- A **directed graph** or **digraph** is a pair *G*= (*X*, *U*) of:
  - a set *X*, whose elements are called **vertices** or **nodes,**
  - a set *U* of ordered pairs of vertices, called **arcs**, **directed edges**, or **arrows**.
- It differs from an ordinary, or undirected graph in that the latter one is defined in terms of edges, which are unordered pairs of vertices.
- A valuated graph is G = (X, U, v) where (X, U) is a graph and v an application from U to R (real numbers).

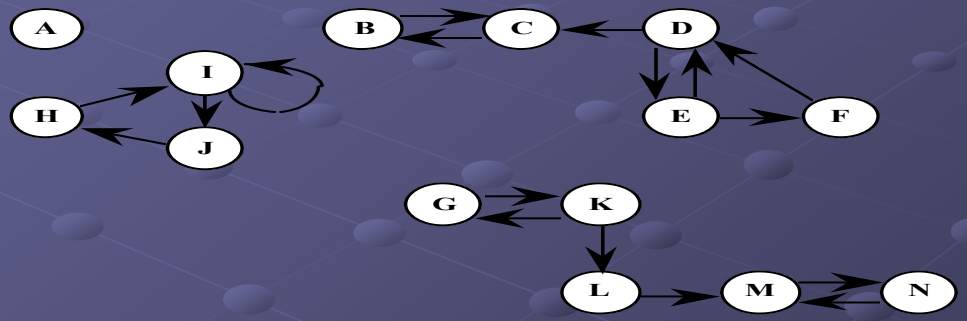
- Successors, predecessors, vertex degrees…

Basic definitions

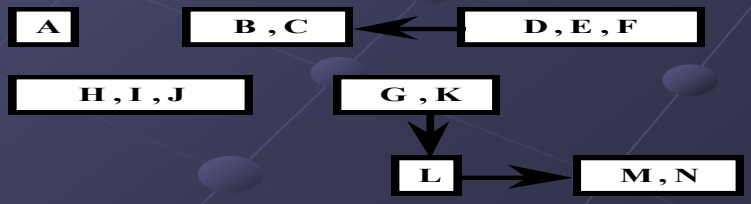
- A **walk (chain)** is an alternating sequence of vertices and edges, beginning and ending with a vertex, where each vertex is incident to both the edge that precedes it and the edge that follows it in the sequence, and where the vertices that precede and follow an edge are the end vertices of that edge. A walk is **closed** if its first and last vertices are the same (called a **cycle)**, and **open** if they are different (called a path).
- A **trail** is a walk in which all the edges are distinct;
- The **length** *l* of a walk is the number of edges that it uses.
- A directed path is when edges “have the same orientation”
- A directed cycle: without the arrows, it is just a cycle.
- A path is **simple (resp. elementary)**, meaning that no vertices (resp. no edges) are repeated.
- A graph is **acyclic** if it contains no cycles;
- A path or cycle is **Hamiltonian (resp. Eulerian)** if it uses all vertices (resp. edges) exactly once.

Connectivity

- Simple **Connectivity**: connected component
- Strong **Connectivity**: strong connected component



- Reduced Graph:



Connectivity and strong connectivity relations

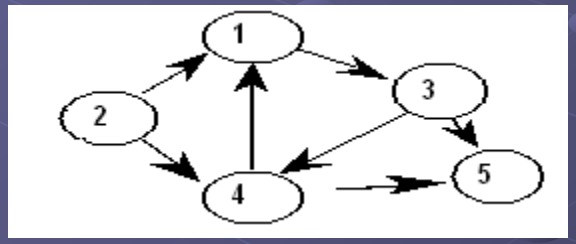
- An **equivalence relation** is a binary relation on a set that specifies how to split up (i.e. partition) the set into subsets such that every element of the larger set is in exactly one of the subsets.
  - *reflexive*, *symmetric* and *transitive*.

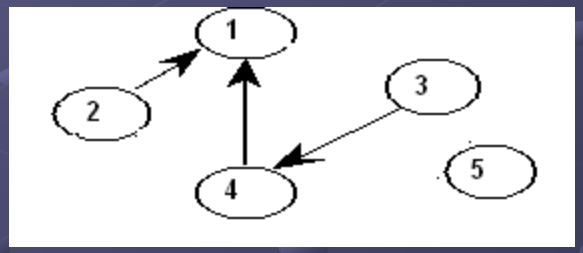
- **equivalence class of** *x* in *E* , denoted R(x), is given by: R(x)={y: xRy}.

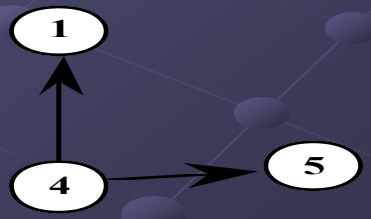
- *What can-we say about connectivity and strong connectivity relations?*

Associated graphs

- Subgraph, spanning subgraph,
- Induced subgraph
- Complementary graph

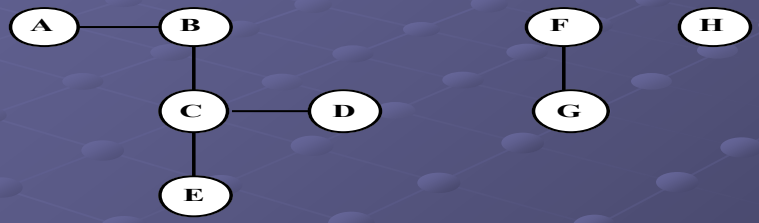




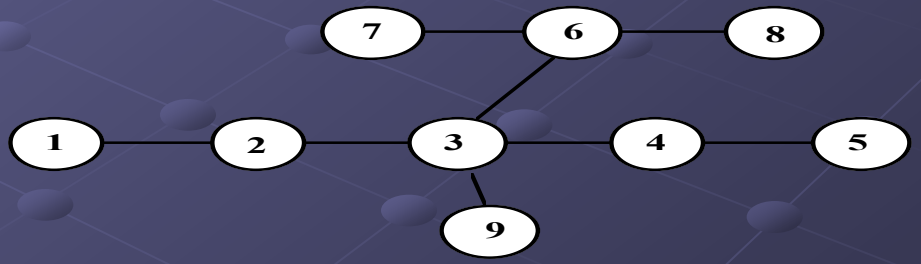


Particular Graphs

- Forest is a graph without cycle



- Tree is a connected graph without cycle



  - Spanning tree

Exercise

Let G be a non oriented graph with n vertices and m edges. The edges of G are noted u1, u2 .. um. Gi=(V, Ei), is the graph created with the edges u1, u2 .. ui. Let us write : G0= (V, Φ).

Question 1. How many connected components of G0 are there ?

Question 2. Let us note C(Gi) the number of connected components of Gi. Show that

C(Gi) = C(Gi-1) or C(Gi) = C(Gi-1) – 1.

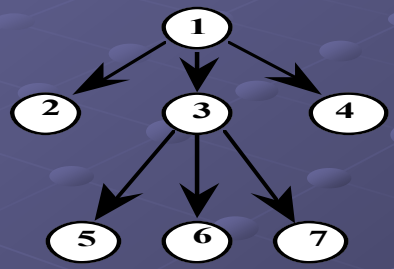
Question 3. Interpret the case where C(Gi) = C(Gi-1), then the case where C(Gi) = C(Gi-1) –1.

Question 4. Show that a connected graph has at least n-1 edges.

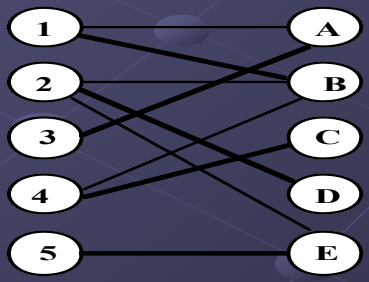
Question 5. Show that a graph without any cycles has at the most n-1 edges.

Particular Graphs

- In graph theory, an **arborescence** is a directed graph in which, for a vertex *v* called the root and any other vertex *u*, there is exactly one directed path from *v* to *u*.

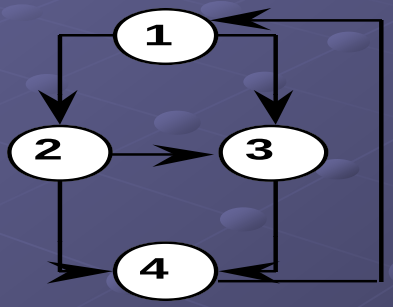


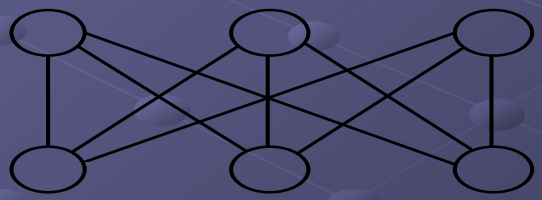
A bipartite graph is a graph whose vertices can be divided into two disjoint sets *U* and *V* such that every edge connects a vertex in *U* to one in *V*;



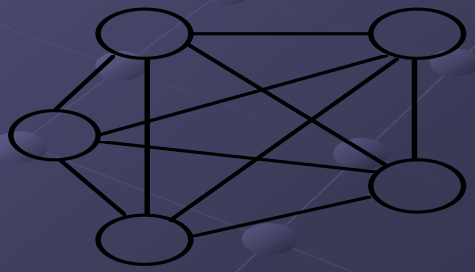
Particular Graphs

- In graph theory, a **planar graph** is a graph which can be embedded in the plane, i.e., it can be drawn on the plane in such a way that its edges intersect only at their endpoints.





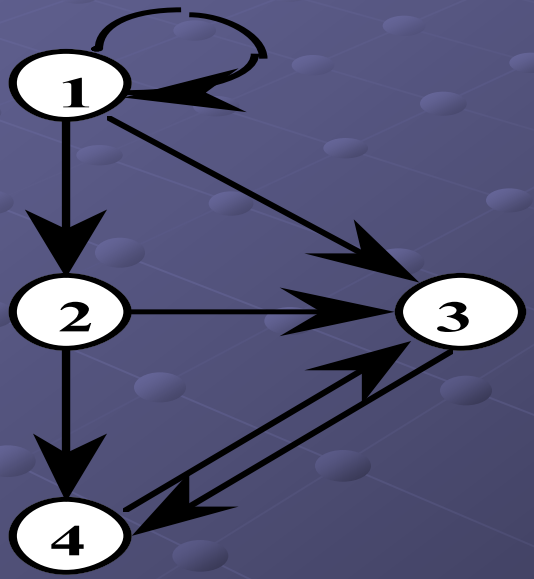
Graph of 3 entreprises ()



Complete Graph of 5 vertices ()

Coding a graph

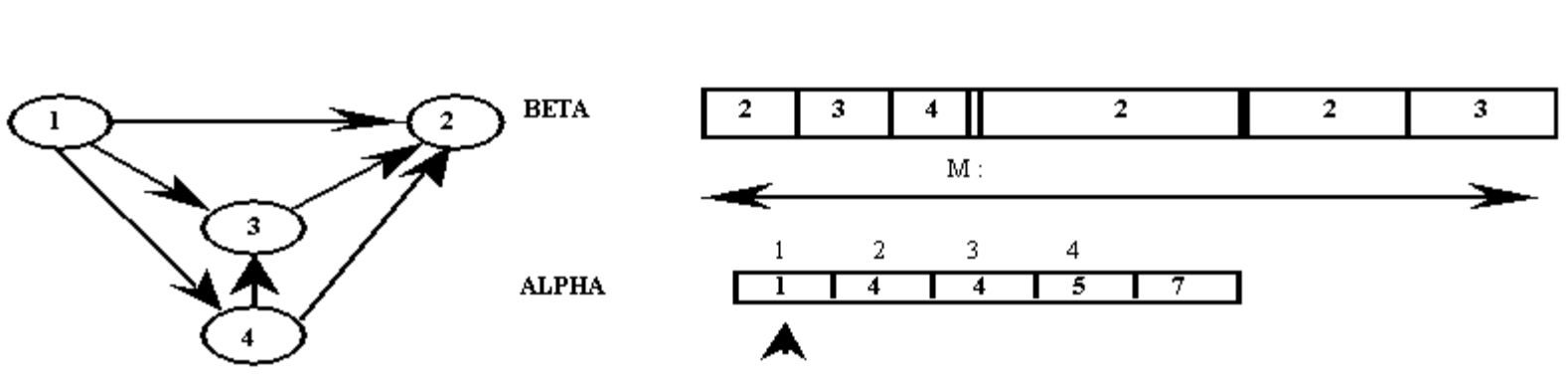
- Adjacency Matrix



$$\begin{vmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{vmatrix}$$

Coding a graph

- Successor queue β and α



Exercise: Write an algorithm which allows the passage from the successor queue to the adjacency matrix.

Algorithm's evaluation (I)

- An algorithm: a set of instructions describing how some task can be achieved or a problem can be solved.
  - A program is the computational implementation of an algorithm.

- Any algorithm is composed of two main stages: initialization and computing one
- The complexity parameter is the size data n (binary coding).

Definition:

Let be $n>0$ and $T(n)$ the running time of an algorithm expressed in terms of the size data n, $T(n)$ is of $O(f(n))$ iff $\exists\, n_0$ and some constant c such that:

$\forall n \geq n_0$, we have $T(n) \leq c\, f(n)$.

If f(n) is a polynomial then the algorithm is of polynomial complexity.

- Study the complexity in the worst case.

Algorithm's evaluation (II) example

Given N numbers a_1, a_2, .., a_n in {1,…, K}.

The algorithm MIN finds the minimum value.

Algorithm MIN

```
Begin
  for i=1 to n read a_i;
  B:=a_1, j:=1;
  for i=2 to n do :
            if B=1 then j:=n;
            elseif a_i < B then
                        begin
                                    j:=i;
                                    B:=a_I;
                        end;
  write : the min value is a_j;
End.
```

Exercises (I)

Exercice 1 :

Prove KOENIG's lemma : it is always possible to extract, from a (directed) path going from i to j, a simple path going from i to j.

Exercice 2 :

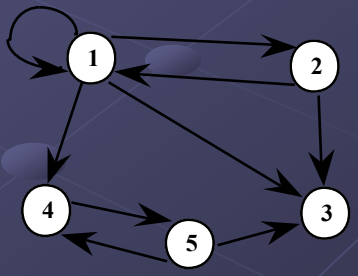
- Which methods would you suggest to code a graph using a computer ?

Exercises (II)

Exercice :

Two methods for coding a graph using a computer are the following : the adjacency matrix A and the successor queue with the tables ALPHA1, ALPHA2 and BETA. The associated matrix est defined by : A = (aij) with aij = 1 if the edge (i,j) belongs to U and aij =0 otherwise. The successor queue BETA is the successor table, with the successors of the vertex I located between the addresses ALPHA1(I) and ALHPA2(I) in the table BETA, (in the lecture only one table ALPHA was seen).

- Determine the tables A, ALPHA and BETA for the following graph :

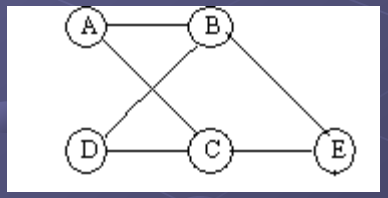


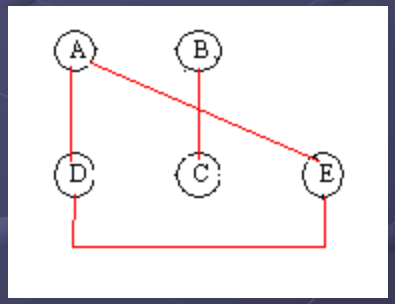
Exercises (III)

- Determine the number of edges of a complete undirected graph with n vertices.
- Explain why, if G=(X,E) is an undirected graph without any loops, the sum of the degrees is equal to two times the number of edges of the graph.

Basic definitions

- An **independent set** or **stable set** is a set of vertices in a graph no two of which are adjacent. The size of an independent set is the number of vertices it contains ($\alpha(G)$).
  - A maximal independent set is an independent set such that adding any other node to the set forces the set to contain an edge.
- A **clique** in an undirected graph G, is a set of vertices V, such that for every two vertices in V, there exists an edge connecting the two. e

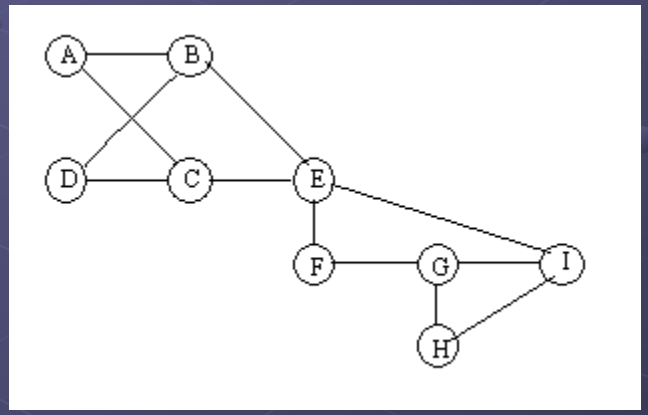




A **clique** in a graph G corresponds to a stable in its complementary graph and vice-versa.

Basic definitions

- Some graph G is called **c-chromatic** if its vertices can be colored with c colors such that no two adjacent vertices have the same color. Similarly, an **edge coloring** assigns a color to each edge so that no two adjacent edges share the same color.



Conjecture of 4 colors (1875 Pertersen) : “all planar graphs are 4-chromatic”

Four colors theorem

Exercises

- Bipartite graph and bicoloration
- Minimum cost spanning tree

Bipartite graph and bicoloration (I)

Algorithme BIPAR(G) :

{-1- Initialisation}

Read a connected graph G=(X,U); {initially, no vertex is coloured}

colour the vertex 1 in red;

initialise the boolean variable BIPARTI to TRUE.

{-2- Examination of the coloured vertices}

While a non examined, coloured vertex i exists do

Begin

If i is red then for each neighbour j of i do

Begin

if j is red then BIPARTI=FALSE

if j is non coloured then colour j in green

End

else { i is therefore green } for each neighbour j of i do

Begin

if j is green then BIPARTI=FALSE

if j is non coloured then colour j in red

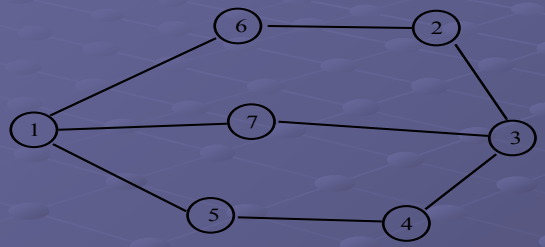
End

End

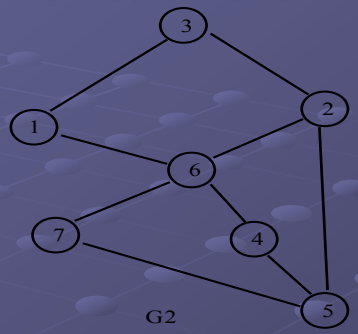
{-3- Exiting the algorithm} : If BIPARTI = TRUE then write "G is bipartite", else "G is not bipartite".

Bipartite graph and bicoloration (II)

Question 1: Apply the algorithm to the graphs G1 et G2.Show the colours for each iteration of the loop.



G1



G2

Question 2: We shall code the graph G=(X,U) using the associated matrix A, in other words if (i,j) is an edge of the graph, then aij = aji =1 , otherwise aij =0.

(a) Which extra tables will be necessary for BIPAR(G) ?

(b) What is the resulting complexity of the algorithm ? Explain.

Question 3: Propose, in detail, a better data structure. What does the complexity become ? Explain.

Question 4: (This question is independant of the algorithm) Show that a graph is bipartite if and only if it is bicolourable.

Question 5: (This question is independant of the algorithm)

Show that a bipartite graph contains no cycle which has an odd length.

Question 6: What are the properties of vertices coloured in red and green during the algorithm. Explain. Where does the fact that the graph is connected come into play ?

Question 7: Prove the validity of the algorithm. Deduce the reciprocal of question 5 in the case of connected graphs.

Minimum cost spanning tree (I)

KRUKSAL's algorithm determines a spanning tree of minimal cost.

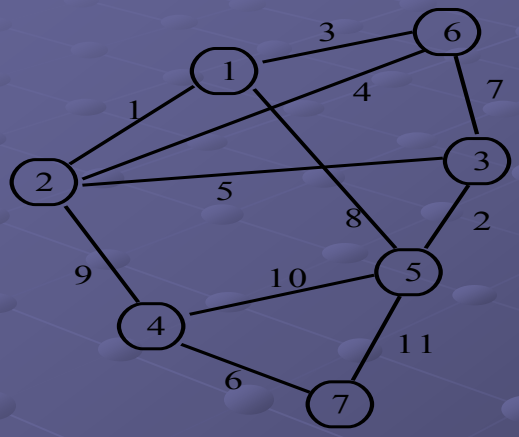
It shall be assumed that the costs are all distinct.

KRUSKAL

```
{conventionally, the edge eL is written eL = [I, J] with I strictly smaller than J}
Start
        Index the edges in order of increasing cost: v (e1) ≤ v (e2) ≤ ... ≤ v (em)
        For H: = 1, N do CONNEXE(H): =H;
        K: = 1; (the subgraph is initially empty)
        For L: = 1, M do
        Start
                set eL = [I, J];
                If (CONNEXE(I) different to CONNEXE(J)) then
                Start
                        AUXI= CONNEXE(J);
                        fK=[I, J] (fK is put into the  subgraph)
                        K:= K+1;
                        For H: = 1, N do
                        If (CONNEXE(H) = AUXI) then CONNEXE(H):= CONNEXE(I)
                        End
        End
End
```

Minimum cost spanning tree (II)

Question 1: Run the algorithm on the graph above : draw the successive subgraphs and the eleven successive values of the CONNEXE table obtained before each iteration the outer loop.



Question 2: Evaluate the complexity of the algorithm in function of N and M. Two types of iterations in the loop shall be distinguished.

Question 3 (this question is independent of the algorithm) *Reminder : this is the case where all the costs are distinct.*

Show that a spanning tree has a minimum cost iff the intersection of each cocycle of G and the tree contains the edge of minimum cost of this cocycle.

Question 4: Show that the algorithm determines the spanning tree of minimal cost.