Graphs or Networks
Les graphes ou réseaux

Michel Habib
habib@irif.fr
http://www.irif.fr/~habib

Cours predoc, January 19, 2026

Content

- Introduction to the graph object as a model. Precise definitions.
- Degree theorems. Limits of the model.
- Minimum weight spanning trees, Steiner trees.
- Matroidal structures and greedy algorithms.
- Graph traversal and shortest path algorithms
- Flows algorithms and matchings
- Spectral graph theory and application to PageRank (Google)
- We'll be introducing the essential notion of a certificate, which guarantees the result produced by an algorithm – program.

The undirected graphs considered in this course will, unless otherwise stated, be **simple**: without multiple arcs (several arcs between two vertices). In addition, we'll assume no **loop** (edge of type xx).
The graph structure is a powerful modeling tool for formalizing many problems. In some models, loops are essential; in others, multiple edges are indispensable. Certain graph operations, such as vertex contractions, naturally give rise to loops and multiple edges.

A database of interesting graphs and more

https://houseofgraphs.org/

- H is an induced subgraph of G, if $V(H) \subseteq V(G)$ and if $E(H) = \{e \in E(G) |$ if the two end points of e belong to $V(H)\}$.
- H is a partial subgraph of G, if $V(H) \subseteq V(G)$ and if $E(H) \subseteq E(G) \cap V(H)^2$.
- Quicksands In many papers both are denoted as subgraphs. Please check the definition at the beginning of the paper, since the notions are very distinct.

For an undirected graph G, in general $|V(G)| = n$ (vertex set) and $|E(G)| = m$ (edge set), for the evaluation of the algorithmic complexity on graphs.
The neighbourhood of a vertex is:
$N(x) = \{y \in V(G) \text{ such that } xy \in E(G)\}$
The **degree** of a vertex denoted by $d(x)$ is the number of edges adjacent to x. Using previous notations $d(x) = |N(x)|$.
A **pendant** vertex in a graph, is a vertex x such that: $d(x) = 1$.

A **path** of length k is a graph:
$P = (\{x_0, x_1, \ldots, x_k\}, \{x_0x_1, x_1x_2, \ldots x_{k-1}x_k\})$ having $k+1$ vertices and k edges, i.e. often denoted by P_k. This graph is called a path of length k joining x_0 to x_k.
The path is often confused with one of two sequences:
$[x_0, x_1, \ldots, x_k]$ or $[x_k, x_{k-1}, \ldots, x_0]$. A graph reduced to one vertex and with no edge is a path of length 0.
A **cycle** of length k is a graph:
$C = (\{x_1, \ldots, x_k\}, \{x_1x_2, x_2x_3, \ldots x_{k-1}x_k, x_kx_1\})$ having k vertices and k arêtes, noting C_k. The cycle is often confused with one of the following notation: $[x_1, x_2, \ldots, x_k, x_1]$ (up to a circular permutation).

With such definitions, a path (resp. a cycle) only passes through a vertex once, which corresponds to an elementary path (resp. cycle) for some authors.
A path (resp. cycle) of a graph G is said to be Hamiltonian if the set of vertices in the path (resp. cycle) is exactly $V(G)$ the set of vertices of G. In other words, a Hamiltonian path or cycle "passes" once and only once through each vertex of the graph.

We call **walk** ou en Français **marche** a non empty séquence $M = x_0 e_1 x_1 e_2 \ldots e_k x_k$ alternating vertices and edges in G such that:
$\forall 1 \leq i \leq k$, the ends of e_i are x_{i-1} and x_i. The walk is said to be of length k and to join x_0 to x_k. This notion is very useful for modeling certain probabilistic problems ("Markov chains" and random walks).
When $x_0 = x_k$, the walk is said to be **closed** or a **pseudocycle**. If the graph is simple, we can omit the edges and denote the walk by $[x_0, x_1, \ldots, x_k]$.

Given the definition, a walk can use the same edge several times; when all the edges of the walk are different, the walk is said to be a **path**[1]; finally, when all the vertices are distinct, the vertices and edges of the walk form a partial graph of G, which is a chain.
A path is **Eulerian** if it uses all the edges of the graph. In other words, an Eulerian path "passes" each edge once and only once.

[1]Please note that these definitions of chains, walks, paths and other cycles are not yet standard in the literature

Proposition

If there is a walk of finite length joining x to y in G, then there is a path joining x to y in G, and the vertices and edges of this chain belong to the initial walk.

proof

Since there is at least one walk of finite length in G joining x to y, among all walks of finite length, there is at least one of minimum length joining x to y in G. Let's denote this walk by M. If it is not elementary, we can extract a walk of strictly smaller length, hence the contradiction.

$\square$

N.B. This proof relies on the finiteness of the walk, because if M is of infinite length, it's not certain that M' the extracted walk is of length strictly smaller, and then we have no contradiction.

Similarly

Proposition

If there is a pseudo-cycle of finite length in G, then there is a cycle of finite length in G.

Connectivity

A graph G is **connected**, if $\forall x, y \in V(G)$, there exists a path from x to y.

Proposition

Consider a finite connected graph G, if $\forall x \in V(G)$, $d(x) \leq 2$, then G is either a path or a cycle.

proof

If G has no edge, given the connectivity hypothesis, G is reduced to a vertex and is therefore a path of length 0. Otherwise, G possesses at least one edge, and therefore at least one path, and let C be a path of maximum length G. Only the extremes x, y of the path can be adjacent to edges not belonging to the path (since the other vertices of the path are of degree 2). But if these edges have an extremity outside the path, the path would not be of maximum length. So either $d(x) = d(y) = 1$ and G is a path, or $xy \in E(G)$ and G is a cycle.
$\square$

As a consequence

A finite connected undirected graph G for which $\forall x \in V(G)$, $d(x) = 2$, G is a cycle.
But
an infinite connected undirected graph G for which $\forall x \in V(G)$, $d(x) = 2$ G is an infinite path.

Trees

theorem

The following 6 conditions are equivalent and characterize the finite trees:

1. G is minimally connected (if an edge is removed, the graph is no longer connected)
2. G is maximal cycle-free (if an edge is added, the graph admits a cycle)
3. G is connected without cycle
4. G is connected with $n-1$ arêtes
5. G is cycle-free with $n-1$ arêtes
6. $\forall x, y \in X$, there exists a unique chain joining x à y.

Lemma

A connected graph with $n-1$ edges necessarily has a pendant vertex.

proof

Since G is connected, $\forall x \in V(G)$, $d(x) \geq 1$ (there is no isolated vertex). If there is no pendant vertex, then $\forall x \in X$, $d(x) \geq 2$ and therefore $\Sigma_{x \in V(G)} d(x) = 2m = 2n-2 \geq 2n$, therefore $n-1 \geq n$, hence the contradiction. $\square$

Lemma

A G graph without cycle, having at least one edge, necessarily has a pendant vertex.

proof

There is at least one path in G. Let's consider a path $[x = x_1, \dots, x_k = y]$ of maximum length.
If x is not a pendant vertex, then it admits another neighbor z different from x_2. If $z \in \{x_3, \dots, x_k\}$, then the tree possesses a cycle, which is excluded. So $z \notin \{x_2, \dots, x_k\}$, but then the chosen path wasn't maximal.
$\square$

Exercises

1. Show that an undirected connected graph without loops or multiple arcs has at least two vertices of the same degree.
2. What is the structure of a graph G whose vertices all have a degree equal to 2?
3. Show that in any finite undirected graph, there is an even number of odd-degree vertices.
   Remark: if the graph is not finite, this property is false, as shown by the graph of the successor function in N, which has only one vertex of degree 1, all the others being of degree 2.

Finite versus infinite for graphs

Modeling a problem by means of a graph can be restrictive, as not everything can be modeled using graphs. For example, no graph is an infinite cycle.
For simplicity's sake, let's assume that the cycle is countably infinite and denote it by $[x_1, x_2, \ldots, x_i \ldots]$, the vertices being indexed by $\mathcal{N}$. x_1 admits a neighbor other than x_2, let's denote it by x_j, but then the cycle is $[x_1, x_2, \ldots, x_i \ldots, x_j]$ and is therefore finite, a contradiction. Similarly, consider the following variant of the previous proposition.

Proposition

Consider a finite connected graph G, if $\forall x \in V(G)$, $d(x) = 2$, then G is a cycle.

Let $P = (X, \leq)$ a partial order.
If P is finite all linear extensions are isomorphic.
If P is infinite it could be false as shows the example of the partial order obtained by 2 infinite parallel chains N_1 and N_2 both isomorphic the N (natural integers).
Let us define τ a level by level linear extension and $\sigma = N_1 \cdot N_2$.
$|\tau = \omega$ and $|\sigma| = 2\omega$, so they cannot be isomorphic !

Another example

Planar graphs for which the duality does not work for infinite graphs.

Euler's theorem

Theorem

A finite connected graph admits an Eulerian pseudo-cycle (i.e. a cycle which uses exactly once every edge) iff all its degrees are even.

- Prove this theorem.
- Design a linear time algorithm to compute such an Eulerian pseudo-cycle.

In the previous proposition and theorem:

Note: The graph of the successor function in $\mathcal{Z}$, which is a chain, provides a counterexample to this proposition when the graph is infinite.

Ramsey numbers

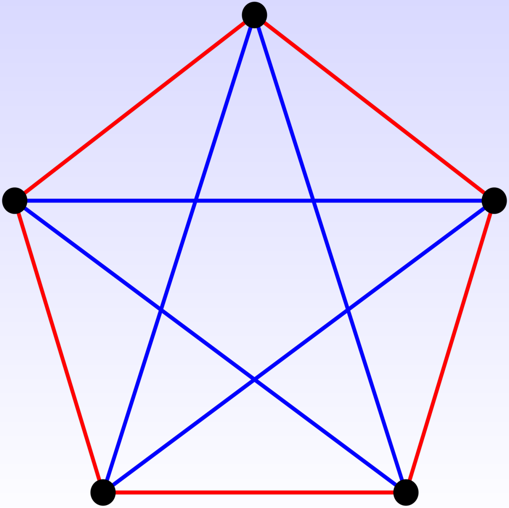
$Ramsey(3,3) = 6$

In other words, if we color the edges of the complete graph K_n with two colors, $n \geq 6$, there is always a monochromatic triangle.

Proof

$Ramsey(3) \geq 6$, because K_5 can be seen as the sum of two monochromatic cycles has 5 elements.
$Ramsey(3) \leq 6$. Consider a vertex $x \in K_n$, with $n \geq 6$, so $d(x) \geq 5$. Suppose x does not belong to any monochromatic triangle. Since $d(x) \geq 5$, there are at least 3 edges adjacent to x with the same red color: xa, xb, xc. If one of the edges ab, bc or ac, e.g. ab, but then the triangle abx is uni-coloured. So edges ab, bc, ac are blue. But then the a, b, c triangle is necessarily monochromatic (blue).

K_5 with no monochromatic triangle



Corollary

Any graph with more than 6 vertices has either a triangle or an independent set of 3 elements.

Or equivalently

This result can be rephrased as follows: at a party attended by at least six people, there are at least three people who know each other or at least three who are strangers to each other.

Frank Ramsey numbers (1903-1930)

$Ramsey(s, t)$ is the smallest integer n such that any graph with more than n vertices has either an independent independent of size s or a clique of size t.

Ramsey showed the existence of these numbers $Ramsey(s, t)$ for any pair s, t of integers.
We don't know their exact values, just a recurrence inequation:
$Ramsey(s, t) \leq Ramsey(s - 1, t) + Ramsey(s, t - 1)$
This shows that these numbers are finite and the bound:
Furthermore:
$Ramsey(s, t) \leq \binom{s+t-2}{s-1}$

$Ramsey(4,4)$

To show $Ramsey(4,4) = 18$, we use the probabilistic method.
We take a complete graph with 18 vertices.

Next values ?

- Nobody knows the exact value of $Ramsey(5,5)$
- We simply know $43 \leq Ramsey(5,5) \leq 46$.
- The general bound, which is very hard to improve significantly: $2^{k/2} \leq Ramsey(k,k) \leq 4^k$. Recenlty using computers $Ramsey(5,5) = 43$
-

Erdös's history

On Wikipedia there is a famous quote of Paul Erdös showing the complexity of computing Ramsey numbers,

Erdös asks us to imagine an alien force, vastly more powerful than us, landing on Earth and demanding the value of R(5) or they will destroy our planet. In that case, he claims, we should marshal all our computers and all our mathematicians and attempt to find the value. But suppose, instead, that they ask for R(6). In that case, he believes, we should attempt to destroy the aliens.

Ramsey Theory

- Ramsey theory, named after the British mathematician and philosopher Frank P. Ramsey, is a branch of mathematics that studies the conditions under which order must appear. Problems in Ramsey theory typically ask a question of the form: "how many elements of some structure must there be to guarantee that a particular property will hold?"
- Application: in the game of cards "SET" , the warranty that there exists always a solution with a set of 12 cards is a Ramsey type theorem for hypergraphs or edge-colored graphs. see `https://www.youtube.com/watch?v=tHo830obZ38`. Where they explain that every set of 20 cards have a solution (kind of clique).

- The very important and powerfull Szemerédi's regularity lemma is another example of these properties.

The politician's theorem

Theorem

Consider an undirected graph with $n \geq 3$ vertices. If each pair of vertices has exactly one common neighbor, then there exists a vertex that is adjacent to all vertices.

Remark

Conditions on degrees can characterize graphs, but this only works for finite graphs.

Notations

Here we deal with finite loopless and simple undirected graphs.
For such a graph G
we denote by $V(G)$ the set of its vertices
and by $E(G)$ the edge set
By convention $|V(G)| = n$ and $|E(G)| = m$

Bound on the number of edges

Triangle free graphs (Turan's theorem)

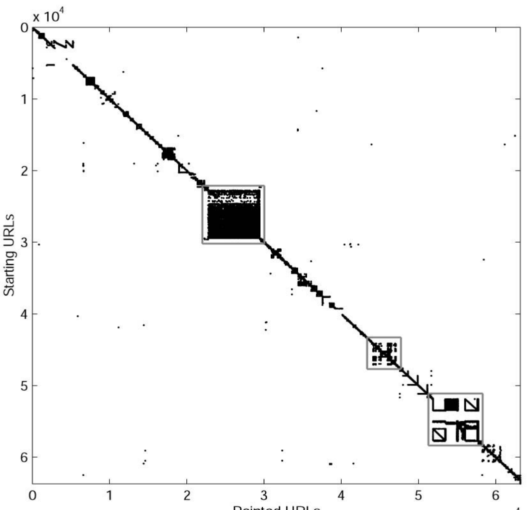
Show that if G has no triangle then :
$|E(G)| \leq \frac{|V(G)|^2}{4}$

Planar graphs

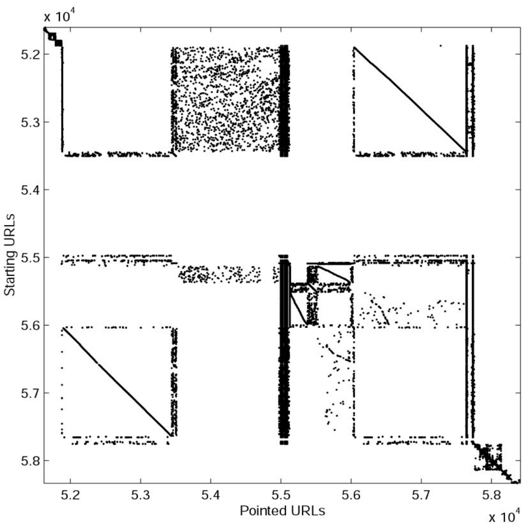
Show that if G is a simple planar graph (i.e. without loop and parallel edge) then:
$|E(G)| \leq 3|V(G)| - 6$

Sparse graphs satisfy : $|E(G)| \in O(|V(G)|)$
Planar graphs are sparse, but also many graphs coming from applications are sparse.
For example the WEB graph is sparse.
For sparse graphs one has to use an adjacency lists representation.

Matrice ordonnée par l'ordre alphabétique des noms des URL



Un zoom autour de la diagonale



- Implicit hypothesis: the memory words have k bits with $k > \lceil log(|V(G)|) \rceil$
- To be sure, consider the bit encoding level

- Adjacency lists
  $O(|V(G)| + |E(G)|)$ memory words
  Adjacency test: xy is an arc in $O(|N(x)|)$
- Basic one :
  The number of vertices and a list of edges. This often the format in which you cand find graphs in graph databases.
- Customized representations, a link for each arc . . .
- All these representations are linearly equivalent.

Analogical representations

- With rings and chords.
- With DNA fragments.
- For flow problems satisfying Kirchoff's laws, one can use analogic electric networks to compute the flow.
- Or link into the memory to directly mimick the edges.

Adjacency Matrix

Adjacency Matrix
$O(|V(G)|^2)$ memory words (can be compressed)
Adjacency test: xy is an arc in $O(1)$

Comment

This representation is not linearly equivalent to the previous one (adjacency lists)

Representations in between the 2 previous ones

- Use a heap to represent the adjacency lists of vertices.
- access to an edge in $O(logn)$, memory size $O(n+m)$.
- Unpleasant feature : not possible to write a true linear algorithm
- I saw these representations in part of SAGE library.

Exercise*

How to combine the advantages of both representations?
adjacency list: $O(n+m)$ construction
adjacency matrices: access to an arc in $O(1)$

$O(n^2)$ space, but still able to write linear algorithms in time?

Quadratic space in linear time

- Select a 2-dimensional array GRAF of size n^2
  construct an auxillary unidimensional array of size m EDGE:
  For j=1 to m
  xy being the j^{th} edge of G
  $GRAF[x, y] = j$
  $EDGE[j] =$ a pointer to the memory word $GRAF[x, y]$
- The construction of the EDGE array requires $O(m)$ time
- Memory used $n^2 + m \in O(n^2)$

- $xy \in E$ iff $EDGE[GRAF[x, y]]$ contains a pointer pointing to the memory word $GRAF[x, y]$
- Therefore the query : $xy \in E$?
  Can be done in 2 tests $O(1)$.

For some large graphs, the Adjacency matrix, is not easy to obtain and manipulate.
But the neighbourhood of a given vertex can be obtained. (WEB Graph or graphs is Game Theory)

Sorting the vertices by their degrees

1. Compute the degrees by scanning the adjacency lists in an array D
2. Use any per value sort algorithm (Radix or other) since all numbers are bounded by n

The previous solutions need that the data structure is available at once in the memory.
This will be implicit in the remaining of the course, as well as the RAM model.

Exercises

Suppose that a graph $G = (V, E)$ is given by its adjacency lists and let σ be some total ordering of its vertices.

- How can we sort the adjacency with respect to σ (increasing)?
- What is the complexity of this operation?
- Let σ be the total ordering of the vertices with decreasing degrees, how to compute σ?

Sorting an adjacency list

Suppose that a graph $G = (V, E)$ is given by its adjacency lists A and let σ be some total ordering of its vertices. How can we sort the adjacency with respect to σ (increasing) ?

One solution

Build another adjacency list structure B from the old one by taking the vertices from $\sigma(n)$ down to $\sigma(1)$
read $A(\sigma(i))$ and add $\sigma(i)$ in front of the lists of B corresponding to the neighbors of $\sigma(i)$
At the end the lists of B are sorted in the right way.

Complexity

Time : Linear time complexity (the size of the data structure A).
Memory: Twice the size of the adjacency lists $2|A|$

What is an elementary operation for a graph ?

- Traversing an edge or Visiting a neighbourhood ?
- It explains the very few lower bounds known for graph algorithms on a RAM Machine.
- Our graph algorithms must accept any auto-complemented representation.
  Partition Refinement

The great importance of the right model

It is crucial to use all the characteristics of the problem you want to solve and find a good model. Two examples from Peter Winkler's nice book on Mathematical Puzzles:

All the discrete structures considered here are supposed to be finite.

1. An odd number of soldiers in a field in such a way that all pairwise distances are different. Each soldier is told to keep an eye on the nearest other soldier.
   Show that at least one soldier is not being watched.
2. A famous problem on rectangles:
   A large rectangle of the plane is partitioned into smaller rectangles, each of which has either integer height or integer width (or both). Prove that the large rectangle also has this property.

Exercise 1

Let us consider the 2 nearest soldiers x,y. They both keep an eye on the other. If another soldier watch on x or y, then we are done because it exists at least one soldier that nobody watches.
Else x,y is a 2-bond which is a connected component and we proceed by induction.

Exercise 2

Start with the graph of the planar tiling and keep exactly 2 integers edges by rectangle.
This yields a graph (possibly with parallel edges) in which all vertices have even degrees except the corners.
Take a maximal path starting in one corner ....

Where can we find graphs ?

- Stanford Large Network Dataset Collection
- `https://snap.stanford.edu/data/`
- A universal data exchange format:
  n, m followed by an ordered list of edges

Quicksands

- A sentence like:
  "To compute this invariant or this property of a given graph G one needs to "see"(or visit) every edge at least once".
- False statement as for example the computation of twins resp. connected components on $\overline{G}$ knowing G.

Exercise

Can the advantages of the 2 previous representations can be mixed in a unique new one ?
Adjacency lists : construction in $O(n+m)$
Incidence matrix: cost of the query : $xy \in E(G)$? in $O(1)$

In other words

Using $O(n^2)$ space, but with linear time algorithms on graphs ?

Auto-complemented representations

Initial Matrix

	1	2	3	4
1	1	1	1	0
2	0	0	1	0
3	1	0	1	1
4	1	0	0	0

Tagged Matrix

	$\overline{1}$	2	$\overline{3}$	4
1	0	1	0	0
2	1	0	0	0
3	0	0	0	1
4	0	0	1	0

- At most $2n$ tags (bits).
  $O(n + m')$ with $m' << m$.
  Dalhaus, Gustedt, McConnell 2000
- What can be computed using such representations ?

- Find a minimum sized representation
- If G is undirected a minimum is unique.

Reduced matrices

A line (resp. row) is flipped if it has more than $\lceil \frac{n}{2} \rceil$ ones.
If we apply till the end such a rule we reach a matrix which has less than $\lceil \frac{n}{2} \rceil$ ones per line (resp. row).
The process necessarily terminates since at each step the total number of ones strictly decreases.

The previous algorithm is clearly in $O(n^2)$.
But experimentation show linearity in practice.
Question: could this computation being done by a linear number of flips in all cases ?

Application

There exists array memories for which it is better to have less than $\lceil \frac{n}{2} \rceil$ ones per line (resp. row).

What is an elementary operation for a graph ?

- Traversing an edge or Visiting the neighbourhood ?
- It explains the very few lower bounds known for graph algorithms on a RAM Machine.
- Our graph algorithms must accept any auto-complemented representation.
  Partition Refinement

Exercise

Suppose that a graph G is given by its adjacency lists and let σ be some total ordering of its vertices.

- How can we sort the adjacency with respect to σ (increasing) ?
- What is the complexity of this operation ?
- Let σ be the total ordering of the vertices with decreasing degrees, how to compute σ ?

Sorting an adjacency list

Suppose that a graph G is given by its adjacency lists A and let σ be some total ordering of its vertices. How can we sort the adjacency with respect to σ (increasing) ?

One solution

Build another adjacency list structure B from the old one by taking the vertices from $\sigma(n)$ down to $\sigma(1)$ in the following way:
read $A(\sigma(i))$ and add $\sigma(i)$ in front of the lists of B corresponding to the neighbors of $\sigma(i)$
At the end of the process, the lists of B are sorted in the right way.

Complexity

Time : Linear time complexity (the size of the data structure A).
Memory: Twice the size of the adjacency lists $2|A|$

Exercise

Adapt the solution for directed graphs

Sorting the vertices by their degrees

1. Compute the degrees by scanning the adjacency lists in an array D
2. Use any per value sorting algorithm since all encoding numbers of vertices are bounded by $log_2(n)$ for a simple graph, to sort the vertices by decreasing degrees. This sorting algorithm is well known to be linear time.
3. Apply the previous algorithm to sort the adjacency lists.

Remark: a second solution to sort the adjacency lists

- Just use any per value sorting algorithm to sort every adjacency lists.
- Can this be implemented linearly in the whole ?
- This method only uses $O(n)$ extra memory (the previous method uses $O(n+m)$).

The previous solutions need that the data structure is available at once in the memory.
This will be implicit in the remaining of the course, as well as the RAM model.

Quicksands

Efficient graph algorithms need to play with lists.
In several programming languages such as Java, lists are somehow hidden to the programmer.
They use a trick: represent a list via an array (or a vector), and when the array is full
recopy it into another array of size doubled.
From the complexity point of view, it is correct using the formula
$2^n - 1 = 1 + 2 + \cdots + 2^{n-1}$
So all the time spent to copy is $O(|L|)$ the maximal size of the list.

But if you program using reference (pointers) you could have problems after a doubling !

Some errors which look randomized.

Ordered graphs

In the John von Neumann architecture model of machine (1945), when a graph is inside the memory of the machine, the vertices are necessarily ordered, so as the adjacency lists.
For some graph algorithms it is very important to play with these orderings.
As for Multisweep algorithms.
Finally with our computers we play with ordered graphs.

As a perfect example

Succint representations of planar embedding of planar graphs with a permutation and an involution.

Neighbourhood as words

Then we can use a trie (tree data structure used to represent a set of words) to represent the graph.
This representation leads sometimes to very efficient algorithm (example computing false twins).
But this representation heavily depends on the initial ordering of the vertices.

Balls of radius r

The Vapnik-Chervonenkis dimension of this family of subsets of the vertex set, measure the complexity of the graph for some optimization problems (such as the diameter and many others)...
This Vapnik-Chervonenkis dimension was introduced in automatic learning theory.

Aigner and Ziegler,
Proofs from the BOOK
Springer-Verlag, 1999.

J.A. Bondy and U.S.R Murty,
Graph Theory,
Springer-Verlag, 2008.

R.E. Tarjan,
Data structures and network algorithms,
SIAM Monography, 44 (1983).