Graph theory

M1 - FIB

Contents:

Anna de Mier

Montserrat Maureso

Departament de Matemàtiques

Translation: Oriol Caño, Anna de Mier

February 2023

Chapter 1

Graphs: basic concepts

1. First definitions
2. Degrees
3. Graph isomorphism
4. Types of graphs
5. Subgraphs

1. First definitions

A graph G is a pair (V, E) where V is a non-empty finite set and E is a set of unordered pairs of different elements of V, that is, $E \subseteq \{\{u, v\} : u, v \in V\}$

We call

the elements of V, vertices

the elements of E, edges

the number of vertices, $|V|$, the order of G

the number of edges, $|E|$, the size of G

Let $u, v \in V$ be vertices and $a, e \in E$ be edges of G. We will say that:

u and v are adjacent or neighbours if $\{u, v\} \in E$, for short $u \sim v$ or $uv \in E$

u and e are incident if $e = \{u, w\}$, for some $w \in V$ $\quad$ e and a are incident if they have a vertex in common

u has degree $d(u)$ if the number of vertices adjacent to u is $d(u)$, that is,

$$d(u) = \#\{v \in V | u \sim v\}$$

Observation: If $n = |V|$ and $m = |E|$, then

$$0 \leq m \leq \frac{n(n-1)}{2} \text{ and } 0 \leq d(v) \leq n-1 \quad \text{for all} \quad v \in V$$

Drawing of a graph $G = (V, E)$

The vertices are represented by a dot and each edge by a curve joining the two dots corresponding to the vertices incident to that edge

Adjacency list (or adjacency table) of a graph $G = (V, E)$

Let $v_1, v_2, \dots, v_n$ be the vertices of G. The adjacency list of G is a list of length n where position i contains the set of vertices adjacent to v_i, for all $i \in [n]$

Let $G = (V, E)$ be a graph of order n and size m with $V = \{v_1, v_2, \dots, v_n\}$ and $E = \{a_1, a_2, \dots, a_m\}$

The adjacency matrix of G is the matrix $M_A = M_A(G)$ of type $n \times n$, such that the entry m_{ij} in the i-th row and j-th column is

$$m_{ij} = \begin{cases} 1, & \text{if } v_i \sim v_j \\ 0, & \text{otherwise} \end{cases}$$

– M_A is binary, with zeros in the diagonal, and symmetric
– The number of ones in the i-th row is the degree of v_i
– It is not unique, it depends on the ordering chosen for the set of vertices

Variations on the definition of graph:

- Multigraph: a graph that admits multiple edges, that is, there can be more than one edge joining two vertices
- Pseudograph: a graph that admits multiple edges and loops (edges that join a vertex with itself)
- Directed graph: a graph where the edges are oriented

2. Degrees

Let $G = (V, E)$ be a graph of order n and let $v \in V$ be a vertex. Define

the minimum degree of G, $\delta(G)$: the minimum of the degrees

the maximum degree of G, $\Delta(G)$: the maximum of the degrees

the degree sequence of G: the sequence of the degrees in decreasing order

a regular graph: a graph such that $\delta(G) = \Delta(G)$, i.e., all the vertices have the same degree

Observation:

– All graphs of order ≥ 2 have at least two vertices with the same degree

Handshaking lemma: $2|E| = \sum_{v \in V} d(v)$

Corollary All graphs have an even number of vertices with odd degree

A decreasing sequence of integers is graphical if there is some graph that has it as a degree sequence

3. Graph isomorphism

Let $G = (V, E)$ and $G' = (V', E')$ be two graphs. We will say that

▷ G and G' are equal, $G = G'$, if $V = V'$ and $E = E'$

▷ G and G' are isomorphic, $G \cong G'$, if there is a bijective map $f : V \to V'$, such that, for all $u, v \in V$,

$$u \sim v \Leftrightarrow f(u) \sim f(v).$$

The map f is called an isomorphism from G to G'

Remarks:

– A vertex and its image by an isomorphism have the same degree

– Two isomorphic graphs have the same size and order. The converse is false

– Two isomorphic graphs have the same degree sequence. The converse is false

– *Being isomorphic* is an equivalence relation

4. Types of graphs

Let n be a positive integer and $V = \{x_1, x_2, \dots, x_n\}$

Null graph of order n, N_n: is a graph with order n and size 0
Trivial graph: N_1

Complete graph of order n, K_n: is a graph of order n with all possible edges
– Size of $K_n = \frac{n(n-1)}{2}$

Path of order n, $P_n = (V, E)$: is a graph of order n and size $n - 1$ with set of edges $E = \{x_1x_2, x_2x_3, \dots, x_{n-1}x_n\}$
– $\delta(P_n) = 1$ and $\Delta(P_n) = 2$

Cycle of order n, $n \geq 3$, $C_n = (V, E)$, with $n \geq 3$: is a graph of order n and size n with set of edges $E = \{x_1x_2, x_2x_3, \dots, x_{n-1}x_n, x_nx_1\}$
– $\delta(C_n) = \Delta(C_n) = 2$

Wheel of order n, $n \geq 4$, $W_n = (V, E)$: is a graph of order n and size $2n - 2$ such that $E = \{x_1x_2, x_2x_3, \dots, x_{n-1}x_1\} \cup \{x_nx_1, x_nx_2, \dots, x_nx_{n-1}\}$

Let r and s be positive integers

An r-regular graph is a regular graph where r is the degree of the vertices

– The complete graph K_n is an $(n-1)$-regular graph
– The cycle graph C_n is a 2-regular graph
– If $G = (V, E)$ is an r-regular graph, then $2|E| = r|V|$

A bipartite graph is a graph $G = (V, E)$ such that there are two non-empty subsets V_1 and V_2 of V such that $V = V_1 \cup V_2$ and $V_1 \cap V_2 = \emptyset$, and for each edge $uv \in E$ we have that $u \in V_1$ and $v \in V_2$, or vice versa.
We call V_1 and V_2 the stable parts of the graph
– $\sum_{v \in V_1} g(v) = \sum_{v \in V_2} g(v) = |E|$

The complete bipartite graph $K_{r,s} = (V, E)$ is a bipartite graph with two stable parts V_1 and V_2 such that $|V_1| = r$ and $|V_2| = s$ and all the vertices of V_1 are adjacent to all the vertices of V_2. I.e., $E = \{uv | u \in V_1, v \in V_2\}$
– The order of $K_{r,s}$ is $r + s$ and the size is rs
– The graph $K_{1,s}$ is called star graph

5. Subgraphs

Let $G = (V, E)$ be a graph

Subgraph of G, $G' = (V', E')$: a graph with $V' \subseteq V$ and $E' \subseteq E$

Spanning subgraph of G, $G' = (V', E')$: a subgraph such that $V' = V$

Subgraph induced by $S \subseteq V$: the graph $G[S] = (S, E')$ such that $E' = \{uv \in E : u, v \in S\}$

5.1. Graphs derived from a graph

Let $G = (V, E)$ be a graph of order n and size m

Complement of G, $G^c = (V^c, E^c)$: is the graph with set of vertices $V^c = V$ and set of edges $E^c = \{uv | u, v \in V \text{ and } uv \notin E\}$
– The order of G^c equals the order of G
– The size of G^c is $\frac{n(n-1)}{2} - |E|$
– $(G^c)^c = G$
– Let H be a graph. Then $G \cong H \Leftrightarrow G^c \cong H^c$

The graph G is self-complementary if $G \cong G^c$

For $S \subset V$, the graph $G - S$ obtained by deleting the vertices from S is the graph with set of vertices $V \setminus S$ and whose edges are those that are not incident to any of the vertices of S. In the case that $S = \{v\}$, we denote it by $G - v$
– The order of $(G - u)$ is $n - 1$. The size of $(G - u)$ is $m - d(u)$

For $S \subset E$, the graph $G - S$ obtained by deleting the edges from S is the graph with set of vertices V and set of edges $E \setminus S$. In the case that $S = \{e\}$, we denote it by $G - e$

– The order and size of $G - e$ are n and $m - 1$, respectively

The graph $G + e$ obtained by adding an edge $e \in E$ is the graph with set of vertices V and set of edges $E' = E \cup \{e\}$

– The order of $G + e$ is n and its size is $m + 1$

5.2. Operations with graphs

Let $G = (V, E)$ and $G' = (V', E')$ be two graphs

Union of G and G', $G \cup G'$: graph with set of vertices $V \cup V'$ and set of edges $E \cup E'$

– If $V \cap V' = \emptyset$, the order of $G \cup G'$ is $|V| + |V'|$ and the size is $|E| + |E'|$

Product of G and G', $G \times G'$: graph with set of vertices $V \times V'$ and adjacencies given by

$$(u, u') \sim (v, v') \Leftrightarrow (uv \in E \text{ and } u' = v') \text{ or } (u = v \text{ and} u'v' \in E')$$

– The order of $G \times G'$ is $|V|\,|V'|$ and its size is $|V|\,|E'| + |V'|\,|E|$

Chapter 2

Walks, connectivity and distance

1. Walks
2. Connected graphs
3. Cut vertices and bridges
4. Distance
5. Characterization of bipartite graphs

1. Walks

Let $G = (V, E)$ be a graph, and let $u, v \in V$

A walk from u to v, or a u-v walk, of length k is a sequence of vertices and edges

$$\mathcal{R} : u_0 a_1 u_1 a_2 u_2 \dots u_{k-1} a_k u_k$$

such that $u_0 = u$, $u_k = v$ and $a_i = u_{i-1}u_i \in E$, for all $i \in [k]$. In general, we just write $u_0 u_1 u_2 \dots u_{k-1} u_k$

We say that the walk $\mathcal{R}$ visits the vertices u_i and visits the edges $a_i = u_{i-1}u_i$

If $u = v$ we say that it is a closed walk, and if $u \neq v$ we say that it is an open walk

A vertex is considered to be a walk of length zero

Kinds of walks: a u-v walk is a

– path if all vertices are different

– cycle if it is a closed walk of length ≥ 3 and all vertices are different

A vertex is considered to be a path of length zero

Remark A cycle visits two vertices u and v if, and only if, there are two u-v paths without any common vertex except u and v

A graph without cycles is called an acyclic graph

Proposition 1
Let $G = (V, E)$ be a graph and u, v different vertices. If there is in G a u-v walk of length k, then there is a u-v path of length $\leq k$

Proposition 2
Let $G = (V, E)$ be a graph and u, v different vertices. If G has two different u-v paths, then G has a cycle.

2. Connected graphs

We say that a graph $G = (G, E)$ is connected if for every pair of vertices u and v there is a u-v path. Otherwise we say that the graph is disconnected

Remark If $G = (V, E)$ is a connected graph of order greater than 1, then $d(v) \geq 1$, for all $v \in V$

We define the following relation $\mathbf{R}$ in V: for all $x, y \in V$

$$x\mathbf{R}y \Leftrightarrow \text{ there is an } x - y \text{ path in } G$$

$\mathbf{R}$ is an equivalence relation:
– Reflexive, $x\mathbf{R}x$: there is an x-x path of length zero
– Symmetric, if $x\mathbf{R}y$, then $y\mathbf{R}x$: an x-y path traversed in opposite direction is a y-x path
– Transitive, if $x\mathbf{R}y$ and $y\mathbf{R}z$, then $x\mathbf{R}z$: from an x-y path $xx_1 \dots x_n y$ and a y-z path $yy_1 \dots y_m z$, we build an x-v walk $xx_1 \dots x_n yy_1 \dots y_m z$, thus, there is an x-z path

If $G = (V, E)$ is a disconnected graph there is a partition of V in $k > 1$ subsets $V_1, V_2, \dots, V_k$, which are the equivalence classes of the relation **R**. Therefore, for all $1 \leq i, j \leq k$,

1. $V_i \neq \emptyset$, $V_i \cap V_j = \emptyset$ for all $i \neq j$, and $V = \bigcup_{i=1}^{k} V_i$
2. $G[V_i]$ (the subgraph induced by V_i) is connected
3. There is no path between vertices of $G[V_i]$ and vertices from $G[V_j]$, with $i \neq j$
4. $G = \bigcup_{i=1}^{k} G[V_i]$

The subgraphs $G[V_1], G[V_2], \dots, G[V_k]$ are called the connected components of G

Remark

Let $G = G_1 \cup G_2 \cup \cdots \cup G_k$, where G_i are the connected components of G. Then

$$\begin{aligned} \text{order } G &= \text{order } G_1 + \cdots + \text{order } G_k \\ \text{size } G &= \text{size } G_1 + \cdots + \text{size } G_k \end{aligned}$$

Proposition 3

A graph is 2-regular if, and only if, its connected components are cycles.

Proposition 4

Let $G = (V, E)$ be a connected graph and let $e = xy \in E$ and $u \in V$. Then

1. The graph $G - e$ has at most 2 connected components; if it has 2, vertex x belongs to one of them and vertex y to the other

2. The graph $G - u$ has at most $d(u)$ connected components

Proposition 5

Every connected graph of order n has at least $n - 1$ edges

2.1 Algorithm DFS: (Depth-first search)

```
DFS list(graph G, int v)
/* Pre: a graph G and a vertex v (assume that the vertices are integers)
/* Post: the list of vertices of G that belong to the same connected component as v
{
 Stack S;
 S.push(v);
 List W;
 W.add(v);
 int x;
 while (not S.is_empty) {
  x=S.top;
  if(''there is y adjacent to x that does not belong to W'') {
  S.push(y);
  W.add(y);
  }
  else {
  S.pop;
  }
 }
 return W;
}
```

Theorem 6 Let $G = (V, E)$ be a graph and v a vertex of G. The subgraph $G[W]$ induced by the vertices of G visited by the algorithm DFS is the connected component of G that contains v

3. Cut vertices and bridges

Let $G = (V, E)$ be a graph and let $v \in V$ and $e \in E$. We say that

– v is a cut vertex or articulation point if $G - v$ has more connected components than G

– e is a bridge if $G - e$ has more connected components than G

Remarks

1. If G is connected and u is a cut vertex, then $G - u$ is a disconnected graph with at most $d(u)$ connected components

2. The vertices of degree 1 are not cut vertices

3. If G is connected and e is a bridge, then $G - e$ is a disconnected graph with exactly 2 connected components

Theorem 7 Characterization of cut vertices

Let $G = (V, E)$ be a connected graph. A vertex u of G is a cut vertex if, and only if, there exists a pair of vertices x, y different from u such that every x-y path visits u

Theorem 8 Characterization of bridges

Let $G = (V, E)$ be a connected graph and $e = uv$ an edge of G. The following are equivalent:

(a) e is a bridge

(b) there exists a pair of vertices x, y such that every x-y path visits e

(c) no cycle contains e

Remarks

1. A graph may have cut vertices and no bridges
2. Let $e = uv$ be a bridge. If $d(u) = 1$, u is not a cut vertex; if $d(u) \geq 2$, the vertex u is a cut vertex
3. The only connected graph with a bridge and without cut vertices is K_2

4. Distance

Let $G = (V, E)$ be a graph and u, v vertices of G

– If u, v are in the same connected component, we define the distance between u and v, $d(u, v)$, as the minimum value among the lengths of all u-v paths. Otherwise we say that the distance is infinite

– The eccentricity of vertex u, $e(u)$, is the maximum distance between u and any other vertex of G, that is, $e(u) = \max\{d(u, v) | v \in V\}$

– The diameter of G, $D(G)$, is the maximum of the distances between the vertices of G, or, equivalently, the maximum of the eccentricities; that is, $D(G) = \max\{d(u, v) | u, v \in V\} = \max\{e(u) | u \in V\}$

– The radius of G, $r(G)$, is the minimum of the eccentricities of the vertices of G, that is, $r(G) = \min\{d(u, v) | u, v \in V\} = \max\{e(u) | u \in V\}$

– The central vertices of G, are the vertices whose eccentricities equal the radius, that is, the set $\{u \in V : e(u) = r(G)\}$

– The center of G is the subgraph induced by the central vertices

Remarks

1. $xy \in E \Leftrightarrow d(x, y) = 1$
2. G is not connected $\Leftrightarrow D(G) = \infty \Leftrightarrow r(G) = \infty \Leftrightarrow e(u) = \infty, \forall\ u \in V$
3. The following inequality holds $r(G) \leq D(G) \leq 2r(G)$

In a (connected) graph $G = (V, E)$, the following hold for all vertices u, v, z

1. $d(u, v) \geq 0$, and $d(u, v) = 0$ if, and only if, $u = v$
2. $d(u, v) = d(v, u)$
3. $d(u, v) + d(v, z) \geq d(u, z)$ (triangle inequality)

4.1 Algorithm BFS: (Breadth First Search)

```
vector BFS(graph G, int v)
/* Pre: a connected graph G of order n and a vertex v (assume that the vertices are integers)
/* Post: a vector D such that D[x]=d(v,x)
{
 Queue Q;
 Q.enqueue(v);
 List W;
 W.add(v);
 vector<int> D(n);
 D[v]=0;
 int x;
 while (not Q.is_empty) {
  x=Q.front;
  if(''there is y adjacent to x and y does not belong to W'') {
  Q.enqueue(y);
  W.add(y);
  D[y]=D[x]+1;
  }
  else {
  Q.advance;
  }
 }
 return D;
}
```

Theorem 9 Let $G = (V, E)$ be a graph and $v \in V$. The vector D given by the algorithm BFS stores the distance from vertex v to all other vertices in the graph

Chapter 3

Eulerian and Hamiltonian graphs

1. Eulerian graphs
2. Hamiltonian graphs

1. Eulerian graphs

A walk in a graph is called a trail if it is open and it does not repeat any edges, and it is called a circuit if it is closed, non-trivial, and does not repeat any edges.

Let G be a connected graph.

- An Eulerian trail is a trail that visits all the edges of G
- An Eulerian circuit is a circuit that visits all the edges of G
- An Eulerian graph is a graph that has an Eulerian circuit

Theorem Characterization of Eulerian graphs

Let G be a connected, non-trivial graph. Then,

G is Eulerian if, and only if, all its vertices have even degree

Corollary

A connected graph has an Eulerian trail if, and only if, it has exactly two vertices of odd degree

In that case, the Eulerian trail starts at a vertex of odd degree and finishes at the other vertex of odd degree

2. Hamiltonian graphs

Let G be a connected graph.

– A Hamiltonian path is a path that visits all the vertices of G

– A Hamiltonian cycle is a cycle that visits all the vertices of G

– A Hamiltonian graph is a graph that has a Hamiltonian cycle

Necessary conditions

Let $G = (V, E)$ be a Hamiltonian graph of order n, then

(1) $d(v) \geq 2$, for all $v \in V$

(2) if $S \subset V$ and $k = |S|$, the graph $G - S$ has at most k connected components

Sufficient conditions

Ore's Theorem Let $G = (V, E)$ be a graph of order $n \geq 3$ such that for all different and non adjacent $u, v \in V$ we have $d(u) + d(v) \geq n$. Then, G is a Hamiltonian graph

Dirac's Theorem Let $G = (V, E)$ be a graph of order $n \geq 3$ such that $d(u) \geq n/2$, for all $u \in V$. Then, G is Hamiltonian

Chapter 4

Trees

1. Trees and the characterization theorem
2. Spanning trees
3. Counting trees

1. Trees and the characterization theorem

– A tree is a connected acyclic graph
– A forest is an acyclic graph
– A leaf is a vertex of a tree or a forest that has degree 1

Observation: The connected components of a forest are trees

Remarks: Let $T = (V, E)$ be a tree, e an edge and u a vertex of T. Then

1. T contains at least one leaf
2. e is a bridge
3. $T - e$ is a forest with 2 connected components
4. if $d(u) \geq 2$, then u is a cut vertex
5. $T - u$ is a forest with $d(u)$ connected components
6. if u is a leaf, then $T - u$ is a tree

Proposition 1
All acyclic graphs of order n have size at most $n-1$.

Theorem 2 Characterization of trees
Let $T=(V,E)$ be a graph of order n and size m. The following are equivalent

(a) T is a tree
(b) T is acyclic and $m=n-1$
(c) T is connected and $m=n-1$
(d) T is connected and all edges are bridges
(e) for each pair of vertices u and v there is a unique u-v path in T
(f) T is acyclic and the addition of an edge creates exactly one cycle

Corollary 3
A forest G of order n with k connected components has size $n-k$

Corollary 4
If T is a tree of order $n \geq 2$, T has at least two leaves

2. Spanning trees

A spanning tree of a subgraph G is a spanning subgraph of G that is a tree

Theorem 5
A graph $G = (V, E)$ is connected if, and only if, G has a spanning tree

2.1 DFS algorithm to obtain spanning trees

```
DFS tree(graph G, int v)
/* Pre: a graph G and a vertex v
/* Post: a spanning tree of the connected component of G to which v belongs
{
 Stack S;
 S.push(v);
 List W;
 W.add(v);
 List B;
 int x;
 while (not S.is_empty) {
  x=S.top;
  if(''exists y adjacent to x that does not belong to W'') {
  S.push(y);
  W.add(y);
  B.add(xy);
  }
  else {
  S.pop;
  }
 }
 return (W,B);
}
```

Theorem 6

$T = (W, B)$ is a spanning tree of the connected component containing v

2.2 BFS algorithm to obtain spanning trees

```
BFS tree(graph G, int v)
/* Pre: a connected graph G of order n and a vertex v
/* Post: a spanning tree of the connected component of G to which v belongs
{
 Queue Q;
 Q.enqueue(v);
 List W;
 W.add(v);
 List B;
  int x;
 while (not Q.is_empty) {
  x=Q.peek;
  if(''exists y adjacent to x that does not belong to W'') {
  Q.enqueue(y);
  W.add(y);
  B.add(xy);
  }
  else {
  Q.dequeue;
  }
 }
 return (W,B);
}
```

Theorem 7

$T = (W, B)$ is a spanning tree of the connected component containing v

3. Counting trees

Cayley's formula
The number of different spanning trees of the complete graph K_n is n^{n-2}

The theorem is equivalent to saying that the number of different trees with set of vertices $[n]$ is n^{n-2}

The proof is based in the construction of a bijective map

$$Pr : \{T : T \text{ spanning tree of } K_n\} \longrightarrow [n]^{n-2},$$

The Prüfer's sequence of T is the image of T by the map Pr:

$$Pr(T) = (a_1, a_2, \cdots, a_{n-2})$$

- Construction of the Prüfer's sequence of a tree $T = ([n], E)$

Recursive construction

```
vector seqPrufer(tree T, int n)
/* Pre: a tree T with set of vertices {1,2,...,n}
/* Post: a vector of length n-2 containing the Prüfer's sequence of T

{
 tree Taux=T;
 int k=0;
 int leaf;
 vector<int> seq(n)
 while(k < n-2) {
  leaf=''leaf of Taux with the smallest label'';
  seq[k]=''vertex adjacent to leaf'';
  Taux=Taux-leaf;
  k++;
}
return seq;
}
```

Comments:

Let $b_1, \dots, b_{n-2}$ be the vertices of T that at some point in the execution have been a leaf

- `Taux` is a tree at each step of the algorithm
- the vertices $b_1, \dots, b_{n-2}$ are pairwise different
- $T - \{b_1, \dots, b_{n-2}\} \simeq K_2$
- n is one of the vertices of $T - \{b_1, \dots, b_{n-2}\}$
- $x \in [n]$ appears in the Prüfer's sequence as many times as $d(x) - 1$
- the vertices that do not appear in the Prüfer's sequence are leaves of T

• Reconstruction of the tree T from a word $(a_1, \ldots, a_{n-2})$ in the alphabet $[n]$. I.e., the inverse map of Pr

```
tree PruferTree(vector<int> seq, int n)
/* Pre: a vector with n-2 integers between 1 and n
/* Post: the tree that has seq as Prüfer's sequence

{
 List A;
 vector<int> leaves(n-1);
 leaves[0]=min([n]-{seq[0],seq[1],...,seq[n-3]});
 A.add({seq[0],leaves[0]});
 int k=1;
 while(k < n-2) {
  leaves[k]=min([n]-{seq[k],seq[k+1],...,seq[n-3],leaves[0],...,leaves[k-1]});
  A.add({seq[k],leaves[k]});
  k++;
  }
 leaves[n-2]=min([n]-{leaves[0],...,leaves[n-3]});
 A.add({leaves[n-2],n});
 return ([n],A);
}
```