Week 5

Elementary Graph Algorithms

General remarks

- We consider **graphs**, an important data structure.
- We discuss their representation (on the computer).

Reading from CLRS for week 4

- Chapter 22, Sections 22.1.
- Plus appendices

  B.4 "Graphs"

Towards the mathematical notion of "graph"

A "graph" consists of "vertices" and "edges".
An edge connects two vertices, without direction.

An edge connecting vertices u, v is denoted by $\{ u, v \}$.

The precise **Definition** is:

A **graph** G *is a pair* $G = (V, E)$, *where:*

- V *is the set of* **vertices** *(an arbitrary set);*
- E *is the set of (undirected)* **edges**;
- *an edge is a two-element subset of* V.

For an edge $\{ u, v \}$ we thus require $u \neq v$.

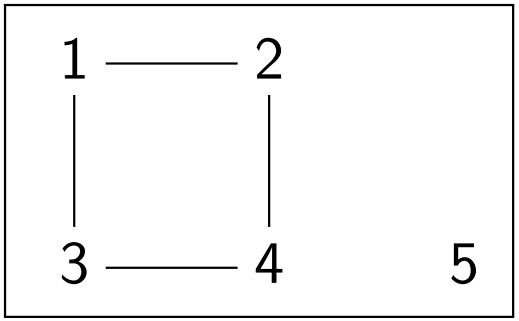
(This definition is the same as the basic definition on Wikipedia.)

Towards the mathematical notion of "graph" (cont.)

An example is

$$G = \big(\{\,1,2,3,4,5\,\},\{\,\{\,1,2\,\},\{\,1,3\,\},\{\,2,4\,\},\{\,3,4\,\}\,\}\big)$$

which is a graph with $|V| = 5$ vertices and $|E| = 4$ edges. A possible **drawing** is as follows:



We see that G has two connected components.

Definition: Connected

Definition 1

A graph $G = (V, E)$ is **connected** if for all vertices $v, w \in V$ there is a path in G (via the edges of G) from v to w. Otherwise G is **disconnected**, and has at least two connected components.

Definition 2

A **connected component** of a graph $G = (V, E)$ is a subset $V' \subseteq V$ of vertices, such that each pair of vertices $v, w \in V'$ is connected in G, while this property is destroyed when adding any other vertex to V'.

Social networks typically have "islands of connectivity".

A simple example of a graph with three connected components:

$$(\{1, 2, 3, 4, 5, 6, 7, 8\}, \{\{1, 2\}, \{8, 2\}, \{3, 4\}, \{5, 6\}, \{5, 7\}\}).$$

The connected components are $\{1, 2, 8\}, \{3, 4\}, \{5, 6, 7\}$.

Directed graphs

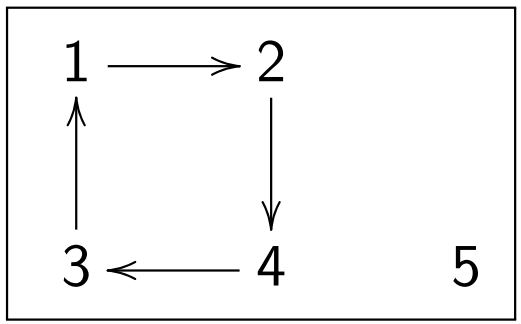
A "graph" is synonymous with "undirected graph". Now for a "directed graph" the edges are directed:

Definition *A* **directed graph** *(or* **digraph***) G is a pair $G = (V, E)$, where V is again the set of vertices, while E is the set of* **directed edges** *or* **arcs***. A directed edge from vertex u to vertex v is a pair, denoted by (u, v). Again we require $u \neq v$.*

An example is

$$G = \big(\{\,1, 2, 3, 4, 5\,\}, \{\,(1, 2), (3, 1), (2, 4), (4, 3)\,\}\big)$$

which is a graph with $|V| = 5$ vertices and $|E| = 4$ edges. A possible **drawing** is as follows:



Remarks on these fundamental notions

A **loop** is an edge or arc connecting a vertex with itself:

1. The notion of graph doesn't allow loops; only the extension of "graphs with loops" allows them, but we do not consider them here.
2. Also the notion of directed graph doesn't allow loops (different from the book we keep the symmetry between "graphs" and "directed graphs"); the extension of "directed graphs with loops" allows them, but again we do not consider them.

For a graph G with $n = |V|$ vertices we have

$$|E| \leq \binom{n}{2} = \frac{n(n-1)}{2}$$

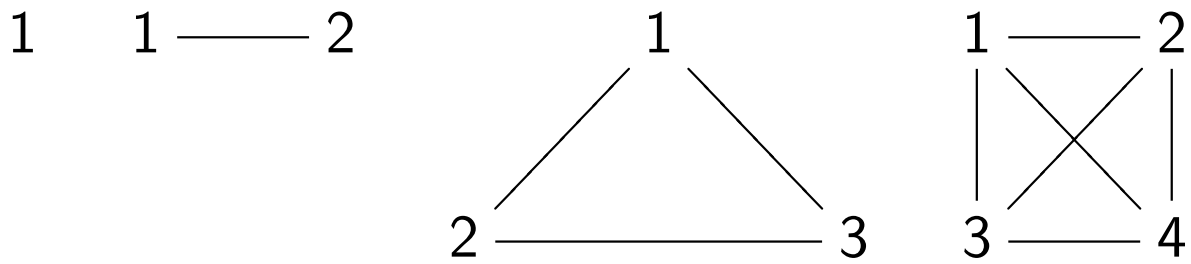
while for a directed graph G we have

$$|E| \leq n(n-1).$$

Small concrete examples

The complete graphs K_n with $n = 0, 1, 2, 3, 4$ vertices and their drawings (these graphs are all connected):

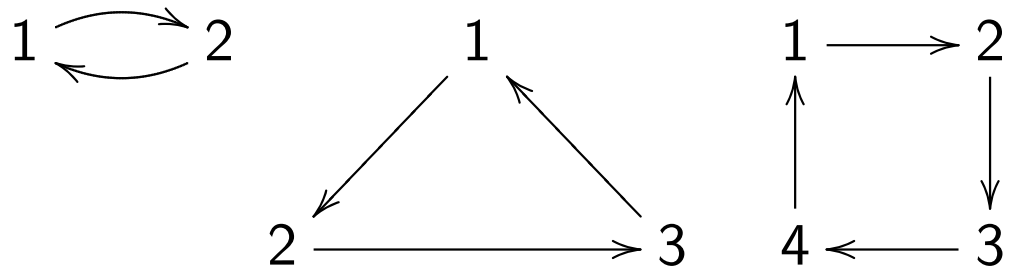
$$
\begin{aligned}
K_0 &= (\emptyset, \emptyset) \\
K_1 &= (\{1\}, \emptyset) \\
K_2 &= (\{1,2\}, \{\{1,2\}\}) \\
K_3 &= (\{1,2,3\}, \{\{1,2\},\{1,3\},\{2,3\}\}) \\
K_4 &= (\{1,2,3,4\}, \\
&\quad \{\{1,2\},\{1,3\},\{1,4\},\{2,3\},\{2,4\},\{3,4\}\})
\end{aligned}
$$



Small concrete examples (cont.)

The directed cycles C_n for $n = 2, 3, 4$:

$$
\begin{aligned}
C_2 &= (\{1, 2\}, \{(1, 2), (2, 1)\}) \\
C_3 &= (\{1, 2, 3\}, \{(1, 2), (2, 3), (3, 1)\}) \\
C_4 &= (\{1, 2, 3, 4\}, \{(1, 2), (2, 3), (3, 4), (4, 1)\})
\end{aligned}
$$



Applications in computer science

Graphs in algorithms are very versatile:

At least implicitly:
going through data, from node to node via edges,
occurs nearly everywhere.

An explicit use is for the modelling of logical implications:

$$A \longrightarrow B$$

can be directly modelled as arcs.

Applications in the sciences

Social network analysis is a trendy application:

> *Network science is an academic field which studies complex networks such as telecommunication networks, computer networks, biological networks, cognitive and semantic networks, and social networks, considering distinct elements or actors represented by nodes (or vertices) and the connections between the elements or actors as links (or edges).*

Representations of graphs

The above notions of graphs and digraphs yield mathematical objects. These are the eternal platonic ideas, which have many different representations in computers.

There are two main ways to represent a graph $G = (V, E)$:

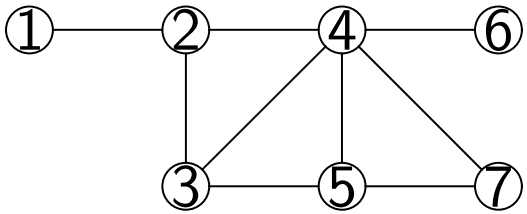
Adjacency lists: The graph is represented by a V-indexed array of linked lists, with each list containing the *neighbours* of a single vertex.

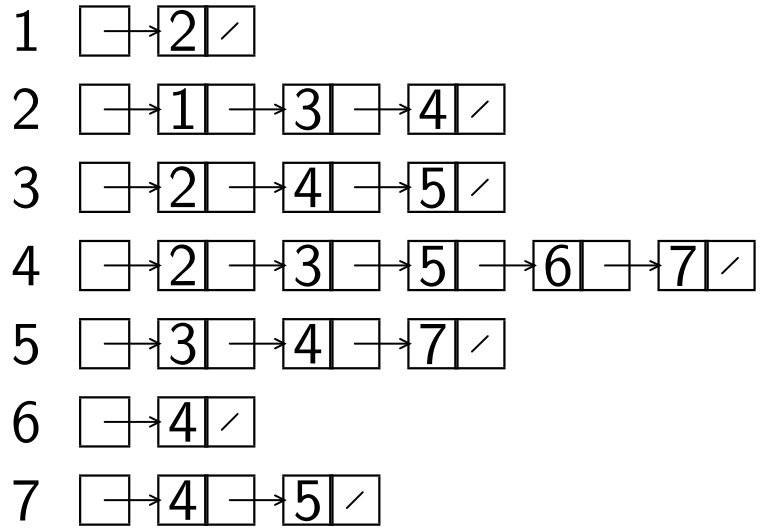
Adjacency matrix: The graph is represented by a $|V| \times |V|$ bit matrix A where $A_{i,j} = 1$ if and only if vertex i is *adjacent* to vertex j.

Two concepts here need definition for a graph G:

- The *neighbours* of a vertex $v \in V(G)$ are the vertices $w \in V(G)$ with $\{ v, w \} \in E(G)$. Since graphs are undirected, v is a neighbour of w iff w is a neighbour of v.
- Vertices v, w are *adjacent* if one is a neighbour of the other.

Adjacency list representation





- For (undirected) graphs this representation uses two list elements for each edge (since both directions are present).
- So the above graph has 9 edges, and there are $1 + 3 + 3 + 5 + 3 + 1 + 2 = 18 = 2 \cdot 9$ list elements.
- The total space required for this representation is $\mathrm{O}(V + E)$ (that is, $\mathrm{O}(|V| + |E|)$).
- This representation is especially suited to *sparse* graphs, where E is much less than V^2.

Remarks on the adjacency list representation

Note that this representation orders the neighbours in an arbitrary way.

- The graph itself is not affected by this, since graphs use vertex- and edge-**sets**.
- However algorithms using the data structure will be affected in general.
- Algorithms operating on graphs often compute output, which is **not unique**.

  The specification of computational problems often use **relations** between inputs and (many) possible outputs.
- For example they compute "spanning trees" for connected graphs, and a connected graph in general has many spanning trees.
- The choice, which possible output is actually computed, is affected by the choices made in the representation.

Adjacency lists for digraphs

In a digraph G one does not speak of (unqualified) "neighbours" of a vertex $v \in V(G)$ anymore, but:

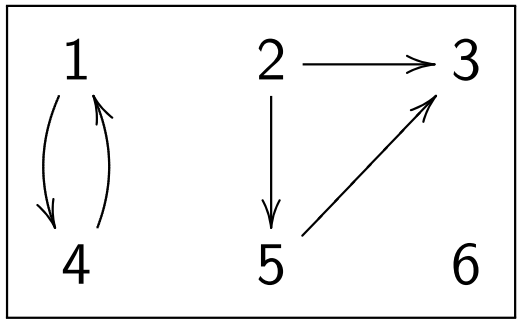
- the **out-neighbours** of v are the vertices $w \in V(G)$ with $(v, w) \in E(G)$;
- the **in-neighbours** of v are the vertices $w \in V(G)$ with $(w, v) \in E(G)$.

The adjacency lists for digraphs only contain the out-neighbours.

Thus in a digraph the total number of list elements equals the number of arcs.

That's natural: there is "less connectivity" in a digraph compared to a graph.

Example for adjacency lists for digraphs

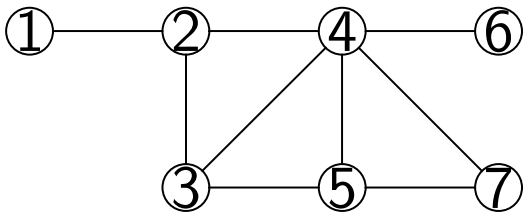


Showing the adjacency lists as tuples:

1: (4)
2: (5, 3)
3: ()
4: (1)
5: (3)
6: ().

Note that here we have exactly two possible choices, namely for vertex 2 we can either use $(5, 3)$ or $(3, 5)$.

Adjacency matrix representation



$$A = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \end{pmatrix}$$

- For (undirected) graphs the matrix is *symmetric*.
- The total space required for this representation is $\mathrm{O}(V^2)$.
- This representation is suited to *dense* graphs, where E is on the order of V^2.

Note that here another choice in general is involved:

The vertices must be given numbers $1, \ldots, |V|$.

Adjacency matrices for digraphs

Representing a digraph G, one needs to break the symmetry between rows and columns.

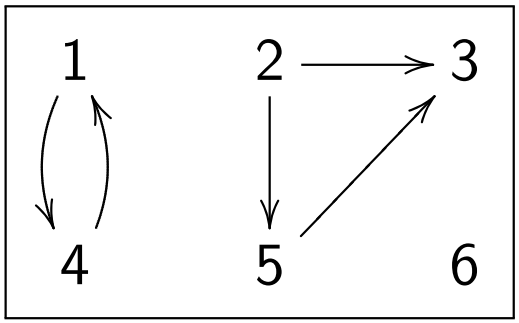
Again we need to give indices $1, \ldots, |V|$ to the vertices of G.

Now for the cell in row i and column j
the entry is 1 iff there is an arc from i to j,
and 0 otherwise.

Thus

- the rows show the out-neighbours,
- the columns contain the in-neighbours.

The old digraph as new adjacency matrix



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

- The diagonal is zero (since no loops).
- The anti-parallel arcs between vertices $1, 4$ correspond to the matrix having at position $(1, 4)$ *as well as* $(4, 1)$ value 1.
- Otherwise there are no such pairs.
- The number of 1's in a row is the **out-degree** of the vertex.
- The number of 1's in a column is the **in-degree**.

Adjacency matrices are very important
to link (di)graphs to *linear algebra*.

Comparing the two representations

Apart from the different space requirements needed for sparse and dense graphs, there are other criteria for deciding on which representation to use.

The choice of representation will depend mostly on what questions are being asked.

For example, consider the time required to decide the following questions using the two different representations:

- Is vertex v adjacent to vertex w in an undirected graph?
- What is the out-degree of a vertex v in a directed graph?
- What is the in-degree of a vertex v in a directed graph?

Comparing the two representations (cont.)

The answers (for time-complexity) are as follows:

- Deciding whether vertices v, w are adjacent (connected by an edge) in a graph:
  1. linear in the vertex degree (number of neighbours) for adjacency-list representation;
  2. constant time for adjacency-matrix representation.
- Determining the out-degree in a digraph:
  1. linear in the vertex out-degree for adjacency-list representation;
  2. linear in number of vertices for adjacency-matrix representation.
- Determining the in-degree in a digraph:
  1. linear in the size of the digraph (number of vertices + number of edges) for adjacency-list representation;
  2. linear in number of vertices for adjacency-matrix representation.

The formal notion of a graph

Are the following objects graphs or not?

1. 0

The formal notion of a graph

Are the following objects graphs or not?

1. 0 NO (not a pair)
2. $(0, 0)$

The formal notion of a graph

Are the following objects graphs or not?

1. 0 NO (not a pair)
2. $(0, 0)$ NO (not a pair of sets)
3. $(\{\ \}, \{\ \})$

The formal notion of a graph

Are the following objects graphs or not?

1. 0 NO (not a pair)
2. $(0, 0)$ NO (not a pair of sets)
3. $(\{\,\}, \{\,\})$ YES
4. $(\{1, 2\}, \{2\})$

The formal notion of a graph

Are the following objects graphs or not?

1. 0 NO (not a pair)
2. $(0, 0)$ NO (not a pair of sets)
3. $(\{\,\}, \{\,\})$ YES
4. $(\{1, 2\}, \{2\})$ NO (edge-set must contain edges)
5. $(\{1, 2\}, \{1, 2\})$

The formal notion of a graph

Are the following objects graphs or not?

1. 0 NO (not a pair)
2. $(0, 0)$ NO (not a pair of sets)
3. $(\{\,\}, \{\,\})$ YES
4. $(\{1, 2\}, \{2\})$ NO (edge-set must contain edges)
5. $(\{1, 2\}, \{1, 2\})$ NO (edge-set must contain edges)
6. $(\{1, 2\}, \{\{1, 2\}\})$

The formal notion of a graph

Are the following objects graphs or not?

1. 0 NO (not a pair)
2. $(0, 0)$ NO (not a pair of sets)
3. $(\{\,\}, \{\,\})$ YES
4. $(\{1, 2\}, \{2\})$ NO (edge-set must contain edges)
5. $(\{1, 2\}, \{1, 2\})$ NO (edge-set must contain edges)
6. $(\{1, 2\}, \{\{1, 2\}\})$ YES
7. $(\{1, 2\}, \{(1, 2)\})$

The formal notion of a graph

Are the following objects graphs or not?

1. 0 NO (not a pair)
2. $(0, 0)$ NO (not a pair of sets)
3. $(\{\,\}, \{\,\})$ YES
4. $(\{1, 2\}, \{2\})$ NO (edge-set must contain edges)
5. $(\{1, 2\}, \{1, 2\})$ NO (edge-set must contain edges)
6. $(\{1, 2\}, \{\{1, 2\}\})$ YES
7. $(\{1, 2\}, \{(1, 2)\})$ NO (edge-set must contain edges)
8. $(\{1, 2\}, \{\{1, 3\}\})$

The formal notion of a graph

Are the following objects graphs or not?

1. 0 NO (not a pair)
2. $(0, 0)$ NO (not a pair of sets)
3. $(\{\,\}, \{\,\})$ YES
4. $(\{1, 2\}, \{2\})$ NO (edge-set must contain edges)
5. $(\{1, 2\}, \{1, 2\})$ NO (edge-set must contain edges)
6. $(\{1, 2\}, \{\{1, 2\}\})$ YES
7. $(\{1, 2\}, \{(1, 2)\})$ NO (edge-set must contain edges)
8. $(\{1, 2\}, \{\{1, 3\}\})$ NO (only vertices from the vertex-set)
9. $(\{1, 2, 3\}, \{\{1, 2, 3\}\})$

The formal notion of a graph

Are the following objects graphs or not?

1. 0 NO (not a pair)
2. $(0, 0)$ NO (not a pair of sets)
3. $(\{\,\}, \{\,\})$ YES
4. $(\{1, 2\}, \{2\})$ NO (edge-set must contain edges)
5. $(\{1, 2\}, \{1, 2\})$ NO (edge-set must contain edges)
6. $(\{1, 2\}, \{\{1, 2\}\})$ YES
7. $(\{1, 2\}, \{(1, 2)\})$ NO (edge-set must contain edges)
8. $(\{1, 2\}, \{\{1, 3\}\})$ NO (only vertices from the vertex-set)
9. $(\{1, 2, 3\}, \{\{1, 2, 3\}\})$ NO (edges connect two vertices)
10. $(\{1, 2, 3, 3\}, \{\{1, 3\}\})$

The formal notion of a graph

Are the following objects graphs or not?

1. 0 NO (not a pair)
2. $(0, 0)$ NO (not a pair of sets)
3. $(\{\,\}, \{\,\})$ YES
4. $(\{1, 2\}, \{2\})$ NO (edge-set must contain edges)
5. $(\{1, 2\}, \{1, 2\})$ NO (edge-set must contain edges)
6. $(\{1, 2\}, \{\{1, 2\}\})$ YES
7. $(\{1, 2\}, \{(1, 2)\})$ NO (edge-set must contain edges)
8. $(\{1, 2\}, \{\{1, 3\}\})$ NO (only vertices from the vertex-set)
9. $(\{1, 2, 3\}, \{\{1, 2, 3\}\})$ NO (edges connect two vertices)
10. $(\{1, 2, 3, 3\}, \{\{1, 3\}\})$ YES
11. $(\{1, 2, 3\}, \{\{1, 1\}\})$

The formal notion of a graph

Are the following objects graphs or not?

1. 0 NO (not a pair)
2. $(0, 0)$ NO (not a pair of sets)
3. $(\{\,\}, \{\,\})$ YES
4. $(\{1, 2\}, \{2\})$ NO (edge-set must contain edges)
5. $(\{1, 2\}, \{1, 2\})$ NO (edge-set must contain edges)
6. $(\{1, 2\}, \{\{1, 2\}\})$ YES
7. $(\{1, 2\}, \{(1, 2)\})$ NO (edge-set must contain edges)
8. $(\{1, 2\}, \{\{1, 3\}\})$ NO (only vertices from the vertex-set)
9. $(\{1, 2, 3\}, \{\{1, 2, 3\}\})$ NO (edges connect two vertices)
10. $(\{1, 2, 3, 3\}, \{\{1, 3\}\})$ YES
11. $(\{1, 2, 3\}, \{\{1, 1\}\})$ NO (edges are two-element-sets)
12. $(\{a, b, c\}, \{\{a, b\}, \{b, c\}\})$

The formal notion of a graph

Are the following objects graphs or not?

1. 0 NO (not a pair)
2. $(0, 0)$ NO (not a pair of sets)
3. $(\{\,\}, \{\,\})$ YES
4. $(\{1, 2\}, \{2\})$ NO (edge-set must contain edges)
5. $(\{1, 2\}, \{1, 2\})$ NO (edge-set must contain edges)
6. $(\{1, 2\}, \{\{1, 2\}\})$ YES
7. $(\{1, 2\}, \{(1, 2)\})$ NO (edge-set must contain edges)
8. $(\{1, 2\}, \{\{1, 3\}\})$ NO (only vertices from the vertex-set)
9. $(\{1, 2, 3\}, \{\{1, 2, 3\}\})$ NO (edges connect two vertices)
10. $(\{1, 2, 3, 3\}, \{\{1, 3\}\})$ YES
11. $(\{1, 2, 3\}, \{\{1, 1\}\})$ NO (edges are two-element-sets)
12. $(\{a, b, c\}, \{\{a, b\}, \{b, c\}\})$ YES

Tricky question: What if in the last examples, a, b, c are not considered as letters, but as *variables*?

The formal notion of a digraph

Are the following objects digraphs or not?

1. $(\{1, 3\}, \emptyset)$

The formal notion of a digraph

Are the following objects digraphs or not?

1. $(\{1,3\}, \emptyset)$ YES
2. $(\{2,4\}, \{(4,2)\})$

The formal notion of a digraph

Are the following objects digraphs or not?

1. $(\{1,3\}, \emptyset)$ YES
2. $(\{2,4\}, \{(4,2)\})$ YES
3. $(\{1,2\}, \{(1,2),(2,1)\})$

The formal notion of a digraph

Are the following objects digraphs or not?

1. $(\{1,3\}, \emptyset)$ YES
2. $(\{2,4\}, \{(4,2)\})$ YES
3. $(\{1,2\}, \{(1,2),(2,1)\})$ YES
4. $(\{4\}, \{(4,4)\})$

The formal notion of a digraph

Are the following objects digraphs or not?

1. $(\{1,3\}, \emptyset)$ YES
2. $(\{2,4\}, \{(4,2)\})$ YES
3. $(\{1,2\}, \{(1,2),(2,1)\})$ YES
4. $(\{4\}, \{(4,4)\})$ NO (no loops)
5. $(\{1,2,3\}, \{(1,2), \{2,3\}\})$

The formal notion of a digraph

Are the following objects digraphs or not?

1. $(\{1,3\}, \emptyset)$ YES
2. $(\{2,4\}, \{(4,2)\})$ YES
3. $(\{1,2\}, \{(1,2),(2,1)\})$ YES
4. $(\{4\}, \{(4,4)\})$ NO (no loops)
5. $(\{1,2,3\}, \{(1,2), \{2,3\}\})$ NO (only directed edges)
6. $(\{1,2,3\}, \{\})$

The formal notion of a digraph

Are the following objects digraphs or not?

1. $(\{1,3\}, \emptyset)$ YES
2. $(\{2,4\}, \{(4,2)\})$ YES
3. $(\{1,2\}, \{(1,2),(2,1)\})$ YES
4. $(\{4\}, \{(4,4)\})$ NO (no loops)
5. $(\{1,2,3\}, \{(1,2), \{2,3\}\})$ NO (only directed edges)
6. $(\{1,2,3\}, \{\})$ YES.

What is a graph, formally, short?:

The formal notion of a digraph

Are the following objects digraphs or not?

1. $(\{1,3\}, \emptyset)$ YES
2. $(\{2,4\}, \{(4,2)\})$ YES
3. $(\{1,2\}, \{(1,2),(2,1)\})$ YES
4. $(\{4\}, \{(4,4)\})$ NO (no loops)
5. $(\{1,2,3\}, \{(1,2), \{2,3\}\})$ NO (only directed edges)
6. $(\{1,2,3\}, \{\})$ YES.

What is a graph, formally, short?:

> A pair, consisting of the vertex-set and the set of edges, which are two-elements subsets of the vertex-set.

What is a digraph, formally, short?:

The formal notion of a digraph

Are the following objects digraphs or not?

1. $(\{1,3\},\emptyset)$ YES
2. $(\{2,4\},\{(4,2)\})$ YES
3. $(\{1,2\},\{(1,2),(2,1)\})$ YES
4. $(\{4\},\{(4,4)\})$ NO (no loops)
5. $(\{1,2,3\},\{(1,2),\{2,3\}\})$ NO (only directed edges)
6. $(\{1,2,3\},\{\})$ YES.

What is a graph, formally, short?:

A pair, consisting of the vertex-set and the set of edges, which are two-elements subsets of the vertex-set.

What is a digraph, formally, short?:

A pair, consisting of the vertex-set and the set of arcs, which are pairs of different vertices.