

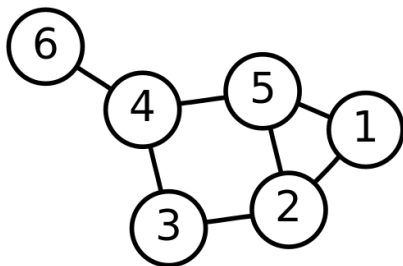
Table of Contents

Introduction to graphs

Graph representation

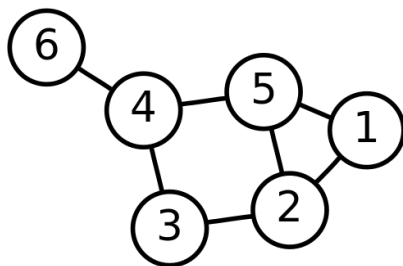
Graphs, intuitively

- ▶ Points (vertex, nodes)
- ▶ Lines between the points (edges, links)



Graphs, mathematically

Graph = (**V**ertices, **E**edges)

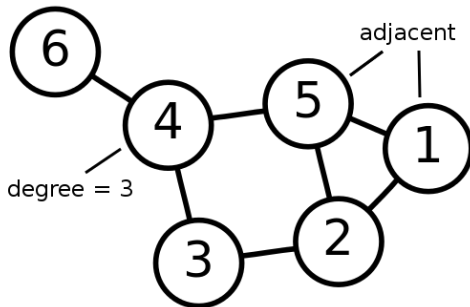


$$V = \{1, 2, 3, 4, 5, 6\}$$

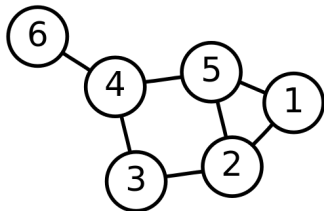
$$E = \{(1, 2), (1, 5), (2, 3), (2, 5), (3, 4), (4, 5), (4, 6)\}$$

Terminology

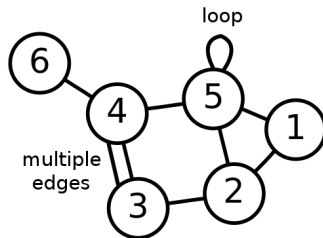
- ▶ u and v adjacent $\Leftrightarrow (u, v) \in E$
- ▶ Degree of $u = \#\{\text{edges from } u\}$



Simple graph vs multigraph

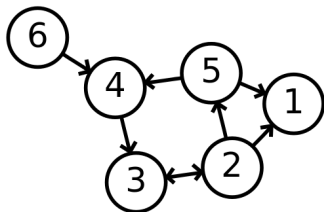


simple graph

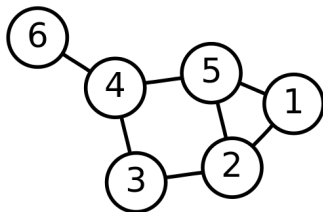


multigraph

Directed vs undirected



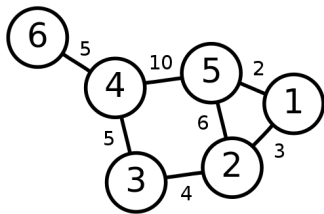
directed



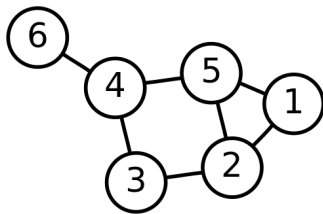
undirected

Weighted vs unweighted

- Represent costs, times, lengths, capacities



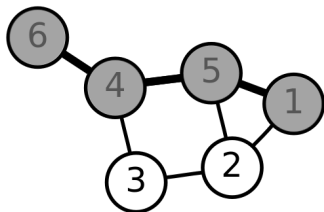
weighted



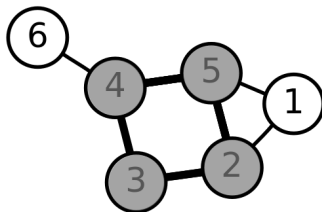
unweighted

Paths and cycles

- ▶ u and v connected \Leftrightarrow path $u \rightarrow v$
- ▶ G connected \Leftrightarrow all pairs connected



path



cycle

Trees

- ▶ Connected
- ▶ Acyclic
- ▶ $|E| = |V| - 1$

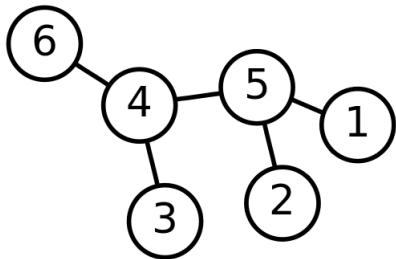


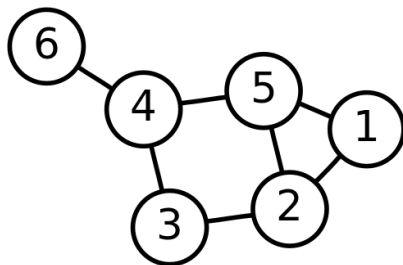
Table of Contents

Introduction to graphs

Graph representation

Edge list

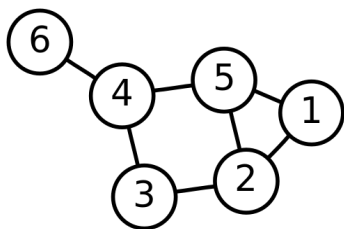
- ▶ Vector of pairs: `vector<pair<int,int>> edges`
- ▶ Same as input / mathematical definition
- ▶ Limited use (only Kruskal)



`edges[] = {(1, 2), (1, 5), (2, 3), (2, 5), (3, 4), (4, 5), (4, 6)}`

Adjacency matrix on undirected graph

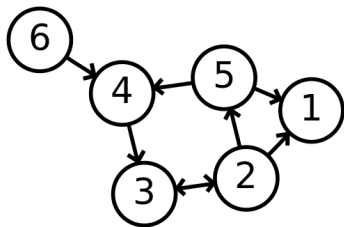
- ▶ Static two-dimensional array: `bool adj[MAXN][MAXN]`
- ▶ `adj[i][j] == true` if edge $i \rightarrow j$
- ▶ Symmetric



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

Adjacency matrix on directed graph

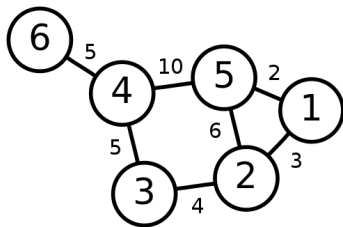
- ▶ Only put true in one direction
- ▶ Not symmetric



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

Adjacency matrix on weighted graph

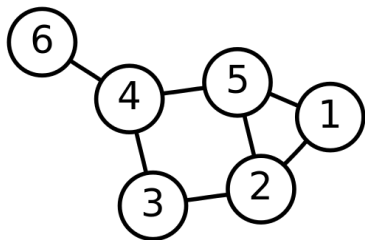
- ▶ Change the type: `int adj[MAXN][MAXN]`
- ▶ Instead of true put the weights



$$\begin{pmatrix} 0 & 3 & 0 & 0 & 2 & 0 \\ 3 & 0 & 4 & 0 & 6 & 0 \\ 0 & 4 & 0 & 5 & 0 & 0 \\ 0 & 0 & 5 & 0 & 10 & 5 \\ 2 & 6 & 0 & 10 & 0 & 0 \\ 0 & 0 & 0 & 5 & 0 & 0 \end{pmatrix}$$

Adjacency list on undirected graph

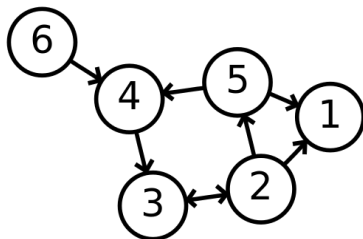
- ▶ Array of vectors: `vector<int> neigh[MAXN]`
- ▶ For each vertex, list the neighbors



1. {2, 5}
2. {1, 3, 5}
3. {2, 4}
4. {3, 5, 6}
5. {1, 2, 4}
6. {4}

Adjacency list on directed graph

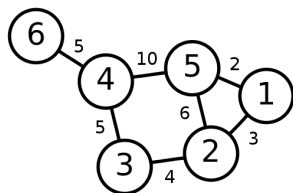
- Only list in one direction



1. $\{\}$
2. $\{1, 3, 5\}$
3. $\{2\}$
4. $\{3\}$
5. $\{1, 4\}$
6. $\{4\}$

Adjacency list on weighted graph

- Add the weight lists: `vector<int> weight[MAXN]`



`neigh[]`

1. {2, 5}

2. {1, 3, 5}

3. {2, 4}

4. {3, 5, 6}

5. {1, 2, 4}

6. {4}

`weight[]`

1. {3, 2}

2. {3, 4, 6}

3. {4, 5}

4. {5, 10, 5}

5. {2, 6, 10}

6. {5}

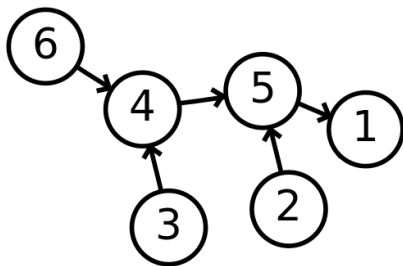
Comparison

- ▶ Use adjacency list (most of the time)
- ▶ Adjacency matrix only used for fast edge lookup

	Size	List all edges	Edge lookup
Matrix	$O(V^2)$	$O(V^2)$	$O(1)$
List	$O(V + E)$	$O(V + E)$	$O(\deg(u))$

Tree, parent representation

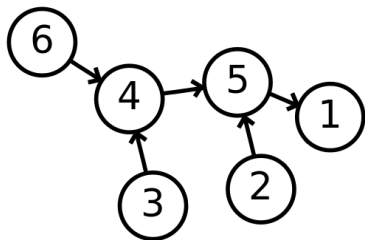
- ▶ Choose a root
- ▶ Every node has a parent (except the root)
- ▶ Parents lead to the root



`parent[] = {-, 5, 4, 5, 1, 4}`

Tree, child representation

- ▶ For each node, list the children
- ▶ Nodes 2, 3, 6 are leaves (no children)



1. {5}
2. {}
3. {}
4. {3, 6}
5. {2, 4}
6. {}

Source of figures

- ▶ <https://en.wikipedia.org/wiki/File:6n-graf.svg>