Graphs: $G = (V, E)$ V - Vertex set (Node set) E - Edge Set (Arcs, Links)

$E \subseteq V \times V$

Graphs
- Undirected ✓
- Directed ✓

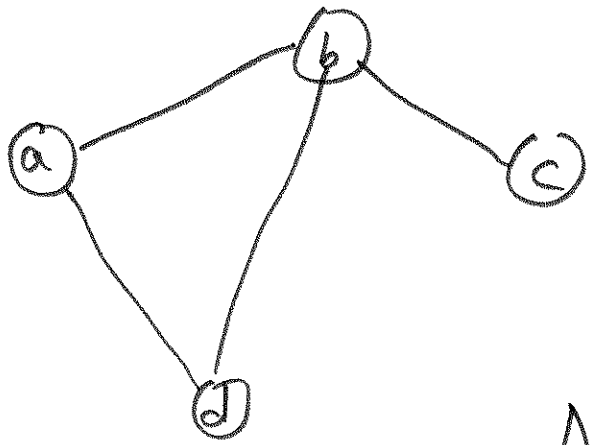
- Simple ✓
- ~~Complex~~

- General ✓
- Bipartite ✓ $G = (V, E)$

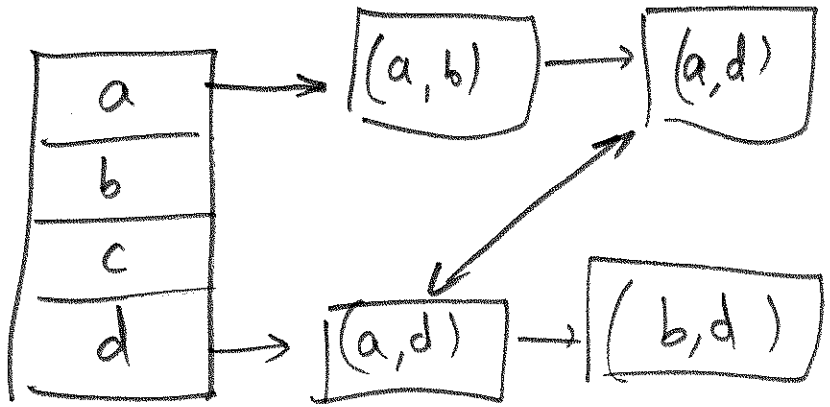
$V = X \cup Y$, $X \cap Y = \emptyset$

such that all edges $e = (u, v) \in E$

$u \in X$ and $v \in Y$.



Representations: Adjacency List representation.



Intersection of two sorted lists implementing sets:

Pseudocode:

Iterate through the lists L_1 and L_2 using iterators I_1 and I_2, respectively

while there are elements left to process
    Let x_1, x_2 be the current elements of I_1 and I_2, respectively
    If $x_1 < x_2$ then
        Advance I_1
    else if $x_1 > x_2$ then
        Advance I_2
    else
        Add x_1 to output List
        Advance I_1.
        Advance I_2

Return

Helper function: return
next(it) { (it.hasNext()) ? it.next() : null; }

Code (Symmetric solution) - Some type details missing.

```
it1 = l1.iterator();
it2 = l2.iterator();
x1 = next(it1); x2 = next(it2);
While (x1 != null && x2 != null) {
    cmp = x1.compareTo(x2);
    if (cmp < 0)
        x1 = next(it1);
    else if (cmp > 0) x2 = next(it2);
    else { outList.add(x1);
        x1 = next(it1); x2 = next(it2);
    }
}
```

Solution using Foreach loop for one list:

```
it1 = l1.iterator(); x1 = next(it1);
for (x2 : l2) {
    while (x1 != null && x1.compareTo(x2) < 0)
        x1 = next(it1);
    if (x1 == null) return;
    if (x1.compareTo(x2) == 0) {
        outList.add(x1);
        x1 = next(it1);
    }
```