# Informatics Concepts

## Strings

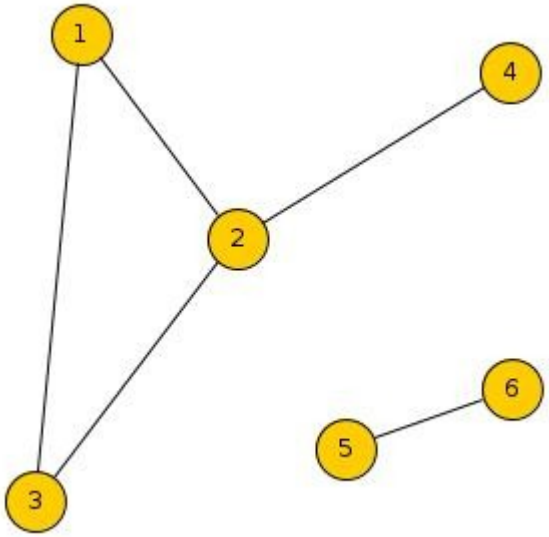
- s = AATGCA
- |s| = 6, s[1] = a, s[3] = t
- ϵ : empty string
- Subsequence, substring
- Supersequence, superstring
- Concatenation, prefix, suffix

# Informatics Concepts

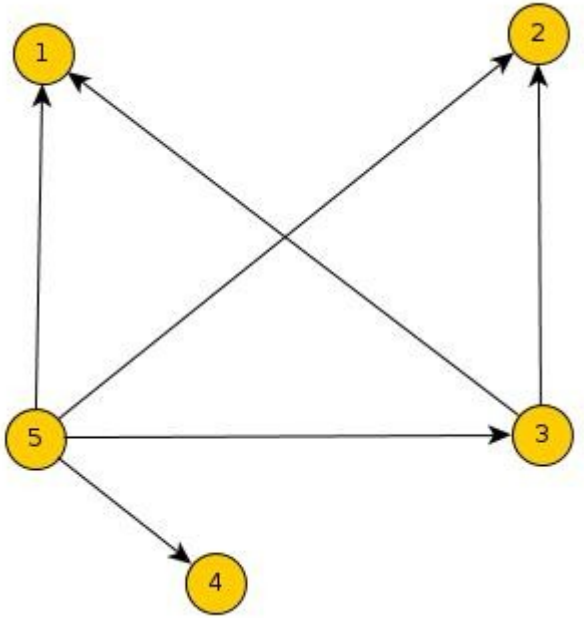
**Graphs**

- V, E
- Directed, undirected edges
- Simple graphs: no loops or multiple edges
- Incidence, adajcency
- Head ← tail
- Degree, outdegree, indegree

# Graphs




Undirected graph




Directed graph

# Graphs

- Weighted graphs (cost or distance)
- Subgraph, proper subgraph, spanning subgraph, induced subgraph
- Paths, cycles (simple), reachability
- Path weight
- Connected, disconnected graphs
- Connected components

# Graph classes

- Acyclic graphs (forests)
- Complete graphs (all possible edges present)
- Bipartite
- Trees (connected forests)
  - Leaves, internal nodes
  - Rooted tress, root
  - Parent, children
  - Ancestors, descendants
  - Lowest common ancestor

# Computational problems

- Eulerian graphs, Eulerian paths
- Hamiltonian graphs, Hamiltonian paths
- Minimum Spanning Trees (MSTs)
  - weighted, undirected graphs
- Vertex-coloring
- Maximum mathcing (weighted graphs)

# Graph representation

- Adajcency matrix
- Adjacency lists
- Sparse and dense graph families (infinite)

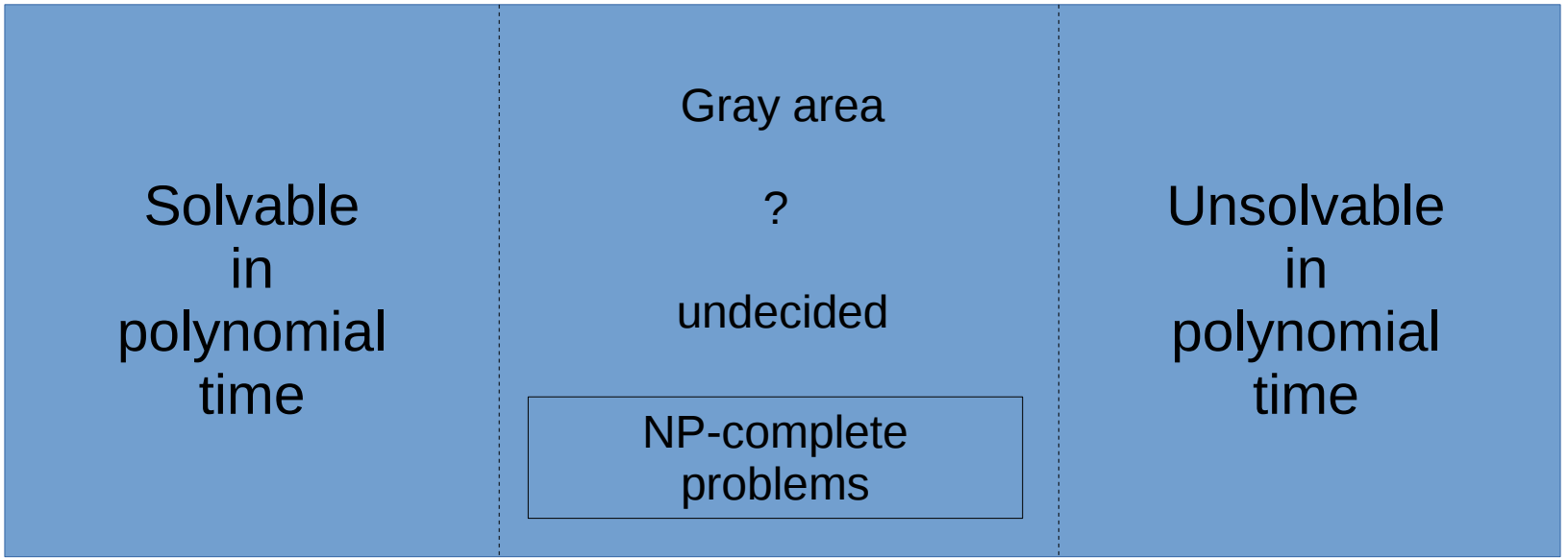
# Algorithms

- Random Access Machine (RAM)
- Memory, registers
- Program: informal instructions
  - **`input:`**
  - **`output:`**
  - `CONSTANT`
  - *variable*
  - **keyword** (**for, if, while,** etc.)

# Algorithms

- Correct algorithm
- Worst case running time
  - as a function of input size, *n*
- $f(n) = 7n^2 + 3n + 1 = O(n^2)$
- Linear time, quadratic time, polynomial time

# NP-completeness



# NP-completeness

- NP-complete vs. NP-hard
- Reductions: A < B
- Eulerian graph: P
- Hamiltonian graph: NP-hard
- MST: P
- Vertex coloring: NP-hard
- Maximum matching|: P

# NP-hard problem: What to do?

- Restrict input: e.g., vertex coloring is in P for bipartite graphs
- Brute-force: maybe your instance can be solved in reasonable time
- Approximation algorithms
- Heuristics

# Important algorithms

- Depth-first search (DFS): stack
- Breadth-first search (BFS): queue
- Sorting: O($n$ log $n$), sometimes O($n$)
- Greedy algorithms
- Dynamic programming

# Important data structures

- Stack
- Queue
- Hash table
- Binary search tree
- Disjoint forest (union-find)