

Decision Problems

Is there a (feasible) solution? yes or no?

Examples:

1) REACHABILITY (PATH): Given a (directed) graph G and two vertices u, v in G , is there a (directed) path between u and v ?

2) HAMILTONIAN PATH: Given a graph G , is there a simple path containing all vertices in G ?

Optimization Problems

Feasible solution with the best value.

Examples:

1) Given a graph G and two vertices u, v in G find a shortest path (with minimum number of edges) between u and v .

2) Given a graph G , find the longest simple path in G .

If the optimization problem is easy, its decision version is easy as well.

Usually, we can recast the optimization problem as a decision problem that is no harder.

REACHABILITY

Can be solved by *depth-first* search or *breadth-first* search in **polynomial** time.

If the graph is input as an adjacency matrix then the above algorithms take $O(n^2)$ time, where n is the number of vertices of the graph.

Computation Model: RAM

RAM - Random Access Machine:

1. A computing unit plus a memory.
2. Instructions are executed sequentially.
3. Accessing (reading/writing) any one memory cell takes one step.
4. Any “reasonable” one operation takes one step.

Asymptotic Notation

$g(n) = O(f(n))$ if there is a positive constant c (independent of n), such that for any $n \geq n_0$, $g(n) \leq cf(n)$. " g does not grow faster than f "

$g(n) = \Omega(f(n))$ if there is a positive constant c (independent of n), such that for any $n \geq n_0$, $g(n) \geq cf(n)$.

$g(n) = \Theta(f(n))$ if $g(n) = O(f(n))$ AND $g(n) = \Omega(f(n))$.

Examples

$$1) n^3 + 2^{1000}n^2 + 2^{100000}n = O(n^3) = \Omega(n^3) = \Theta(n^3)$$

2) In general for any polynomial

$p(n) = \sum_{i=0}^d a_i n^i$ where a_i are constants and $a_d > 0$, we have

$$p(n) = \Theta(n^d).$$

$$3) n = O(n^2)$$

$$4) n^{1000} = O(1.1^n)$$

5) In fact, if $p(n)$ is any polynomial, then

$$p(n) = O(c^n) \text{ and } p(n) \neq \Omega(c^n), \text{ for any } c > 1.$$

Polynomial time

An **abstract problem** is a binary relation on a set I of instances and a set S of solutions.

For example, REACHABILITY problem is a mapping from an instance $\langle G, u, v \rangle$ to 0 (no) or 1 (yes).

An algorithm takes an **encoding** of the problem instances as input.

A **binary** encoding of a problem is a mapping from the set I of instances to the set of binary strings.

Then, we say that a problem is polynomial-time solvable if there exists an algorithm to solve an instance of length n (in binary encoding) in time $O(n^k)$ for some constant k .

P is the set of decision problems which are polynomial-time solvable.

We will assume that all problem instances are binary encoded in some “standard” format.

Example: Hamiltonian Path

Given a graph G , does G have a Hamiltonian Path?

One possible algorithm is to check all possible permutations of the vertices.

A “reasonable encoding” of G is that of its adjacency matrix.

If n is the number of vertices of G , then the size of the encoding is $O(n^2)$, but the running time of the algorithm is $\Omega(n!) = \Omega(2^n)$ which is not a polynomial in the size of the encoding.

Polynomial-time Reductions

We want to solve a decision problem A in polynomial time.

Suppose we know how to solve a different decision problem B in polynomial time.

We can solve A using B if we have a polynomial-time procedure that transforms **any** instance x of A into some instance y of B such that the answer for x is “yes” if and only the answer for y is “yes”.

B is as hard as A .

Suppose A is a “hard” problem, say, it has no polynomial time algorithm.

Then the above reduction can be used to show that B has no polynomial time algorithm as well.

3-SAT and CLIQUE

A boolean formula is in 3-CNF form if it is in CNF form and each clause has exactly three literals per clause.

3-SAT: Given a 3-CNF formula F is there a satisfying assignment for F ?

CLIQUE: Given a graph G , does G contain a complete subgraph of size k ?

Theorem 1. *3-SAT is polynomial time reducible to CLIQUE.*

Proof

We give a polynomial time reduction from 3-SAT to CLIQUE.

Let ϕ be an instance of 3-SAT with k clauses:

$$\phi = (a_1 \vee b_1 \vee c_1) \wedge (a_2 \vee b_2 \vee c_2) \wedge \dots \wedge (a_k \vee b_k \vee c_k).$$

We construct an instance G as follows:

Each literal in each clause corresponds to a node in G . The nodes are organized into k groups of 3 nodes each called the triples.

There is an edge between all pairs of nodes except between:

1. two nodes in the same triple.
2. two nodes corresponding to opposite literals.

We show that ϕ is satisfiable iff G has a clique of size k .

If ϕ has a satisfying assignment, then choose one true literal in every clause. The corresponding nodes form a k -clique.

If G has a k -clique, then each of the k triples contains exactly one of the k clique nodes. Assign truth values such that each corresponding literal is made true.

Matching

Given a graph $G = (V, E)$ a set of edges M is a **matching** in G if

1. $M \subseteq E$;
2. no two edges of M share the same node;

A **maximum** matching in G is a matching with maximal cardinality.

A graph has a **perfect** matching if it has a matching of size $|V|/2$ (i.e. every vertex is covered by the matching).

Flow Network

A **flow network** is a directed graph $G = (V, E)$, and a capacity function

$$c : V \times V \rightarrow R$$

such that:

1. For each $(u, v) \in E$ the capacity $c(u, v) \geq 0$.
2. If $(u, v) \notin E$ then $c(u, v) = 0$.
3. There is a source s and a sink t .
4. Each vertex is on a directed path from s to t .

Flow

A flow in a flow network G with capacity function c is a function $f : V \times V \rightarrow R$ such that:

1. For all $(u, v) \in E$ (capacity constraint)

$$f(u, v) \leq c(u, v)$$

2. For all $u, v \in V$, (symmetry)

$$f(u, v) = -f(v, u).$$

3. For all $u \in V - \{s, t\}$, (flow conservation)

$$\sum_{v \in V} f(u, v) = 0.$$

The **value** of the flow is

$$|f| = \sum_{v \in V} f(s, v) = \sum_{v \in V} f(v, t).$$

MAX FLOW: Given a flow network G with source s and sink t , find the **maximum** flow.

Can be solved in polynomial time. (Hint: Use an algorithm for REACHABILITY !)

Reducing Bipartite Matching To Max Flow

A graph $G = (V, E)$ is **bi-partite** if the set of vertices V can be partitioned to two sets A and B , ($A \cup B = V$ and $A \cap B = \emptyset$), such that every edge in E has one adjacent vertex in A and one in B .

Given a bi-partite graph $G = (A, B, E)$ we can use a maximum flow algorithm to find maximum matching in G .

Define a directed graph $G' = (V' E')$:

- $V' = A \cup B \cup \{s, t\}$
- Connect s to all vertices in A . Direct all edges in E from A to B . Connect all vertices in B to t .
- All edges have capacity 1.