Decision Problems

Instance: an input, an output, and a size. (You know this already.)
Problem: a set of instances.
Decision Problem: Problem where each instance’s output is “T” or “F”.

“Given graph G and nodes x and y, what is the shortest path from x to y?" is not a decision problem.

“Given G, x, y, and k, is there a path from x to y of length ≤ k?“

Usually, given an algorithm for a decision problem, one can use it to solve the associated optimization problem.
- E.g., use binary search: “Is there a path of length ≤ 50?“, “≤ 25?“, “≤ 37?“, ...
Reductions

Given two decision problems A and B, we say that a function f from A to B is a reduction of A to B if, for every instance x in A, x is true if and only if f(x) is true.

We write $A \leq B$, and say A is reducible to B.

More about reductions

Example:
- Reducing a maximum matching problem on bipartite graph to a max flow problem.

Except for 2 trivial problems, every computable function is reducible to every other computable function.
- Let $b_T$ be a true instance of B & $b_F$ a false instance.
- Given instance $x$ of problem A, f could figure out the answer, then return $b_T$ if $x$ is true, $b_F$ otherwise.

Mathematicians use reductions to study uncomputable functions (like the halting problem).
**Polynomial Time Reducibility**

In computer science, we limit how much work a function $f$ can do.

- Typically, $f$ must be a polynomial-time algorithm.
- We write $A \leq_p B$, and say “$A$ is polynomial-time reducible to $B$” or “$A$ is no harder than $B$”.

If $A \leq_p B$, can we conclude $B \leq_p A$?

- Isn’t $f^{-1}$ a reduction of $B$ to $A$?
  
  (Give two reasons this doesn’t work)

**IMPORTANT!** To show $A \leq_p B$, we must show:

- For every instance $x$ of $A$, how to construct an instance $f(x)$ of $B$ relatively quickly (i.e., in polynomial time), and
- If $f(x)$ is true, then $x$ is true, and
- If $f(x)$ is false, then $x$ is false.

---

**Typical Garey&Johnson Style Entry**

Actually from [www.csc.liv.ac.uk/~ped/teachadmin/COMP202/annotated_np.html](http://www.csc.liv.ac.uk/~ped/teachadmin/COMP202/annotated_np.html)

Name: 3-Dimensional Matching (3DM) [SP1] 3

Input:

- 3 disjoint sets $X$, $Y$, and $Z$ each comprising exactly $n$ elements;
- A set $M$ of $m$ triples $\{(x_i, y_i, z_i) : 1 \leq i \leq m\}$ such that $x_i$ is in $X$, $y_i$ is in $Y$, and $z_i$ is in $Z$, i.e., $M$ is a subset of $X \times Y \times Z$.

Question: Does $M$ contain a matching?

- i.e. is there a subset $Q$ of $M$ such that $|Q|=n$ and for all distinct pairs of triples $(u,v,w)$ and $(x,y,z)$ in $Q$, $u \neq x$, $v \neq y$ and $w \neq z$.

Comments: The variant 2-dimensional matching in which 2 disjoint sets $X$ and $Y$ form the basis of a set of pairs, can be solved by a number of fast methods.
Some NP-complete problems
every literate computer scientist should know ...

Traveling Salesman
Subset Sum
Hamiltonian Cycle
K-Clique
3-Colorability
Satisfiability
3-Sat
...

3-SAT \leq_P 3-Colorable

Given 3-CNF formula \((x_1 \lor \overline{x}_3 \lor x_4) \land (x_2 \lor x_3 \lor \overline{x}_5) \land \ldots\)

Note that one of \(x_1\) and \(\overline{x}_1\) will be \(T\),
the other will be colored \(F\).
3-SAT \leq_p 3-Colorable

Given 3-CNF formula \((x_1 \lor \overline{x}_3 \lor x_4) \land (x_2 \lor x_3 \lor \overline{x}_5) \land \ldots\)

- If \(x_1\) is \(F\), this node must be \(U\).
- If \(x_1\) is \(T\), this can be \(F\).

This node is forced to be \(F\) unless \(x_1\) is \(T\) or \(x_3\) is \(F\). (Etc.)