P, NP Problems

– What is an efficient solution/algorithm?
– How to show that a problem does/does not have any efficient algorithm?
– How to show problems are as hard as another problem?
– NP-problems, NP-complete problems

Travelling Salesman Problem

– A weighted graph G (representing cities and distances between them)
– To find a tour (i.e., circuit passing through all the vertices exactly once) with minimal weight
– "Hard" to get the optimal answer

– Figuring out which problems are hard is important, so that

- We can try near best solutions rather than optimal
- Stop trying to find a "fast" algorithm
- Some algorithms, such as for cryptography, rely on some problems being hard

How do we show that some problems are hard?
– Prove lower bounds
(for example as in sorting using comparisons)
not always easy, specially for problems such as travelling salesman problem.
– Prove that it is at least as hard as some other problems, for which we do not know a fast algorithm
– Some of these are imporant problems, ...
– NP-complete problems

Efficient:
Polynomial time complexity.
– That is, for some polynomial p the time needed to solve a problem of size n is $p(n)$.

Polynomials: are n^3, $n^5 + 6n^2$, $n \log n$ etc
Exponential: 2^n, $2^{(n^2)}$, $2^{\sqrt{n}}$ etc.
Superpolynomials: $n^{\log n}$.
– Note that $2^{\log n}$ is polynomial!
– However, $2^{((\log n)^2)}$ is superpolynomial (as it is $n^{\log n}$)

Polynomials are slower growing than the exponentials.
— Though n^{100000} may not be so slow, ...

– Polynomials are closed under addition, multiplication and composition
(i.e., if $f(n)$ and $g(n)$ are polynomials, then so are $f(n) + g(n)$, $f(n) * g(n)$ and $f(g(n))$).
– If we can convert "fast" a problem A into another problem B, and there is a "fast" algorithm for B, then it reasonable to say that A can be solved fast.

Decision Problems and Function Problems

Decision Problem: Yes/No answer (which can also be taken as 1/0 answer).
– Is p a prime number?
– Is T a spanning tree of G?

Function Problem: Problems with general output.
– Size of the smallest tour.
– Size of the longest path.

Note that some function problems have a corresponding decision problem:
– Is the shortest tour of size smaller than 100?
– Is the shortest path of size smaller than k?
– Is the shortest path of size exactly k?
– Does the shortest path pass through vertex v?

In some cases, answers to decision problem can be converted to answers to function problem,

(for example by checking whether the:
1st vertex in the (lexicographically) shortest path is v_1?
1st vertex in the (lexicographically) shortest path is v_2? $\ldots$
2nd vertex in the (lexicographically) shortest path is v_1?
2nd vertex in the (lexicographically) shortest path is v_2? $\ldots$

– where if there are several shortest paths, then we take the one which is lexicographically first among these)

– Function problem/optimization problem is at least as hard as the decision problem.
– We will be concentrating on decision problems only.

P is the set of decision problems which can be solved in polynomial time in the length of the input.

Note: We are using length of the input. Number of bits needed to write the input.
So for example, if you use one positive integer x as input, then length of the input is $\lceil \log_2(x+1) \rceil$ (where if $x = 0$, then length of input is 1)
We use $|x|$ to denote the length of x.

If you have several inputs $x_1, x_2, \ldots, x_n$, then length of the input is sum of lengths of all the inputs.

– most of the problems we have seen so far are in P.

– EXP is the set of all decision problems which can be solved in exponential time in the length of the input (that is in time bounded by $2^{p(n)}$ for some polynomial p).
– Clearly, $P \subseteq EXP$.
– There are problems which are not in EXP. In fact there are "arbitrarily" hard problems, even problems which cannot be solved by a computer.

NP (Informally):
Set of problems for which 'proofs for answers' (certificates) can be verified in polynomial time.

NP: Formally, L is in NP if there exist an algorithm A (which runs in time polynomial in the length of the input) such that
– If $x \in L$, then for some y (called a certificate/proof) $A(x, y)$ accepts.
– If $x \notin L$, then for all y, $A(x, y)$ rejects.
– The length of certificate y in above cases are bounded by polynomial in the length of x (that is $|y| \leq p(|x|)$, for some fixed polynomial p).

SAT:

Input some variables $(x_1, x_2, \ldots, x_n)$ and a boolean formula over the variables such as:

$(x_1 \vee x_2 \vee \overline{x_3}) \wedge (x_4 \vee x_3) \wedge (x_1 \vee \overline{x_4})$

Question: Is there a truth assignment to the variables such that the formula evaluates to true?

– We can solve in exponential time by considering all possible truth assignments (2^n possibilities for the truth assignment to the variables).
– Certificates: "truth assignment" which makes the formula true.
— Then we can verify if the certificates are correct, by checking if the assignment indeed evaluates to true.
– size of the certificate is polynomial (actually, $O(n)$)
– The verification process takes time $O(m)$, where m is the length of the formula.
Thus, SAT is in NP.

Note: If the certificate is not valid (wrong), then the verifier must reject.

If $x \in L$: Then there is a valid/correct certificate
If $x \notin L$: Then there is no valid/correct certificate.

Certificates are of length polynomial in the size of input x.
There may be several valid certificates in case $x \in L$.

$P \subseteq NP$

– If $L \in P$, then there is a polynomial time algorithm A that can decide L.

– To show that L is in NP, the potential certificates we use is one bit.

– Verifier ignores the value of the certificate and just runs A on input x. If A accepts, then verifier also accepts. Otherwise, verifier rejects.

Note that in the proof above, if $x \in L$, then all certificates are correct. If $x \notin L$, then all certificates are wrong.

We can also show that

$NP \subseteq EXP$

(sketch: On a input, try all possible certificates. If one of them can be verified as correct, then accept the input. Otherwise reject the input

Time taken is exponential, as the number of possible certificates is bounded by $2^{p(n)}$, and each certificate can be checked in polynomial time).

Thus, $P \subseteq NP \subseteq EXP$.
We know that $P \neq EXP$.
We do not know whether $P = NP$ (most people think no)
We do not know whether $NP = EXP$ (most people think no)
If you show that $P = NP$ or $P \neq NP$, then you will become instantly famous, get Turing Award, ...

Reductions

To show that a problem B is at least as hard as A.

We say that $A \leq_m^p B$:
if there is a function f which can be computed in polynomial time such that
$x \in A$ iff $f(x) \in B$.

Note: Hardness is modulo polynomial time. That is, if B can be solved in time T(length of input), then A can be solved in time p(length of input) $+ T(p($ length of input $))$, where p is some fixed polynomial.

Theorem: If $A \leq_m^p B$ (as witnessed by f), and if B can be solved in polynomial time, then A can be solved in polynomial time.
This can be done as follows.

```
$A(x)$
Compute $f(x)$
Use algorithm for $B$ to find $B(f(x))$.
Output the answer as computed by the algorithm above.
```

Time taken by the above algorithm is:
$q(p(|x|)) + p(|x|)$, where $p(\cdot)$ is the polynomial time complexity of f, and q is the polynomial time complexity of solving B.

Some properties:

Transitive: If $A \leq_m^p B$ and $B \leq_m^p C$, then $A \leq_m^p C$.
(If f witnesses $A \leq_m^p B$ and g witnesses $B \leq_m^p C$. Then $g(f(\cdot))$ witnesses $A \leq_m^p C$).

Reflexive: $A \leq_m^p A$.

If $A \leq_m^p B$, and $B \in P$, then $A \in P$.

NP Complete Problems

L is said to be NP complete if
1. $L \in NP$
2. $L' \leq_m^p L$, for all $L' \in NP$.

If condition 2 is met, then we say that L is NP-hard.
Intuitively, NP-complete problems are the “hardest” problems in NP.

Theorem: If L is NP-complete and L can be solved in polynomial time, then all problems in NP can be solved in polynomial time (that is $P = NP$).

Theorem: If L_1 and L_2 are NP-complete, then both $L_1 \leq_m^p L_2$ and $L_2 \leq_m^p L_1$.
All NP-complete problems are equally hard (easy) since each of them can be reduced to each other.

Theorem: If B is NP-complete, $B \leq_m^p A$ and $A \in NP$, then A is NP-complete.
Proof: Suppose $L' \in NP$.
Since B is NP-complete, $L' \leq_m^p B$.
Along with $B \leq_m^p A$, and transitivity of reductions we get $L' \leq_m^p A$.
Thus, A is NP-hard.
As $A \in NP$, we get that A is NP-complete.

To show that a problem A is NP-hard we can:

1. Directly show that it is NP-hard, by showing
$L' \leq_m^p A$, for every $L' \in NP$.

2. Show that $B \leq_m^p A$, for some NP-complete problem B.

(1) is usually hard to show.
(2) is easier.
But we need one NP-complete problem to start with ....

CNF formulas:
$(x_1 \vee x_2 \vee \overline{x_5}) \wedge (x_7 \vee x_3) \wedge (x_1 \vee \overline{x_4} \vee x_7 \vee x_9)$

Variables: $x_1, x_2, \ldots$
Literals: $x_1, x_2, \ldots, \overline{x_1}, \overline{x_2}, \ldots$

clause: disjunction over literals.
$(x_1 \vee \overline{x_4} \vee x_7 \vee x_9)$

CNF: conjunction over clauses (sometimes we also say “set of clauses”)

3-CNF formulas: each clause has exactly three literals.

SAT (satisfiability) problem:
Input: A set of variables, and a CNF over the variables.
Question: Is there a truth assignment to the variables which makes all the clauses true?
If yes, the CNF formula is said to be satisfiable. Otherwise it is said to be unsatisfiable.

Example: $(x_1 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_3}) \wedge (\overline{x_2} \vee x_3)$

is satisfiable (by taking $x_1 = true$ and $x_2 = x_3 = false$)

Example: $(x_1 \vee x_2) \wedge (\overline{x_1} \vee x_2) \wedge (x_1 \vee \overline{x_2}) \wedge (\overline{x_1} \vee \overline{x_2})$ is not satisfiable.

SAT problem is NP-complete.
In NP:
1. Certificates are the truth assignments to the variables which make the formula true. Note that the length of the certificate is proportional to the number of variables, and thus linear in the length of the input.
2. To check a certificate: just evaluate the formula on the truth assignment. It takes time proportional to the length of the formula, and thus linear in the length of the input.
NP-Hard:
– Proved by Cook
– Proof a bit complicated to do in this course.

Some famous NP complete problems.

1. Satisfiability:
INSTANCE/INPUT: A set U of variables and a collection C of clauses over U.
QUESTION: Is there a satisfying truth assignment for C?

2. 3-Dimensional Matching:
INSTANCE/INPUT: Three disjoint finite sets X, Y, Z, each of cardinality n, and a set $S \subseteq X \times Y \times Z$.
QUESTION: Does S contain a matching?
That is, is there a subset $S' \subseteq S$ such that $card(S') = n$ and no two elements of S' agree in any coordinate?

3. Vertex Cover:
INSTANCE/INPUT: A graph $G = (V, E)$ and a positive integer $K \leq card(V)$.
QUESTION: Is there a vertex cover of size K or less for G?
That is, is there a subset $V' \subseteq V$ such that, $card(V') \leq K$ and for each edge $(u, v) \in E$, at least one of u, v belongs to V'?

4. MAX-CUT:
INSTANCE/INPUT: An undirected graph $G = (V, E)$, and a positive integer $K \leq card(E)$.
QUESTION: Is there a cut of G with size $> K$?
Here (X, Y) is said to be a cut of G, if (X, Y) is a partition of V. That is, $X \cap Y = \emptyset$ and $X \cup Y = V$. Size of a cut (X, Y) of G, is $card(\{(v, w) \mid v \in X \text{ and } w \in Y \text{ and } (v, w) \in E\})$. That is, size of a cut (X, Y) is the number of edges in G which connect X and Y.

5. Clique:
INSTANCE/INPUT: A graph $G = (V, E)$ and a positive integer $K \leq card(V)$.
QUESTION: Does G contain a clique of size K or more?
That is, is there a subset $V' \subseteq V$, such that $card(V') \geq K$, and for all distinct $u, v \in V'$, $(u, v) \in E$?

6. Hamiltonian Circuit:
INSTANCE/INPUT: A graph $G = (V, E)$
QUESTION: Does G contain a Hamiltonian circuit?
That is, is there a simple circuit which goes through all the vertices of G?

7. Partition:
INSTANCE/INPUT: A finite set A and a size $s(a) > 0$, for each $a \in A$.
QUESTION: Is there a subset A' of A such that $\Sigma_{a \in A'} s(a) = \Sigma_{a \in A - A'} s(a)$?

8. Traveling Salesman Problem:
INSTANCE/INPUT: A complete weighted graph $G = (V, E)$, and a bound B.
QUESTION: Is there a Hamiltonian circuit of weight $\leq B$?