CS350 ABET Objective 7

Describe the notions of P, NP, NPC, and NP-hard.

Textbook Section 11.3, ~5%.

Some problems can be solved by relatively efficiently, e.g., sorting a sequence of 1,000,000 numbers. Some problems cannot be solved in practice, e.g., printing all the permutations of a sequence of 1,000,000 numbers. Some problems cannot be solved in theory, e.g., telling whether a program terminates for some input.

There are also gray areas. We do not know how to solve some problem efficiently, and we do not know whether solving them efficiently is impossible or we have not yet found an efficient algorithm.

This objective introduces some concepts and examples to help understanding and reasoning about these issues.

© 2012, S. Antoy

P Problems

Problems that can't be solved in polynomial time are called *intractable*.

Can solve these problems only for small inputs.

E.g.: Minimum number of colors necessary to color a map.

Decision problem, answer is yes/no.

Are m colors sufficient to color a map, for m=1,2,...

Class P: decision problems solvable in polynomial time (with deterministic algorithm).

Field called *computational complexity.*

© 2012, S. Antoy

Undecidable Problems

Some problems have no algorithm to solve them. E.g.: Halting Problem.

A algorithm
P program
I input

$$A(P,I)=\begin{cases}1 & if\ \ P\ \ halts\ \ on\ \ I \\ 0 & otherwise\end{cases}$$

$$Q(P)=\begin{cases}halts & if\ A(P,P)=0 \quad (P\ loops\ on\ P) \\ loops & if\ A(P,P)=1 \quad (P\ halts\ on\ P)\end{cases}$$

Next is a contradiction.

$$Q(Q)=\begin{cases}halts & if\ A(Q,Q)=0 \quad (Q\ loops\ on\ Q) \\ loops & if\ A(Q,Q)=1 \quad (Q\ halts\ on\ Q)\end{cases}$$

Therefore, A does not exist.

© 2012, S. Antoy

Difficult Problems

No polynomial time algorithm is known (but might exist) for the following:

1. Hamiltonian circuit: a closed path through each node of a graph once
2. Traveling salesman: shortest tour through n cities with positive integers distances (shortest Hamiltonian circuit)
3. Knapsack: most valuable subset of n items that fit into knapsack
4. Partition: partition n positive integers into subsets with same sum
5. Bin-packing: Put n values in (0..1] into fewest bins of size 1
6. Graph-coloring: chromatic number of a map/graph (adjacent node have different colors)
7. Integer linear programming: min or max of linear function of several integer variables with finite set of eq and neq constraints

They all have exponential number of choices

Eulerian circuit: graph traverse all edges exactly once (exponential number of choices, but O(n^2) algorithm).

© 2012, S. Antoy

NP problems

2-step non-deterministic algorithm: (1) guess solution, (2) verify solution.

Non-det polynomial if (2) is polynomial time.

Class NP is the set of all problems solved by non-deterministic polynomial time algorithm.

P is contained in NP.

NP contains Hamiltonian circuit, traveling saleman, etc.

Halting problem is not in NP

Question: is P = NP ? Probably not.

NP-Complete problems

Problem D1 is polynomially reducible to D2 if exists function t that maps instances of D1 to instances of D2 with:

1. t maps a yes/no instance to same answer instance;
2. t is computable in polynomial time.

Problem D is *NP-complete* if it is in NP and every problem in NP is polynomially reducible to D.

E.g., map Hamiltonian circuit to Traveling Salesman.

1. if (n1,n2) is an edge in HC, then weight(n1,n2)=1 in TS.
2. Connect all pairs of other node in TS with weight=2.

© 2012, S. Antoy

CNF-SAT

First NP-complete problem, e.g.:

$$(x \vee \neg y) \wedge (\neg x \vee y \vee z) \wedge (x \vee z)$$

Many other problem known to be NP-complete: HC, TS, etc., see above.

Primality was thought be NP-complete, but it is not [2002].

Not yet efficient algorithm for factoring integers.

How to prove that some X is NP-complete:

1. X is in NP;
2. every (one is enough) NP-complete problem Y maps to X in poly time.

NP-hard

A problem P is NP-hard if there is an NP-complete problem Q poly time reducible to P. P may or may not be in NP.

P can be used to solve Q.

P is at least as difficult as any NP-problem.

If there is a polynomial algorithm for any NP-hard problem, then there are polynomial algorithms for all problems in NP, and consequently P = NP.

If P ≠ NP, then NP-hard problems have no solutions in polynomial time, while P = NP does not resolve whether the NP-hard problems can be solved in polynomial time.

References

Textbook Section 11.3

Web:

http://mathworld.wolfram.com/ComplexityTheory.html

http://en.wikipedia.org/wiki/Computational_complexity_theory