COM2031 Advanced Algorithms, Autumn Semester 2019

Lab 11: Complexity Classes Visualizations

Purpose of the lab

This lab is a tutorial to empower your understanding of Complexity Classes using interactive visualisation to study for the final exam as well (Maji, 2015). It is followed by exercises to check your understanding.

Create a teacher account in the following website:
https://canvas.instructure.com/

and enrol in the following course: Senior Algorithms Generic Course
Go to Chapter 8: Limits to computing. You can empower your Computing Limits and reductions and NP-Complete definition and proof understanding by reading the first four sections.

The first NP-Complete problem is circuit satisfiability. Other problems were discussed in the lecture including: Formula Satisfiability, 3-CNF satisfiability, Hamiltonian cycle, Independent Set, Vertex Cover, and Travelling salesman. Go through the slides of the interactive visualisations of these problems to study their dynamics, and do 4 exercises at least from the following sections:

08.05 The Circuit Satisfiability problem
08.06 The Formula Satisfiability problem
08.07 The 3-CNF Satisfiability problem
08.09 The Independent Set problem
08.10 The Vertex Cover Problem
08.11 The Hamiltonian Cycle problem
08.12 The Traveling Salesman problem

Then study how these problems are reduced to each other by going through the slides of the interactive visualisations in the following sections:

08.13 Reduction of circuit SAT to SAT.
08.14 Reduction of SAT to 3-SAT.
08.16 Reduction of Independent Set to Vertex Cover
08.17 Reduction of 3-SAT to Hamiltonian Cycle Problem.

Exercises:

Now test your understanding by attempting a solution in the following questions and discuss on SurreyLearn discussion Forum. A sample solution will be posted next week.

1) For each of the two questions below, decide whether the answer is (i) "Yes," (ii) "No," or (iii) "Unknown, because it would resolve the question of whether P = NP." Give a brief explanation of your answer.

(a) Let's define the decision version of the Interval Scheduling Prob- lem from Chapter 4 as follows: Given a collection of intervals on a time-line, and a bound *k*, does the collection contain a subset of nonoverlapping intervals of size at least *k*?

Question: Is it the case that Interval Scheduling $\leq_P$ Vertex Cover?

(b) Question: Is it the case that Independent Set $\leq_P$ Interval Scheduling?

2) A store trying to analyse the behaviour of its customers will often maintain a two-dimensional array *A*, where the rows correspond to its customers and the columns correspond to the products it sells. The entry *A*[*i*, *j*] specifies the quantity of product *j* that has been purchased by customer *i*.

Here's a tiny example of such an array *A*.

	liquid detergent	beer	diapers	cat litter
Raj	0	6	0	3
Alanis	2	3	0	0
Chelsea	0	0	0	7

One thing that a store might want to do with this data is the following. Let us say that a subset *S* of the customers is *diverse* if no two of the of the customers in *S* have ever bought the same product (i.e., for each product, at most one of the customers in *S* has ever bought it). A diverse set of customers can be useful, for example, as a target pool for market research.

We can now define the Diverse Subset Problem as follows: Given an $m \times n$ array *A* as defined above, and a number $k \leq m$, is there a subset of at least *k* of customers that is *diverse*?

a) Show that Diverse Subset is NP-complete.

For more theoretical discussions, go through the following sections:
08.18 Coping with NP-complete problems
08.19 Unsolveable Problems
08.20 Turing Machines

Key Concepts:

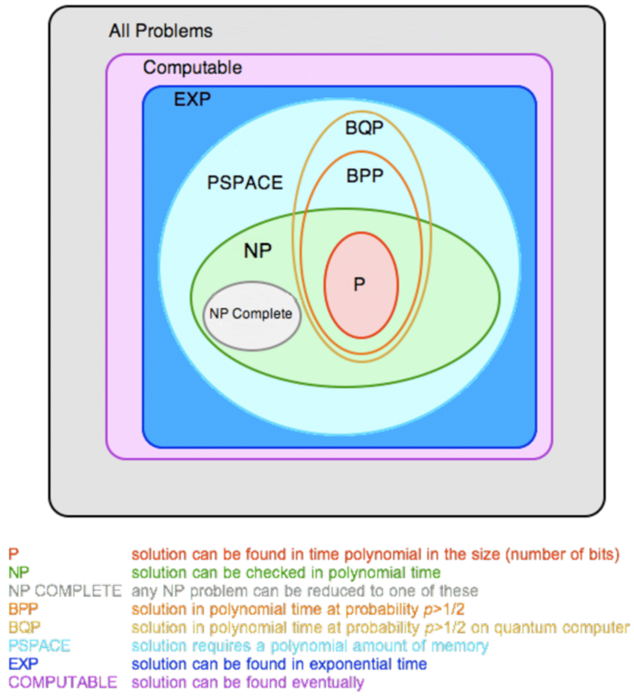

Figure 1 From published paper: Adiabatic Quantum Computing (Pinski, 2011)

P (Polynomial) Class Problems: Problems that have polynomial time algorithm in n: the size of the input.

NP (Non-deterministic Polynomial) Class Problems: Problems that we do not have a polynomial time algorithm to produce the solution. However, if the solution is produced to us using a non-deterministic parallel machine that can produce all permutations of the possible solutions for us, we can use a polynomial certifier to verify that this solution is a valid one. We do not have this huge machine yet for large n.

EXP Class Problems: Problems in which a solution can be produced in exponential time $O(c^n)$ in which the constant $c > 1$ and n is the data size.

NP-Hard Class Problems: problems that are NP, we can verify a solution in polynomial time, but cannot produce a solution in polynomial time. "Hard" means that the best-known algorithm for the problem is expensive in its running time. One example of a hard problem is Towers of Hanoi. It is easy to understand this problem and its solution. It is also easy to write a program to solve this problem. But it takes an extremely long time to run for any "reasonably" large value of n. Try running a program to solve Towers of Hanoi for only 30 disks! NP-Hard are EXP Class of problems mainly.

NP-Complete Problems: Since this class of problems is defined such that any NP problem can reduce to one of these, then if a polynomial time algorithm is found to an NP-Complete problem, then P = NP, and you win a million-dollar prize ☺, and solve all problems that can be reduced to them.

Bibliography

Maji, N., 2015. An Interactive Tutorial for NP-Completeness (Thesis submitted in partial fulfilment of the requirements for the degree of Master of Science in Computer Science and Applications). Faculty of the Virginia Polytechnic Institute and State University, Blacksburg, Virginia.

Pinski, S.D., 2011. Adiabatic Quantum Computing. arXiv:1108.0560 [physics, physics:quant-ph].