Mathematical Insights in Computing, Day 15

P versus NP, and NP-completeness

Problem 1. Knights and Knapsacks

You are playing Your Favorite Video Game™. You're saving up to buy a *magic knapsack of holding*, which costs $***C*** and can store an *infinite* number of items, but for now you just have your *ordinary* knapsack which isn't magical at all. It only stores a finite number of items. To be precise, it can store up to ***K*** pounds of stuff.

Fortunately, you have recently defeated the dragon which was guarding a giant stash of treasure – look at all of the valuable, expensive items laying on the ground! Gold cups, rubies, marble statues! That magic knapsack of holding will be yours for sure!

Unfortunately, you have recently defeated the dragon which was guarding a giant stash of treasure – and you only brought your *ordinary* knapsack with you! It is a long, long trek back to civilization, so you can only bring home with you *some* of the fabulous riches that gleam in front of you. They're not all going to fit.

At least you brought your computer along with you to help.

Consider the following problem, which we'll call the Knapsack Problem:

Knapsack*(items, K, C):*

> *Given a list of items, each of which has a weight* ***w*** *pounds and a value $****v****, is it possible* ***(yes or no)*** *to fill up a knapsack with maximum-capacity* ***K*** *pounds in such a way that the knapsack contains more than $****C*** *worth of stuff?*

a. You remember from your Mathematical Insights in Computing class that the Knapsack Problem is **NP-complete**. This means two things: that it is **NP-hard** and that it is **in NP**.

   Show that the Knapsack Problem is in NP:
   What **witness** could you give in order to prove to someone that the answer is yes?

(We won't show that the Knapsack Problem is NP-hard, because it's rather tricky.)

b. Even if the problem is NP-complete, you can still give your best shot at making an algorithm for it – it'll just be **slow**. Describe an algorithm you could use to find some items to put in your knapsack so that you can afford the magic knapsack once you reach civilization. *It doesn't have to be fast, it just has to work.*

c. If there are *n* items that you can choose from, how long does your algorithm take?

d. Is your algorithm: (circle one)

exponential or polynomial

Problem 2. Instances and Witnesses

The following problems are in NP (though, note: they might not be NP-complete!) As we go along, fill in the blanks.

Problem & Description	What is an example of an "instance" of the problem?	What is an example of a "witness" for that instance?
SUDOKU(X): Is the Sudoku problem X solvable?		
HamiltonPath(G): If G is a graph of cities and roads, is there a path that hits each city once and only once?		
HasFactor(N,m): If N is a number and $0 < m < N$, does N have a prime factor d with $0 < d < m$?		
IsProduct(a,b,c): Is $c = a \times b$?		
3COLOR(G): If G is a graph of vertices connected by edges, can the vertices be *colored* red, yellow, or blue so that no two same colors touch?		

Problem 3. Will We Ever Know?

The following are a series of *True, False,* or *Open* questions. They're like True or False questions... except sometimes modern mathematics isn't sure of the answer.

Answer, and give a one-sentence explanation for each. Careful: some are tricky!

a. If P ≠ NP, then there is no polynomial-time solution to the Traveling Salesman Problem. T F O

b. If P ≠ NP, then there is no polynomial-time factoring algorithm. T F O

c. Either P ≠ NP, or NP ≠ EXP, or both. T F O

d. All of the problems in NP are NP-complete. T F O

e. All of the NP-hard problems are NP-complete. T F O

f. All NP-complete problems are also NP-hard. T F O

g. There is an algorithm for the Traveling Salesman Problem. T F O

And some review problems...

h. Every sound mathematical theory is consistent. T F O

i. Every consistent mathematical theory is sound. T F O

j. Every decidable language is context-free. T F O

k. Every context-free language is regular. T F O

l. Every regular language is decidable. T F O

m. If we had an oracle for the Halting Problem, everything would be solvable. T F O