Algorithms – CMSC-27200

http://alg15.cs.uchicago.edu Homework set #9.

Due Wednesday, March 11, 2015

Posted 3-5. Updated 3-6 12:15am. Problem 9.6 added 3-7 1:15pm.

Read the homework instructions on the website. The instructions that follow here are only an incomplete summary.

Hand in your solutions to problems marked "HW." Do not hand in problems marked "DO." If you hand in homework marked **"XC" (extra credit)**, do so on **separate and separately stapled sheets, please**. PRINT YOUR NAME and SECTION NUMBER ON EVERY SHEET you submit. **Use LaTeX to typeset your solutions**. Hand in your solutions on paper, do not email.

When writing pseudocode, **explain the meaning of your variables.** Use comments to explain what is happening in each line. Also, give a short explanation of the idea behind the algorithm. **Unless otherwise stated, describe your algorithms in pseudocode. Elegance of your code matters.**

Carefully study the policy (stated on the website) on collaboration, internet use, and academic integrity. **State collaborations and sources both in your paper and in email to the instructors.**

9.1 **HW (3+3+8 points) (Knapsack decision problem NP-complete)**

(a) State the decision version of the Integer Knapsack problem. You need to clearly state the input and state the yes/no question the problem asks.

(b) Let KNAP denote the laguage corresponding to this decision problem. Prove: KNAP $\in \mathsf{NP}$. Your answer should be half a line: clearly state the witness of membership. In another copuple of lines, state the relevant facts about this witness. (No need to prove these facts, they should be obvious.)

(c) Recall from class the Subset-Sum problem:
INPUT: positive integers $a_1, \dots, a_n, b$.
QUESTION: $(\exists I \subseteq \{1, \dots, n\}) \left(\sum_{i \in I} a_i = b \right)$?
Let SUBS denote the corresponding language. Recall from class that SUBS is NP-complete. Use this to prove that KNAP is NP-complete. – Instructions: Given (b), what you need to do is

construct a Karp reduction. Clearly state which language you are reducing to which language. (Consult problem 8.2 for the definition of Karp-reduction and the LaTeX code for $\preccurlyeq$.)

9.2 **HW (2+2+5 points) (NP-hard problems)** We say that a computational task f is NP**-hard** if every problem in NP is Cook-reducible to f. (Cook reductions were covered in the 2/27 class.)

(a) Let L_1 and L_2 be languages. Consider the following two statements:
(A) $L_1 \preccurlyeq_{\text{Cook}} L_2$ (B) $L_1 \preccurlyeq_{\text{Karp}} L_2$.
Which of the following two statements is evident from the definitions: (a1) (A) $\Longrightarrow$ (B) (a2) (B) $\Longrightarrow$ (A). Briefly reason your answers. (As always, we view languages as decision problems and vice versa.)

(b) State the Integer Knapsack problem (i. e., the Knapsack Problem with integer weights and values). (Accurately state the input and the output. Note that this is not a decision problem but an optimization problem.)

(c) Prove that the Integer Knapsack problem is NP-hard by Cook-reduction from KNAP. Describe your algorithm in English, no pseudocode needed. Specifically state the number of calls your algorithm makes to the Integer Knapsack[1] oracle. (Such an oracle tells the solution to any instance of the Integer Knapsack problem it is fed.)

9.3 **(Hamilton path vs. Hamilton cycle)** The input is an undirected graph. Recall that a **Hamilton path** is a path that passes through every vertex. (So its length is $|V|-1$.) A **Hamilton cycle** is a cycle that passes through every vertex. (So its length is $|V|$.) A graph is **Hamiltonian** if it has a Hamilton cycle. Let HAMILTONIAN denote the class of Hamiltonian graphs. Let HAMILTON-PATH denote the class of graphs with a Hamilton path. Assuming the theorem that HAMILTONIAN is NP-complete, prove:

(a) **HW (8 points)** HAMILTON-PATH is NP-hard.

(b) **XC (6 points)** HAMILTON-PATH is NP-complete.

[1] A previous version of this problem erroneously referred to a "KNAP oracle." Corrected 03-10 4am.

In each case, solve the problem by reduction from HAMILTONIAN. State what type of reduction you are using. The solution to part (a) should be a very simple algorithm. Write it in pseudocode. Correctness should be obvious. – For part (b), give an accurate mathematical description of the reduction and prove that it is correct. – If you solve part (b) only, you will get half credit for part (a) as well. To get full credit for part (a), you need to give a separate "easy" solution to part (a).

9.4 **(Metric Traveling Salesman)** The input to the Metric Traveling Salesman Problem (MTSP) is an edge-weighted complete graph with positive edge-weights that satisfy the triangle inequality: for any three distinct vertices x, y, z we have $w(x, y) + w(y, z) \geq w(x, z)$. The task is to find a min-cost Hamilton cycle. In the Integer MTSP (IMTSP) we assume that all weights are integers.

(a) **HW (2 points)** State the decision version of IMTSP. Call the corresponding language[2] L_{IMTSP}. Prove that it belongs to NP. (Name the witness.)

(b) **HW (6 points)** Prove that L_{IMTSP} is NP-complete by Karp-reduction from HAMILTONIAN.

(c) **XC (6 points)** Find a polynomial-time algorithm that approximates IMTSP within a factor of 2, i.e., the output must be a Hamilton cycle that is at most twice as expensive as an optimal Hamilton cycle. Prove this approximation ratio. Describe your algorithm in clear English, no pseudocode required. (Hint: Use min-cost spanning tree.) — Comment. One can actually reach an approximation ratio of 3/2 using the fact that min-cost perfect matching can be computed in polynomial time. Do not use this result.

9.5 **HW (3+6 points) (Amortized analysis: Queue via stacks)** Study the "Amortized analysis" handout. Solve the "Queue via stacks" problem stated there.

9.6 **(MAX-3SAT)** Recall that a *Boolean variable* takes values 0 or 1 and a *literal* is a Boolean variable or its negation. A *disjunctive clause* is an OR of literals corresponding to distinct[3] variables, e. g., $x_1 \vee$

[2]This notation was added to parts (a) and (b) 03-10 4:45am.
[3]The distinctness requirement was added 3-8 8:50pm.

$\overline{x_4} \vee x_7$ (but not $x_4 \vee \overline{x_4} \vee x_7$ nor $x_1 \vee \overline{x_4} \vee x_1$). A *disjunctive k-clause* is a disjunction of k literals corresponding to k distinct variables (so the example above is a disjunctive 3-clause). The MAX $-$ 3SAT problem is an optimization problem defined as follows: INPUT: a list of disjunctive 3-clauses, $C_1, \dots, C_s$. Question: Find the maximum number m such that m of the clauses can be simultaneously satisfied, i. e., there exists an assignment of $(0, 1)$-values to the variables that satisfies at least m of the clauses. Let the variables be $x_1, \dots, x_n$.

(a) **HW (2+4 points)** Prove that $m \geq 7s/8$. Reproduce with full details the proof seen in class: flip a coin for each variable to decide its value. (a1) What is the size of the sample space for this experiment? (a2) Let X be the number of satisfied clauses (so X is a random variable). Prove: $E(X) = 7s/8$. Give a clear definition of the random variables you use.

(b) **HW (4 points)** Assume $8 \mid s$. Prove that the inequality $m \geq 7s/8$ is tight. (For every s that is divisible by 8 you need to construct a set of s disjunctive 3-clauses of which no more than $7s/8$ can be simultaneously satisfied.)

(c) **XC (5 points)** Let X be the random variable defined in part (a). Prove: $P(X \geq 7s/8) \geq 1/(s + 1)$. (Use Markov's inequality: If Y is a non-negative random variable and $a > 0$ then $P(Y \geq a) \leq E(Y)/a$.)

(d) **HW (3+2 points)** Part (a) asserts that there exists an assignment to the variables that satisfies at least $7s/8$ of the clauses. We want to find such an assignment as follows: we repeat the experiment described in (a) until we find an assignment that satisfies at least $7s/8$ clauses. (d1) Based on (c), what can we say about the expected number of trials? (d2) Does this randomized algorithm run in polynomial expected time?

(e) **XC (8 points)** Derandomize this algorithm. In other words, construct a polynomial-time deterministic algorithm that will always find a substitution that satisfies at least $7s/8$ of the clauses. Your algorithm should be based on the idea of (a). Prove that your algorithm is correct. (State an appropriate loop invariant.) Analyze the running time of your algorithm.

9.7 **(BONUS problem for your amusement, not required)** If you hand in a solution (make sure to put it on a separate sheet), you will

receive feedback, but no points.
A set S is *countable* if it has an injection into the set $\mathbb{N} = \{1, 2, 3, \dots\}$ (the positive integers). So every finite set is countable; and every subset of $\mathbb{N}$ is countable. S is *countably infinite* if it is countable and it is infinite. S is *uncountable* if S is not countable. Prove:

(a) If S is countably infinite then there is a bijection between S and the set of positive integers.

(b) The union of countably many countable sets is countable.

(c) If Σ is a finite alphabet then Σ^* is countable.

(d) If Σ is a countable alphabet then Σ^* is countable.

(e) The set of rational numbers is countable.

(f) The complexity class P is countable (i. e., there are countably many languages in P).

(g) The complexity class NP is countable (i. e., there are countably many languages in NP).

(h) The complexity class coNP is countable (i. e., there are countably many languages in coNP).

(i) The set of infinite $(0, 1)$-strings in uncountable.

(j) The set of real numbers is uncountable.

(k) If $\Sigma \neq \emptyset$ then the set of languages over Σ is uncountable.