CS 620 Fall 2010 at ISU, Exam 1 sample exam

Prepared by assistant professor Jeff Kinne on September 22, 2010. This sample exam hopefully gives you an idea of what to expect for the exam. You will have 50 minutes to take the exam. The exam itself and blank paper I provide are all you will have to use (no computer, textbook, notes, cellphone, calculator, etc.). This sample exam is slightly longer than the actual exam will be; while this exam has 9 problems, the actual exam will probably have 6 or 7 problems.

Decidable/Undecidable problems (25 Points total)

Problem 1 (10 Points) Is it true or false that all PSPACE problems are decidable? Explain.

Answer: True. PSPACE problems have algorithms that solve them and halt on all inputs (in at most exponential time as we have mentioned in class).

Problem 2 (15 Points) Recall that we know the halting problem is undecidable. The halting problem is defined such that $(M, x) \in S_{halt}$ iff $M(x)$ halts.

Prove that the following problem is undecidable by giving a reduction from the halting problem. The language "n^2 time machines" is defined so that $(M) \in S_{n^2 machines}$ iff M is a machine that halts in at most n^2 time, where n is the length of the input. *Given (M, x), create a machine M'_x such that $M(x)$ halts iff M'_x is a machine that runs in MORE THAN n^2 on some input.*

Answer: Note: there was a mistake in the sample exam I handed out, but I have corrected it above. We create M'_x so that it runs in n^2 iff $M(x)$ DOES NOT halt. This is fine and shows that the n^2 time machines problem is undecidable, but the hint was wrong and it is a bit trickier than what I thought originally.

Let (M, x) be an instance of the halting problem, so we want to know if $M(x)$ halts or not. We create a machine M'_x such that $M(x)$ halts iff M'_x is a machine that runs in more than n^2 time on some input. M'_x is defined so that $M'_x(y)$ simulates $M(x)$ while this simulation uses less than $|y|^2$ time. If the simulation halts then $M'_x(y)$ goes into an infinite loop. If the simulation does not halt in $|y|^2$ time, then $M'_x(y)$ halts. Notice that if $M(x)$ halts, then on a large enough input y, $M'_x(y)$ will discover this while using at most $|y|^2$ time, and will go into an infinite loop, meaning M'_x is not an n^2 time machine. If $M(x)$ does not halt, then M'_x is an n^2 time machine.

NP problems (30 Points total)

Problem 3 (10 Points) Let G be a graph and $k \geq 0$ an integer. Let Clique be the problem defined such that $(G, k) \in S_{clique}$ iff G is a graph that has a clique of size at least k. A clique of size k is a set of k vertices such that all pairs of vertices in the clique are connected in the graph. Show that Clique is contained in NP.

Answer: The witness for a positive instance is a subset of vertices of size k. The verifier simply checks that each of these vertices is connected to each other in the graph G, and accepts if so. This requires checking k^2 edges, and since $k \leq$ the input length, this takes polynomial time. Notice that there is a witness causing the verifier to accept iff there is a clique of size at least k.

Problem 4 (10 Points) Let $x_1, ..., x_k$ and t be numbers. Let Subset Sum be the problem defined such that $(\{x_1, x_2, ..., x_k\}, t) \in S_{subset-Sum}$ iff there is a subset $\{y_1, ..., y_\ell\} \subseteq \{x_1, x_2, ..., x_k\}$ such that $\sum_{j=1}^{\ell} y_j = t$. For example, $(\{2, 3, 4\}, 5)$ is in Subset-Sum because $2 + 3 = 5$, and $(\{2, 3, 4\}, 4)$ is not in Subset-Sum. Show that Subset-Sum is contained in NP.

Answer: The witness for a positive instance is a subset that sums up to t. The verifier simply checks that the sum of the given subset is equal to the target, and accepts if so. This requires adding up at most k numbers, each of which is fewer bits than the input length, which can be done in polynomial time. Notice that there is a witness causing the verifier to accept iff there is a subset adding up to t.

Problem 5 (10 Points) Show that if Π is an NP-complete problem and $\Pi \in$ P then P = NP.

Answer: By the definition of NP-complete, for every NP problem Π' there is a polynomial-time reduction from Π' to Π. Then we can solve Π' by first reducing to Π and then solving the Π problem in polynomial time (since we have assumed P=NP).

P problems (20 Points total)

Problem 6 (10 Points) Let $x_1, ..., x_k$ and t be numbers. Let Unordered-Search be the problem defined such that $(\{x_1, ..., x_k\}, t) \in S_{unordered-Search}$ iff t is in the list of numbers – there exists a $1 \leq i \leq k$ such that $t = x_i$. Give pseudocode for an algorithm that solves this problem.

Answer:

1. For $i = 1$ up to k do the following.
    a. If $x_i == t$, then return "true/accept".
2. If the for loop ends without returning true, then return "false/reject".

Problem 7 (10 Points) For the algorithm you gave in the last problem, give a big-O estimate of the running time.

Answer: Let n be the bit-length of the input. Step (a) of checking whether $x_i == t$ takes $O(n)$ time. We perform this step $O(n)$ times, for a total running time of $O(n^2)$.

PSPACE problems (25 Points total)

Problem 8 (12.5 Points) Explain why PSPACE is contained in EXP (that any problem solvable with a polynomial amount of memory space can be solved in time $2^{n^{O(1)}}$).

Answer: A PSPACE algorithm uses a polynomial amount of memory space, so its internal configuration (memory contents, etc.) can be described by a polynomial number of bits. Then the total number of possible internal configurations is exponential, meaning that if the algorithm halts on a given input, it must do it in exponential time (taking longer time would require going through a cycle in the configuration graph, which would mean an infinite loop). So a PSPACE machine that halts on all inputs can use at most exponential time. (And even a PSPACE algorithm that does not halt on all inputs can be decided in exponential time.)

Problem 9 (12.5 Points) Show that if Π is a PSPACE-complete problem and $\Pi \in$ P then P = NP.

Answer: NP is contained in PSPACE: given a poly-time verifier, the witness/proof length is polynomial, and doing a brute-force search over all possible witnesses/proofs takes only polynomial memory space. Since NP is contained in PSPACE, if PSPACE=P then also all NP problems have poly-time algorithms.