Exam 3 2018 Fall_sol

1) 20 pts
Mark the following statements as **TRUE** or **FALSE**. No need to provide any justifi-cation.

[TRUE/FALSE]
Let X be a decision problem. If we prove that X is in the class NP and give a poly-time reduc-tion from X to 3-SAT, we can conclude that X is NP-complete.

[TRUE/FALSE]
Let A be an algorithm that operates on a list of n objects, where n is a power of two. A spends $\Theta(n^2)$ time dividing its input list into two equal pieces and selecting one of the two pieces. It then calls itself recursively on that list of n/2 elements. Then A's running time on a list of n elements is O(n).

[TRUE/FALSE]
If there is a polynomial time algorithm to solve problem A then A is in NP.

[TRUE/FALSE]
A pseudo-polynomial time algorithm is always slower than a polynomial time algo-rithm.

[TRUE/FALSE]
In a dynamic programming formulation, the sub-problems must be non-overlapping.

[TRUE/FALSE]
A spanning tree of a given undirected, connected graph G = (V, E) can be found in O(E) time.

[TRUE/FALSE]
Ford-Fulkerson can return a zero maximum flow for flow networks with non-zero ca-pacities.

[TRUE/FALSE]
If $f(n) = \Theta(g(n))$ and $g(n) = \Theta(h(n))$, then $h(n) = \Theta(f(n))$

[TRUE/FALSE]
There is a polynomial-time solution for the 0/1 Knapsack problem if all items have the same weight but different values.

[TRUE/FALSE]
If there are negative cost edges in a graph but no negative cost cycles, Dijkstra's algo-rithm still runs correctly.

Exam 3 2018 Spring_sol

1) 20 pts
Mark the following statements as **TRUE** or **FALSE**. No need to provide any justification.

TRUE/FALSE]
To prove that a problem X is NP-hard, it is sufficient to prove that SAT is polynomial time reducible to X.

TRUE/FALSE]
If a problem Y is polynomial time reducible to X, then a problem X is polynomial time reducible to Y.

TRUE/FALSE]
Every problem in NP can be solved in polynomial time by a nondeterministic Turing machine.

TRUE/FALSE]
Suppose that a divide and conquer algorithm reduces an instance of size n into 4 instances of size $n/5$ and spends $\Theta(\underline{n})$ time in the conquer steps. The algorithm runs in $\Theta(n)$ time.

TRUE/FALSE]
A linear program with all integer coefficients and constants must have an integer optimum solution.

TRUE/FALSE]
Let M be a spanning tree of a weighted graph G=(V, E). The path in M between any two vertices must be a shortest path in G.

TRUE/FALSE]
A linear program can have an infinite number of optimal solutions.

TRUE/FALSE]
Suppose that a Las Vegas algorithm has expected running time $\Theta(n)$ on inputs of size n. Then there may still be an input on which it runs in time $\Omega(n^2)$.

TRUE/FALSE]
The total amortized cost of a sequence of n operations gives a lower bound on the total actual cost of the sequence.

TRUE/FALSE]
The maximum flow problem can be efficiently solved by dynamic programming.

Exam 3 2017 Fall_sol

1) 20 pts
Mark the following statements as **TRUE** or **FALSE**. No need to provide any justification.

[TRUE/FALSE]
To prove that a problem X is NP-complete, it is sufficient to prove that 3SAT is polynomial time reducible to X.

[TRUE/FALSE]
Finding the minimum element in a binary max heap of n elements takes O(log n) time

[TRUE/FALSE]
We are told that in the worst case, algorithm A runs in O(n log n) and algorithm B runs in $O(n^2)$. Based on these facts, there must be some N that when n>N, algorithm A runs faster than algorithm B.

[TRUE/FALSE]
The following recurrence equation $T(n)=3T(n/3) + 0.1\ n$ has the solution:
$T(n)= \Theta\ (n \log(n))$.

[TRUE/FALSE]
Every problem in NP can be solved in exponential time by a deterministic Turing machine

[TRUE/FALSE]
In Kruskal's MST algorithm, if we choose edges in decreasing (instead of increasing) order of cost, we will end up with a spanning tree of maximum total cost

[TRUE/FALSE]
If all edges in a graph have capacity 1, then Ford-Fulkerson runs in linear time.

[TRUE/FALSE]
If problem X can be solved using dynamic programming, then X belongs to P.

[TRUE/FALSE]
If Vertex-Cover $\in$ P then SAT $\in$ P.

[TRUE/FALSE]
Assuming P!=NP, and X is a problem belonging to class NP. There is no polynomial time algorithm for X.

1) 20 pts
Mark the following statements as **TRUE** or **FALSE**. No need to provide any justification.

[TRUE/FALSE]
Every problem in P can be reduced to 3-SAT in polynomial time.

[TRUE/FALSE]
If there is a polynomial-time algorithm for 2-SAT, then every problem in NP has a polynomial-time algorithm.

[TRUE/FALSE]
If all edge weights are 1, 2, or 3, the shortest path problem can be solved in linear time.

[TRUE/FALSE]
Suppose G is a graph with n vertices and $n^{1.5}$ edges, represented in adjacency list representation. Then depth-first search in G runs in $O(n^{1.5})$ time.

[TRUE/FALSE]
The weight of a minimum spanning tree in a positively weighted undirected graph is always less than the total weight of a minimum spanning path (Hamiltonian Path with lowest weight) of the graph.

[TRUE/FALSE]
If A is in NP, and B is NP-complete, and A $\leq_p$ B then A is NP-complete.

[TRUE/FALSE]
Given a problem B, if there exists an NP-complete problem that can be reduced to B in polynomial time, then B is NP-complete.

[TRUE/FALSE]
If an undirected connected graph has the property that between any two nodes u and v, there is exactly one path between u and v, then that graph is a tree.

[TRUE/FALSE]
Suppose that a divide and conquer algorithm reduces an instance of size n to four instances of size $n/5$ and spends $\Theta(n)$ time in the divide and combine steps. The algorithm runs in $\Theta(n)$ time.

[TRUE/FALSE]
An integer N is given in binary. An algorithm that runs in time $O(sqrt(N))$ to find the largest prime factor of N is considered to be a polynomial-time algorithm.
Ex: Prime factorization of 84: 2 x 2 x 3 x 7, so the largest prime factor of 84 is 7

1) 20 pts
Mark the following statements as **TRUE** or **FALSE**. No need to provide any justification.

TRUE/FALSE]
Given a minimum cut, we could find the maximum flow value in O(E) time.

TRUE/FALSE]
Any NP-hard problem can be solved in time O(2^poly(n)), where n is the input size and poly(n) is a polynomial.

TRUE/FALSE]
Any NP problem can be solved in time O(2^poly(n)), where n is the input size and poly(n) is a polynomial.

TRUE/FALSE]
If 3-SAT $\leq_p$ 2-SAT, then P = NP.

TRUE/FALSE]
Assuming P ≠ NP, there can exist a polynomial-time approximation algorithm for the general Traveling Salesman Problem.

TRUE/FALSE]
Let (S,V−S) be a minimum (s,t)-cut in the network flow graph G. Let (u,v) be an edge that crosses the cut in the forward direction, i.e., u ∈ S and v ∈ V−S. Then increasing the capacity of the edge (u, v) necessarily increases the maximum flow of G.

TRUE/FALSE]
If problem X can be solved using dynamic programming, then X belongs to P.

TRUE/FALSE]
All instances of linear programming have exactly one optimal solution.

TRUE/FALSE]
Let Y $\leq_p$ X and there exists a 2-approximation for X, then there must exist a 2-approximation for Y.

TRUE/FALSE]
There is no known polynomial-time algorithm to solve an integer linear programming.

1) 20 pts
Mark the following statements as **TRUE** or **FALSE**. No need to provide any justification.

TRUE/FALSE]
Every problem in NP has a polynomial time certifier.

[TRUE/FALSE]
Every decision problem is in NP-complete.

TRUE/FALSE]
An NP problem is NP-complete if 3-SAT reduces to it in polynomial time.

[TRUE/FALSE]
If all edges in a graph have capacity 1, then Ford-Fulkerson runs in linear time.

TRUE/FALSE]
Let T be a minimum spanning tree of G. Then, for any pair of vertices s and t, the shortest s-t path in G is the path in T.

TRUE/FALSE]
If the running time of a divide-and-conquer algorithm satisfies the recurrence $T(n) = 3 T(n/2)+ \Theta(n^2)$, then $T(n) = \Theta(n^2)$.

TRUE/FALSE]
In a divide and conquer solution, the sub-problems are disjoint and are of the same size.

TRUE/FALSE]
All Integer linear programming problems can be solved in polynomial time.

TRUE/FALSE]
If the linear program is feasible and bounded, then there exists an optimal solution.

TRUE/FALSE]
Suppose we have a data structure where the amortized running time of Insert and Delete is O(lg n). Then in any sequence of 2n calls to Insert and Delete, the worst-case running time for the nth call is O(lg n).

1) 20 pts
Mark the following statements as **TRUE**, **FALSE,** or **UNKNOWN**. No need to provide any justification.

[[✓ TRUE]/FALSE/UNKNOWN]
If $X \leq_p Y$, and X is NP-complete, then Y is NP-hard.

[TRUE/[✓ FALSE]/UNKNOWN]
If $X \leq_p Y$, and X is NP-complete, then Y is NP-complete.

[TRUE/[✓ FALSE]/UNKNOWN]
If $X \leq_p$ Integer Programming, then X is NP-hard.

[[✓ TRUE]/FALSE/UNKNOWN]
If $X \leq_p$ Linear Programming, then X is in P.

[TRUE/FALSE/[✓ UNKNOWN]]
3-SAT cannot be solved in polynomial time.

[[✓ TRUE]/FALSE/UNKNOWN]
If graph G has no cycles, then the independent set problem in G can be solved in polynomial time.

[TRUE/[✓ FALSE]]
Although the general Travelling Salesman Problem is NP-complete, in class, we presented a 2-approximation algorithm for it that runs in polynomial time.

[TRUE/[✓ FALSE]]
Breadth first search is an example of a divide-and-conquer algorithm.

[[✓ TRUE]/FALSE]
Memoization requires memory space which is linear in size with respect to the number of unique sub-problems.

[[✓ TRUE]/FALSE]
The smallest element in a binary max-heap of size *n* can be found with at most *n/2* comparisons.

Exam 3 2015 Fall

1) 20 pts
Mark the following statements as **TRUE**, **FALSE**. No need to provide any justification.

[TRUE/FALSE]
If P = NP, then all NP-Hard problems can be solved in Polynomial time.

[TRUE/FALSE]
Dynamic Programming approach only works when used on problems with non-overlapping sub problems.

[TRUE/FALSE]
In a divide & conquer algorithm, the size of each sub-problem must be at most half the size of the original problem.

[TRUE/FALSE]
In a 0-1 knapsack problem, a solution that uses up all of the capacity of the knapsack will be optimal.

[TRUE/FALSE]
If a problem X can be reduced to a known NP-hard problem, then X must be NP-hard.

[TRUE/FALSE]
If SAT $\leq_P$ A, then A is NP-hard.

[TRUE/FALSE]
The recurrence $T(n) = 2T(n/2) + 3n$, has solution $T(n) = \theta(n\log(n^2))$.

[TRUE/FALSE]
Consider two positively weighted graphs $G_1 = (V,E,w_1)$ and $G_1 = (V,E,w_2)$ with the same vertices V and edges E such that, for any edge $e \in E$, we have $w_2(e) = (w_1(e))^2$ For any two vertices $u,v \in V$, any shortest path between u and v in G_2 is also a shortest path in G_1.

[TRUE/FALSE]
If an undirected graph G=(V,E) has a Hamiltonian Cycle, then any DFS tree in G has a depth |V| - 1.

[TRUE/FALSE]
Linear programming is at least as hard as the Max Flow problem.

1) 20 pts
Mark the following statements as **TRUE** or **FALSE**. No need to provide any justification.

TRUE/FALSE]
If all edge capacities in a flow network are integer multiples of 7, then the maximum value of flow is a multiple of 7.

TRUE/FALSE]
If P = NP, then all NP-Hard problems can be solved in Polynomial time.

TRUE/FALSE]
Let T be a complete binary tree with n nodes. Finding a path from the root of T to a given vertex v ∈ T using breadth-first search takes O(log n) time.

TRUE/FALSE]
Halting Problem is an NP-Hard problem.

TRUE/FALSE]
Every decision problem in P has a polynomial time certifier.

TRUE/FALSE]
In a flow network, if we increase the capacity of an edge that happens to be on a minimum cut, this will increase the max flow in the network.

TRUE/FALSE]
If the capacity of every arc is odd, then there is a maximum flow in which the flow on each arc is odd.

TRUE/FALSE]
If the edge weights of a weighted graph are doubled, then the number of minimum spanning trees of the graph remains unchanged

TRUE/FALSE]
The linear programming solution to the shortest path problem discussed in class can fail in the presence of negative cost edges.

TRUE/FALSE]
In a divide and conquer solution, the sub-problems are disjoint and are of the same size.

Exam 3 2015 Spring

1) 20 pts
Mark the following statements as **TRUE** or **FALSE**. No need to provide any justification.

[TRUE/FALSE]
If SAT $\leq_P$ A, then A is NP-hard.

[TRUE/FALSE]
If a problem X can be reduced to a known NP-hard problem, then X must be NP-hard.

[TRUE/FALSE]
If P equals NP, then NP equals NP-complete.

[TRUE/FALSE]
Let X be a decision problem. If we prove that X is in the class NP and give a poly-time reduction from X to Hamiltonian Cycle, we can conclude that X is NP-complete.

[TRUE/FALSE]
The recurrence $T(n) = 2T(n/2) + 3n$, has solution $T(n) = \theta(n \log(n^2))$.

[TRUE/FALSE]
On a connected, directed graph with only positive edge weights, Bellman-Ford runs asymptotically as fast as Dijkstra.

[TRUE/FALSE]
Linear programming is at least as hard as the Max Flow problem in a flow network.

[TRUE/FALSE]
If you are given a maximum s-t flow in a graph then you can find a minimum s-t cut in time O(m) where m is the number of the edges in the graph.

[TRUE/FALSE]
Fibonacci heaps can be used to make Dijkstra's algorithm run in O(|E| + |V| log|V|) time on a graph G=(V,E)

[TRUE/FALSE]
A graph with non-unique edge weights will have at least two minimum spanning trees

Exam 3 2014 Fall

1) 20 pts
Mark the following statements as **TRUE** or **FALSE**. No need to provide any justification.

[TRUE/FALSE]
All the NP-hard problems are in NP.

[TRUE/FALSE]
Given a weighted graph and two nodes, it is possible to list all shortest paths between these two nodes in polynomial time.

[TRUE/FALSE]
In the memory efficient implementation of Bellman-Ford, the number of iterations it takes to converge can vary depending on the order of nodes updated within an iteration

[TRUE/FALSE]
There is a feasible circulation with demands $\{d_v\}$ if $\sum_v d_v = 0$.

[TRUE/FALSE]
Not every decision problem in P has a polynomial time certifier.

[TRUE/FALSE]
If a problem can be reduced to linear programming in polynomial time then that problem is in P.

[TRUE/FALSE]
If we can prove that P $\neq$ NP, then a problem A $\in$ P does not belong to NP.

[TRUE/FALSE]
If all capacities in a flow network are integers, then every maximum flow in the network is such that flow value on each edge is an integer.

[TRUE/FALSE]
In a dynamic programming formulation, the sub-problems must be mutually independent.

[TRUE/FALSE]
In the final residual graph constructed during the execution of the Ford–Fulkerson Algorithm, there’s no path from sink to source.

1) 20 pts
Mark the following statements as **TRUE** or **FALSE**. No need to provide any justification.

[TRUE/FALSE]
Let A and B be decision problems. If A is polynomial time reducible to B and B is in NP-Complete, then A is in NP.

[TRUE/FALSE]
In a network with source s and sink t where each edge capacity is a positive integer, there is always a max s-t flow where the flow assigned to each edge is an integer.

[TRUE/FALSE]
Let ODD denote the problem of deciding if a given integer is odd. Then ODD is polynomial time reducible to 3-SAT.

[TRUE/FALSE]
Not every decision problem in P has a polynomial time certifier.

[TRUE/FALSE]
The set of all vertices in a graph is a vertex cover.

[TRUE/FALSE]
A minimum spanning tree of a connected undirected graph remains being a minimum spanning tree even if each edge weight is doubled.

[TRUE/FALSE]
A minimum spanning tree of a bipartite graph is not necessarily a bipartite graph.

[TRUE/FALSE]
Dijkstra's algorithm can always find the shortest path between two nodes in a graph as long as there is no negative cost cycle in the graph.

[TRUE/FALSE]
Given a binary max heap of size *n*, the complexity of finding the smallest number in the heap is O(log *n*).

[TRUE/FALSE]
Given a graph G=(V,E) and an approximation algorithm that solves the vertex cover problem in G with an approximation ratio *r*, then this algorithm can also provide a solution to the independent set of G with the same approximation ratio *r*.

1) 20 pts
Mark the following statements as **TRUE** or **FALSE**. No need to provide any justification.

[TRUE/FALSE]
Assume P !=NP. Let A and B be decision problems. If A is in NP-Complete and $A \leq_P B$, then B is not in P.

[TRUE/FALSE]
There exists a decision problem X such that for all Y in NP, Y is polynomial time reducible to X.

[TRUE/FALSE]
If P equals NP, then NP equals NP-complte.

[TRUE/FALSE]
The running time of a dynamic programming algorithm is always theta(P) where P is the number of sub-problems.

[TRUE/FALSE]
A spanning tree of a given undirected, connected graph G=(V,E) can be found in O(| E|) time.

[TRUE/FALSE]
To find the minimum element in a max heap of n elements, it takes O(n) time

[TRUE/FALSE]
Kruskal's algorithm for finding the MST works with positive and negative edge weights.

[TRUE/FALSE]
If a problem is not in P, then it must be in NP.

[TRUE/FALSE]
If an NP-complete problem can be solved in linear time, then all NP-complete problems can be solved in linear time.

[TRUE/FALSE]
Linear programming problems can be solved in polynomial time

2 hr exam
Close book and notes

1) 20 pts
Mark the following statements as **TRUE** or **FALSE**. No need to provide any justification except for the question at the bottom of the page.

[TRUE/FALSE]
In a flow network, if all edge capacities are distinct, then the max flow of this network is unique.

[TRUE/FALSE]
To find the minimum element in a max heap of n elements, it takes O(n) time.

[TRUE/FALSE]
Let T be a spanning tree of graph G(V, E), let k be the number of edges in T, then k=O(V)

[TRUE/FALSE]
Linear programming problems can be solved in polynomial time.

[TRUE/FALSE]
Consider problem A: given a flow network, find the maximum flow from a node s to a node t. problem A is in NP.

[TRUE/FALSE]
Given n numbers, it takes O(n) time to construct a binary min heap.

[TRUE/FALSE]
Kruskal's algorithm for finding the MST works with positive and negative edge weights.

[TRUE/FALSE]
Breadth first search is an example of a divide-and-conquer algorithm.

[TRUE/FALSE]
If a problem is not in P, then it must be in NP.

[TRUE/FALSE]
L1 can be reduced to L2 in Polynomial time and L1 is in NP, then L2 is in NP

1) 20 pts
Mark the following statements as **TRUE**, **FALSE, or UNKOWN**. No need to provide any justification.

[TRUE/FALSE]
Given a network *G(V, E)* and flow *f,* and the residual graph G_f(V', E'), then |V|=|V'| and 2|E|>=|E'|.

[TRUE/FALSE]
The Ford-Fulkerson Algorithm terminates when the source *s* is not reachable from the sink *t* in the residual graph.

[TRUE/FALSE/UNKOWN]
NP is the class of problems that are not solvable in polynomial time.

[TRUE/FALSE/UNKOWN]
If problem A is NP complete, and problem B can be reduced to problem A in quadratic time. Then problem B is also NP complete

[TRUE/FALSE/UNKOWN]
If X can be reduced in polynomial time to Y and Z can be reduced in polynomial time to Y, then X can be reduced in polynomial time to Z.

[TRUE/FALSE]
Let G(V,E) be a weighted graph and let T be a minimum spanning tree of G obtained using Prim's algorithm. The path in T between *s* (the root of the MST) and any other node in the tree must be a shortest path in G.

[TRUE/FALSE]
DFS can be used to find the shortest path between any two nodes in a non-weighted graph.

[TRUE/FALSE]
The Bellman-Ford algorithm cannot be parallelized if there are negative cost edges in the network.

[TRUE/FALSE]
A perfect matching in a bipartite graph can be found using a maximum-flow algorithm.

[TRUE/FALSE]
Max flow in a flow network with integer capacities can be found exactly using linear programming.

1) 10 pts
For each of the following sentences, state whether the sentence is known to be TRUE, known to be FALSE, or whether its truth value is still UNKNOWN.

(a) If a problem is in P, it must also be in NP.

(b) If a problem is in NP, it must also be in P.

(c) If a problem is NP-complete, it must also be in NP.

(d) If a problem is NP-complete, it must not be in P.

(e) If a problem is not in P, it must be NP-complete.

If a problem is NP-complete, it must also be NP-hard.

If a problem is in NP, it must also be NP-hard.

If we find an efficient algorithm to solve the Vertex Cover problem we have proven that P=NP

If we find an efficient algorithm to solve the Vertex Cover problem with an approximation factor $\rho \geq 1$ (a single constant) then we have proven that P=NP

If we find an efficient algorithm that takes as input an approximation factor $\rho \geq 1$ and solves the Vertex Cover problem with that approximation factor, we have proven that P=NP.

1) 20 pts
Mark the following statements as **TRUE** or **FALSE**. No need to provide any justification.

[TRUE/FALSE]
If NP = P, then all problems in NP are NP hard

[TRUE/FALSE]
L1 can be reduced to L2 in Polynomial time and L2 is in NP, then L1 is in NP

[TRUE/FALSE]
The simplex method solves Linear Programming in polynomial time.

[TRUE/FALSE]
Integer Programming is in P.

[TRUE/FALSE]
If a linear time algorithm is found for the traveling salesman problem, then every problem in NP can be solved in linear time.

[TRUE/FALSE]
If there exists a polynomial time 5-approximation algorithm for the general traveling salesman problem then 3-SAT can be solved in polynomial time.

[TRUE/FALSE]
Consider an undirected graph G=(V, E). Suppose all edge weights are different. Then the longest edge cannot be in the minimum spanning tree.

[TRUE/FALSE]
Given a set of demands D = *{dv}* on a directed graph *G(V,E)*, if the total demand over *V* is zero, then *G* has a feasible circulation with respect to *D.*

[TRUE/FALSE]
For a connected graph G, the BFS tree, DFS tree, and MST all have the same number of edges.

[TRUE/FALSE]
Dynamic programming sub-problems can overlap but divide and conquer sub-problems do not overlap, therefore these techniques cannot be combined in a single algorithm.

1) 20 pts
For each of the following statements, answer whether it is TRUE or FALSE, and briefly justify your answer.

a) If a connected undirected graph G has the same weights for every edge, then every spanning tree of G is a minimum spanning tree, but such a spanning tree cannot be found in linear time.

b) Given a flow network G and a maximum flow of G that has already been computed, one can compute a minimum cut of G in linear time.

c) The Ford-Fulkerson Algorithm finds a maximum flow of a unit-capacity flow network with n vertices and m edges in time O(mn) if one uses depth-first search to find an augmenting path in each iteration.

d) Unless P = NP, 3-SAT has no polynomial-time algorithm.

e) The problem of deciding whether a given flow f of a given flow network G is a maximum flow can be solved in linear time.

f) If a decision problem A is polynomial-time reducible to a decision problem B (i.e., $A \leq_p B$), and B is NP-complete, then A must be NP-complete.

g) If a decision problem B is polynomial-time reducible to a decision problem A (i.e., $B \leq_p A$), and B is NP-complete, then A must be NP-complete.

h) Integer max flow (where flows and capacities are integers) is polynomial time reducible to linear programming .

i) It has been proved that NP-complete problems cannot be solved in polynomial time.

j) NP is a class of problems for which we do not have polynomial time solutions.

1) 20 pts
Mark the following statements as **TRUE**, **FALSE, or UNKNOWN**. No need to provide any justification.

[TRUE/FALSE]
In a flow network whose edges have capacity 1, the maximum flow always corresponds to the maximum degree of a vertex in the network**.**

[TRUE/FALSE]
If all edge capacities of a flow network are unique, then the min cut is also unique.

[TRUE/FALSE]
A minimum weight edge in a graph G must be in one minimum spanning tree of G.

[TRUE/FALSE]
When the size of the input grows, any polynomial algorithm will eventually become more efficient than any exponential one.

[TRUE/FALSE/UNKNOWN]
NP is the class of problems that are not solvable in polynomial time.

[TRUE/FALSE/UNKNOWN]
If a problem is not solvable in polynomial time, it is in the NP-Complete class.

[TRUE/FALSE/UNKNOWN]
Linear programming can be solved in polynomial time.

[TRUE/FALSE]
$10^{2 \log 4n+3} + 9^{2 \log 3n+21}$ is O(n).

[TRUE/FALSE]
f(n) = O(g(n)) implies g(n) = O(f(n)).

[TRUE/FALSE]
If X can be reduced in polynomial time to Y and Z can be reduced in polynomial time to Y , then X can be reduced in polynomial time to Z.

CS 570
Analysis of Algorithms
Summer 2007
Final Exam Solutions

Kenny Daniel (kfdaniel@usc.edu)

Question 1

[FALSE] If A is linear time reducible to B ($A \leq B$), and B is NP-complete, then A must be NP-complete.

[FALSE] If B is linear time reducible to A ($B \leq A$), and B is NP-complete, then A must be NP-complete.

[TRUE] If any integer programming optimization problem can be converted in polynomial time to an equivalent linear programming problem, then $P = NP$.

[FALSE] It has been determined that NP Complete problems cannot be solved in polynomial time.

[FALSE] If $P = NP$, then there are still some NP complete problems that cannot be solved in polynomial time.

[TRUE] When we say that a problem X is NP Complete, then it means that every NP complete problem can be reduced to X.