CS570 Spring 2018: Analysis of Algorithms Exam III

	Points		Points
Problem 1	20	Problem 5	15
Problem 2	15	Problem 6	10
Problem 3	15	Problem 7	10
Problem 4	15		
Total: 100 Points			

Instructions:

1. This is a 2-hr exam. Closed book and notes
2. If a description to an algorithm or a proof is required please limit your description or proof to within 150 words, preferably not exceeding the space allotted for that question.
3. No space other than the pages in the exam booklet will be scanned for grading.
4. If you require an additional page for a question, you can use the extra page provided within this booklet. However please indicate clearly that you are continuing the solution on the additional page.
5. Do not detach any sheets from the booklet. Detached sheets will not be scanned.
6. If using a pencil to write the answers, make sure you apply enough pressure so your answers are readable in the scanned copy of your exam.
7. Do not write your answers in cursive scripts.

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

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

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

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

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

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

[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]
A linear program can have an infinite number of optimal solutions.

[TRUE]
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 $\Omega(n^2)$.

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

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

2) 15 pts

Consider a uniform hash function h that evenly distributes n integer keys $\{0, 1, 2, \ldots, n\text{-}1\}$ among m buckets, where $m < n$. Several keys may be mapped into the same bucket. The uniform distribution formally means that $\Pr(h(x)=h(y)) = 1/m$ for any $x, y \in \{0, 1, 2, \ldots, n\text{-}1\}$. What is the expected number of total collisions? Namely how many distinct keys x and y do we have such that $h(x) = h(y)$?

Solution:

Let the indicator variable X_{ij} be 1 if $h(k_i) = h(k_j)$ and 0 otherwise for all $i \neq j$. Then let X be a random variable representing the total number of collisions. It can be calculated as the sum of all the X_{ij}'s: $X = \sum_{i<j} X_{ij}$

Now we take the expectation and use linearity of expectation to get:

$$E[X] = E\left[\sum_{i<j} X_{ij}\right] = \sum_{i<j} E[X_{ij}] = \sum_{i<j} P(h(k_i) = h(k_j)) = \sum_{i<j} \frac{1}{m} = \frac{n(n-1)}{2m}$$

Rubrics:

Describe choosing 2 keys from n keys: 10 pts.

Describe $\sum_{i<j} P\left(h(k_i) = h(k_j)\right)$: 10 pts.

Common mistaks:

1. n/m: The mistake is that it thinks the expected number of collisions for each key is 1/m. 7pts.
2. 2n/m: Similar to 1. 7pts.
3. (n-1)/m: Similar to 1. 7pts.
4. (m+1)/2: Basically, no clue. 3 pts.
5. n(n-1)/m: Duplicates. 13 pts.
6. n^2/m: duplicated. 13 pts
7. (1-(1-1/m)^(n-1))*n: wrong approach. 5 pts.
8. n-m: wrong approach. 3pts.
9. $\frac{1}{m}\sum_{i=0}^{n-1} i$: 15 pts
10. n(n-1)/2: The mistake is that it does not consider collision probability. 10 pts.
11. N(n+1)/2m: minor mistake. 13 pts.