CS5800: Algorithms Spring 2014	Feb. 25
Midterm	

Instructions.

- This is a 180 (or more) minutes test. There are six questions totaling 130 points.
- It is very important to justify your answers, because (1) partial credit will be given for correct arguments even if the final answer is incorrect, and (2) correct final answers with no arguments are worth very little.
- When a question asks for an algorithm you are expected to develop an algorithm or give a reduction to a problem we discussed in class. In both cases bound the running time of the algorithm using the Landau notation. Write your algorithm in pseudo code. Most importantly you are expected to find an efficient algorithm.
- Please dont write more than one problem on the same sheet of paper, it makes grading easier.

Question	**Your Score**	**Max**
1 Debate Evaluation		40
2 Search Triples		20
3 Billboards		30
4 Order of Growth		10
5 Binomial Coefficients		30

Name: _____

1 Debate Evaluation Algorithm [40 points]

You have been playing debates (semantic games) and you have recorded your debate results in tables such as the following one. You have column headers **PW**, **PL**, **Forced**. There is an additional column *Fault* which is computed from the first three. Each row of the table represents a debate g. **PW** and **PL** are the columns for the two participants. Because there is always a winner (there are no draws) we choose the **PW** column to show the winner. The **Forced** column shows which participant is forced. Remember that at most one is forced. We use entry none indicating that none of the two participants was forced. The *Fault* column depends on the columns **PL** and **Forced** and indicates whether the participant in column **PL** had a fault (lost in non-forced position).

We use the following Boolean functions for a debate g: g.participates(p) = p is a participant in debate g (either the winner or the loser). g.wins(p) = g.participates(p) and participant p wins in debate g. g.loses(p) = g.participates(p) and participant p loses in debate g. g.forced(p) = g.participates(p) and participant p is forced in debate g. g.fault(p) = g.participates(p) and participant p makes a fault in debate g. Note that !g.wins(p) = !g.participates(p) or g.loses(p).

Definition of fault: g.fault(p) = g.participates(p) and g.loses(p) and !g.forced(p).

Similar to the fault concept there is the control concept.

Definition of control: g.control(p) = g.participates(p) and (g.wins(p) or !g.forced(p)). Informally, a participant p is not in control in a debate if p is not involved in the debate or p loses while forced. Again informally we say that a participant p controls a debate if p either wins or p had a guaranteed opportunity to win (was non-forced). In a debate there is at least one of the two participants in control: the winner.

Consider algorithms that translate a debate table into a ranking of the participants. What is a ranking? A reflexive, transitive and complete binary relation.

Informally, our goal is to find the best participants based on the debates. We consider two ranking algorithms.

The first ranking algorithm, called $\leq_{WC}$, is based on win counting while the second, called $\leq_{FC}$ is based on a special kind of loss counting, called fault counting.

We fix a set of debates D. $x \leq^D y$ means that participant x is ranked weakly better than participant y for D. Sometimes we omit the superscript D when it is clear from context. Informally, weakly better means that either x is better then y or x and y have the same rank. $wins(x)$ is the number of wins of participant x over all debates in D. Formally, $wins(x)$ is the number of rows $g \in D$ which satisfy $g.wins(x)$. $faults(x)$ is the number of faults that x makes over all debates in D. Formally, $faults(x)$ is the number of rows g in D which satisfy $g.fault(x)$.

In principle, all the functions defined above would require a superscript for D. For example,

$wins^D(x)$ is the number of wins of participant x over all debates in D. However, we omit it in order not to clutter the definitions.

1.1 Algorithms to Analyze

$\leq_{WC}$ is **defined** as: $x \leq_{WC} y$ if $wins(x) \geq wins(y)$.

$\leq_{FC}$ is **defined** as: $x \leq_{FC} y$ if $faults(x) \leq faults(y)$.

We want to know whether the algorithms $\leq_{WC}$ and $\leq_{FC}$ are collusion-resilient.

1.2 Property to Check

Definition of collusion resilience: A ranking $\leq$ of participants is said to be collusion-resilient if for all debate tables D and for any two participants x and y, the property $x \leq^D y$ implies $x \leq^E y$, where E is D with added debates g that x cannot control, i.e., for which !g.control(x).

Informally, if $x \leq^D y$ and more debates that x cannot control are added to D resulting in E, then $x \leq^E y$ holds.

1.3 Choose your side and defend it

1.3.1 Question 1

Is $\leq_{WC}$ collusion resilient?

1.3.2 Question 2

Is $\leq_{FC}$ collusion resilient?

Justify your answer by providing a proof why the algorithm has the property or not.

Hint: If you look for a positive answer, the proof is rather short. If you look for a negative answer, it is enough to consider a small number of participants only.

2 Searching for Triples (20 points)

Problem to solve:

Find 3 numbers with sum 5 from a list of n integers. More precisely, given a list of n integers, you have to find three numbers a,b,c in the list such that a+b+c=5 or prove that such a triple does not exist.

2.1 Use Divide and Conquer

You must use Divide and Conquer although there are other solutions.

Divide the problem into subproblems which you solve recursively. Combine the solutions of the subproblems.

2.2 Recurrence Relation

Write the recurrence relation for your divide and conquer algorithm and solve it.

Turn in your algorithm and its analysis. How does your algorithm compare with a brute-force algorithm?

3 Placing Billboards (30 points)

You are managing the construction of billboards on a west-east highway that is M miles long. The possible sites for billboards are given by numbers $x_1, ..., x_n$ each in the interval $[0, M]$ (specifiying their position along the highway, measured in kilometers from the western end). If you place a billboard at location x_i, you receive revenure r_i. The values $r_1, ..., r_n$ are given.

Regulations imposed by the highway department impose that no two of the billboards be within less than 5 kilometers of each other. You would like to place billboards at a subset of the sites so as to maximize your total revenue, subject to this restriction.

Develop an algorithm.

4 Order of Growth [10 points]

Give the asymptotic growth rate of the solution for the following recurrences.

(a) (5 points) $T(n) = 7T(n/7) + n$

(b) (5 points) $T(n) = 8T(n/3) + n^2$

5 Binomial Coefficients [20 points]

As you know, "n choose k" or $\binom{n}{k}$ is the number of possibilities of picking k items out of n, or the number of subsets with k elements of a set with n elements. $\binom{n}{0} = \binom{n}{n} = 1$ since there is only one subset with 0 elements and only one subset with all elements.

5.1 Recurrence

Compute $\binom{n}{k}$ **without using the factorials**, by showing optimal sub-structure (proof required) and then use dynamic programming bottom up computation. What is the running time?

5.2 Connections to HSR

What are the connections between binomial coefficients and HSR, the highest safe rung problem? Give at least one.

5.3 Reduction

Can you solve HSR (just finding the values, not the decision trees) in terms of binomial coefficients?

.