

Midterm Exam. Due March 25th 12:00:00 (Noon) Eastern Time

This is a take-home exam. You have 24 hours to finish it.

You must type your answers. When designing algorithms, you must first describe them in English, and then present pseudo code. Complete as many problems as you can, and justify your solutions (i.e. justify the correctness of your algorithms and provide detailed analysis of their running time). Show your work, as partial credit may be given. You will be graded not only on the correctness of your solutions, but also on the clarity you express them.

You must complete the exam on your own. You may, however, consult the Corman-Leiserson-Rivest-Stein textbook, notes you took in class, and homework solutions posted at the TA's web site. No other references of any form or any kind are allowed.

You must begin with the following statement in your file: **I assert that I have abided by the Honor Code.** Including this statement represents your signature of this statement. **Note:** If this statement is missing, your exam will not be graded.

How to submit your exam? Use the `submit` command on `mercury.cs.uml.edu`, for it provides SSH protection of your submission during the transmission. Use the following command line:

```
submit wang 503midterm <your exam file>
```

If you do not have an account on `mercury.cs.uml.edu`, you may email your exam file to my address at `wang@cs.uml.edu`. The time stamp of your email message will be used to determine whether your submission is in time.

-
1. (25 points) Suppose there are three groups A , B , and C of people with $|A| = \ell$, $|B| = m$, and $|C| = n$. Suppose each person in A knows a few persons (maybe none) in B and each person B knows a few persons (maybe none) in C . A matching is a set of triples (a, b, c) , where $a \in A$, $b \in B$, and $c \in C$ such that a knows b and b knows c ; moreover, each person may only appear at most once in the matching. Design an efficient algorithm using network flows to find a matching with the largest triples. Justify your solution.
 2. (25 points) Ms. Smart has n types of blocks and an unlimited supply of blocks of each type. Each type- i block is a rectangular solid with linear dimensions $\langle x_i, y_i, z_i \rangle$. A block can be oriented so that any two of its three dimensions determine the dimensions

of a base and the other dimension is the height. These blocks may be used to build a tower. To place one block on top of another block, the two dimensions of the base of the upper block must be each strictly less than the corresponding base dimensions of the lower block. That is, larger or equal-sized bases cannot be stacked. Ms. Smart wants to construct the tallest tower possible out of these blocks, and she uses DAG-SHORTEST-PATHS technique to come up with an efficient algorithm to determine the tallest tower she can build. You are asked to design such an algorithm using the same technique and analyze its running time. Justify your solution.

3. (25 points) Show how to implement a queue with two ordinary stacks so that the amortized cost of performing n ENQUEUE and DEQUEUE operations is $O(n)$. Justify your solution.
4. (25 points) Summation of floating-point numbers is ubiquitous. But adding two floating-point numbers using a computer may incur a roundoff error bounded by α , where α is a small positive value. Thus, the result of adding two real numbers x and y is

$$\text{fl}(x + y) = (x + y)(1 + \delta_{xy}), \text{ where } |\delta_{xy}| \leq \alpha.$$

Let $X = \{x_1, \dots, x_n\}$ be a multiset of positive real numbers. We want to calculate $S_n = \sum_{i=1}^n x_i$ with an roundoff error as small as possible.

We note that the operator $+$ is applied to two operands at a time, and so an ordering for adding X corresponds to a binary addition tree of n leaves and $n - 1$ internal nodes, where a leaf is an x_i and an internal node is the sum of its two children. Different orderings yield different addition trees, which may produce different computed sums \hat{S}_n . Thus, our task is to find an optimal ordering that minimizes the error $E_n = |\hat{S}_n - S_n|$. Let I_1, \dots, I_{n-1} be the internal nodes of an addition tree T over X . Since α is very small (even on a 16-bit computer), any product of more than one δ is negligible. Thus, we may simply assume that $E_n \leq \alpha \sum_{i=1}^{n-1} I_i$. This means that finding the optimal ordering with the smallest E_n is approximately equal to finding an addition tree T such that its cost $C(T) = \sum_{i=1}^{n-1} I_i$ is minimum. Design a $O(n \log n)$ -time greedy algorithm to find such an addition tree T . Justify your solution.

5. (25 points) In a ceremony for a sports event, n members from the blue team need to march side by side with n members from the red team, where the red team has m ($m > n$) members to choose from. To make it look good, each pair of a blue team member and a red team member should have the same height, or the difference between their heights should be small. Design an efficient algorithm using dynamic programming to select n members from the red team to pair with the n members in the blue team so that the summation of the absolute height difference in each pair is minimum. Justify your solution. (**Hint:** First show that cross pairings have no advantage. That is, if $b_i < b_j$ and $r_{i'} < r_{j'}$, then pairing b_i with $r_{i'}$ and pairing b_j with $r_{j'}$ are at least as good as any other pairings.)