CS 573: Graduate Algorithms, Fall 2010
Homework 2

Due Monday, September 27, 2010 at 5pm
(in the homework drop boxes in the basement of Siebel)

- For this and all future homeworks, groups of up to three students may submit a single, common solution. Please neatly print (or typeset) the full name and NetID of every group member on the first page of your submission.
- We will use the following rubric to grade all dynamic programming algorithms:
  - 60% for a correct recurrence (including base cases and a plain-English specification); no credit for anything else if this is wrong.
  - 10% for describing a suitable memoization data structure.
  - 20% for describing a correct evaluation order. (A clear picture is sufficient.)
  - 10% point for analyzing the running time of the resulting algorithm.

  Official solutions will always include pseudocode for the final dynamic programming algorithm, but this is ***not*** required for full credit. However, if you do provide correct pseudocode for the dynamic programming algorithm, it is not necessary to separately describe the recurrence, the memoization data structure, or the evaluation order.

  It is ***not*** necessary to state a space bound. There is no penalty for using more space than the official solution, but +1 extra credit for using less space with the same (or better) running time.
- The official solution for every problem will provide a target time bound. Algorithms faster than the official solution are worth more points (as extra credit); algorithms slower than the official solution are worth fewer points. For slower algorithms, partial credit is scaled to the lower maximum score. For example, if a full dynamic programming algorithm would be worth 5 points, just the recurrence is worth 3 points. However, incorrect algorithms are worth zero points, no matter how fast they are.
- Greedy algorithms *must* be accompanied by proofs of correctness in order to receive *any* credit. Otherwise, **any correct algorithm, no matter how slow, is worth at least 2½ points**, assuming it is properly analyzed.

---

1. Suppose you are given an array $A[1..n]$ of positive integers. Describe and analyze an algorithm to find the *smallest* positive integer that is *not* an element of A in $O(n)$ time.

2. Suppose you are given an $m \times n$ bitmap, represented by an array $M[1..m, 1..n]$ whose entries are all 0 or 1. A *solid block* is a subarray of the form $M[i..i', j..j']$ in which every bit is equal to 1. Describe and analyze an efficient algorithm to find a solid block in M with maximum area.

3. Let T be a tree in which each edge e has a weight $w(e)$. A *matching* M in T is a subset of the edges such that each vertex of T is incident to at most one edge in M. The weight of a matching M is the sum of the weights of its edges. Describe and analyze an algorithm to compute a maximum weight matching, given the tree T as input.

4. For any string x and any non-negative integer k, let x^k denote the string obtained by concatenating k copies of x. For example, $\texttt{STRING}^3$ =STRINGSTRINGSTRING and $\texttt{STRING}^0$ is the empty string.

   A ***repetition*** of x is a prefix of x^k for some integer k. For example, STRINGSTRINGSTRINGST and STR are both repetitions of STRING, as is the empty string.

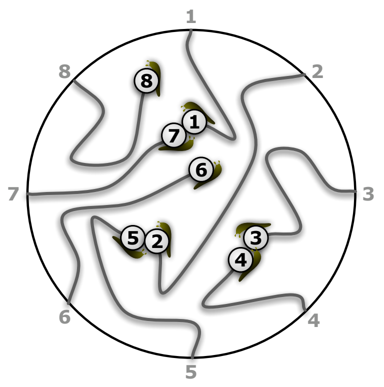
   An ***interleaving*** of two strings x and y is any string obtained by shuffling a repetition of x with a repetition of y. For example, STRWORINDGSTWORIRNGDWSTORR is an interleaving of STRING and WORD, as is the empty string.

   Describe and analyze an algorithm that accepts three strings x, y, and z as input, and decides whether z is an interleaving of x and y.

5. Every year, as part of its annual meeting, the Antarctican Snail Lovers of Upper Glacierville hold a Round Table Mating Race. Several high-quality breeding snails are placed at the edge of a round table. The snails are numbered in order around the table from 1 to n. During the race, each snail wanders around the table, leaving a trail of slime behind it. The snails have been specially trained never to fall off the edge of the table or to cross a slime trail, even their own. If two snails meet, they are declared a breeding pair, removed from the table, and whisked away to a romantic hole in the ground to make little baby snails. Note that some snails may never find a mate, even if the race goes on forever.

   For every pair of snails, the Antarctican SLUG race organizers have posted a monetary reward, to be paid to the owners if that pair of snails meets during the Mating Race. Specifically, there is a two-dimensional array $M[1..n, 1..n]$ posted on the wall behind the Round Table, where $M[i,j] = M[j,i]$ is the reward to be paid if snails i and j meet.

   Describe and analyze an algorithm to compute the maximum total reward that the organizers could be forced to pay, given the array M as input.



The end of a typical Antarctican SLUG race. Snails 6 and 8 never find mates.
The organizers must pay $M[3,4] + M[2,5] + M[1,7]$.