Greedy algorithms/dynamic programming (part 1 of CSC 282),
http://www.cs.rochester.edu/~stefanko/Teaching/19CS282

Homework 1 due: Sep. 12 (Thursday); collected before class in Wegmans 1400.

Problem sessions before Homework 1 is due:

Sep. 9 (Monday) 6:15pm - 7:15pm in Hylan 202
Sep. 10 (Tuesday) 6:15pm - 7:15pm in Goergen 109
Sep. 11 (Wednesday) 6:15pm - 7:15pm in Gavett 310
Spe. 11 (Wednesday) 7:40pm - 8:40pm in Hutchinson 473

IMPORTANT: **Write each problem on a separate sheet of paper. Write your name on each sheet.** (Why? To speed up the grading we will use parallelization—each problem will be graded by a different TA. At the beginning of the class (on the date when the homework is due) there will be a separate pile for each problem.)

---

1 Homework - solve and turn in

There are hints at the end of this assignment. Try to solve the problems without looking at the hints first.

1.1 (**due Sep. 12**) We are given n positive numbers $a_1, \ldots, a_n$. The goal is to select a subset of the numbers with maximal sum and such that no three consecutive numbers are selected. Here are three example inputs together with optimal solutions (the numbers in boxes are selected):

$$\boxed{5}\ \boxed{5}\ 8\ \boxed{5}\ \boxed{5}$$

$$\boxed{5}\ 5\ \boxed{12}\ 5\ \boxed{5}$$

$$1\ \boxed{2}\ \boxed{2}\ 1\ \boxed{2}\ \boxed{1}\ 2\ \boxed{5}\ \boxed{5}$$

Give an $O(n)$-time algorithm for the problem.

1.2 (**due Sep. 12**) We are given an $n \times n$ array A of zeros and ones. We want to find the size of the largest contiguous all-ones square. Give an $O(n^2)$-time algorithm for the problem.

1.3 (**due Sep. 12**) Activity selection problem with two people. We have a set of n activities. For $i \in \{1, \ldots, n\}$ the i-th activity is described by an interval $[a_i, b_i]$ and a value v_i (we assume $a_i \leq b_i$). We have two people that want to pick activities to attend so that

- they attend disjoint sets of activities,
- the activities attended by each person are non-overlapping, and
- the total value of activities attended is maximized.

Formally, we search for $S, Z \subseteq \{1, \ldots, n\}$ such that the following are true.

- $S \cap Z = \emptyset$ ("they attend disjoint sets of activities").
- For $i, j \in S$ such that $i \neq j$ we have $b_i < a_j$ or $b_j < a_i$ ("the activities attended by the first person are non-overlapping").
- For $i, j \in Z$ such that $i \neq j$ we have $b_i < a_j$ or $b_j < a_i$ ("the activities attended by the second person are non-overlapping").

- $\sum_{i \in S} v_i + \sum_{i \in Z} v_i$ is maximized ("the total value of activities attended is maximized").

We will solve the problem using dynamic programming. Assume that the activities are sorted by their end time, that is, $b_1 \le b_2 \le b_3 \le \cdots \le b_n$. For $j, k \in \{1, \ldots, n\}$ let $T[j,k]$ be the maximum value of a valid selection where the last activity for the first person is j and the last activity for the second person is k, that is, we require $\max S = j$ and $\max Z = k$ (if $j = k$ then we let $T[j,k] = -\infty$).

Give an expression (or a piece of code) to compute $T[j,k]$ from smaller subproblems. Argue why your expression is correct.

2 Bonus Homework - solve and turn in

1.4 (**due Sep. 12**) We have an $a \times b$ bar of chocolate (where a, b are integers). By breaking the bar we can either

- create two bars $a_1 \times b$ and $a_2 \times b$ where a_1, a_2 are integers and $a_1 + a_2 = a$, or
- create two bars $a \times b_1$ and $a \times b_2$ where b_1, b_2 are integers and $b_1 + b_2 = b$.

We can further break the resulting bars. Our goal is to 1) end up with bars that are square (that is, have size 1×1, or 2×2, or 3×3, and so on) and 2) minimize the total number of breaks.

For example, if $a = 2$ and $b = 3$ then we use 2 breaks:

$$\begin{aligned} 2 \times 3 \to \\ 2 \times 2 \text{ and } 2 \times 1 \to \\ 2 \times 2 \text{ and } 1 \times 1 \text{ and } 1 \times 1. \end{aligned}$$

For example, if $a = 2$ and $b = 4$ then we use 1 break:

$$\begin{aligned} 2 \times 4 \to \\ 2 \times 2 \text{ and } 2 \times 2. \end{aligned}$$

Give a dynamic programming algorithm which computes a table $T[1..a, 1..b]$ where $T[x,y]$ contains the minimal number of breaks to "squareize" an $x \times y$ bar.

3 Hints

Problem 1.1; **Solution 1:** For $k \in \{1, \ldots, n\}$ let $T[k]$ be the maximum value of a valid selection of numbers $a_1, \ldots, a_k$. We will compute $T[1], T[2], \ldots, T[n]$ in this order. Show how $T[k]$ can be quickly computed from $T[1], \ldots, T[k-1]$ and $a_1, \ldots, a_k$.

Problem 1.1; **Solution 2:** For $k \in \{2, \ldots, n\}$ and $x, y \in \{0, 1\}$ let $T_{xy}[k]$ be the maximum value of a valid selection of numbers $a_1, \ldots, a_k$ where

- a_{k-1} is selected if and only if $x = 1$, and
- a_k is selected if and only if $y = 1$.

We will compute $T_{00}[2], T_{01}[2], T_{10}[2], T_{11}[2], \ldots, T_{00}[n], T_{01}[n], T_{10}[n], T_{11}[n]$.

Problem 1.2 Let $B[i,j]$ be the side-length of the largest square whose bottom-right corner is at position i, j. Compute $B[1,1], B[1,2], \ldots, B[n,n]$. It should take $O(1)$ time to compute each $B[i,j]$ (using values of the previously computed subproblems).

Problem 1.4 We have $T[x,x] = 0$. Here is an incorrect (greedy) way of computing $T[x,y]$ for $x > y$:

$$T[x,y] = 1 + T[x-y, y]. \tag{1}$$

(Find an example where using (1) yields a wrong value for $T[x,y]$.) Don't be afraid to spend more time to compute $T[x,y]$ (any polynomial is OK).