15-351 / 15-650 / 02-613 (Fall 2022): Homework #5

Due: 11:59 PM, Thursday, November 3, 2022
(No late homework assignment will be accepted)

Note #1: You may discuss these problems with your current classmates, but you must write up your solutions independently, without using common notes or worksheets. You must indicate at the top of your homework whom you worked with. Your write-up should be clear, and concise. You are basically trying to convince a skeptical reader that your answers are correct. Your homework should be submitted via Gradescope as a typeset PDF. A LaTeX tutorial and template are available on Piazza (under Resources) if you choose to use that system to typeset (note: you can use Overleaf to do that).

Note #2: For problems asking for an algorithm: describe the algorithm, give an argument why it is correct, and an estimation of how fast it will run.

Homework Problems:

1. Given a specific value w and a set of non-negative integers, determine whether there is a subset of the given set with a total value equals to w.

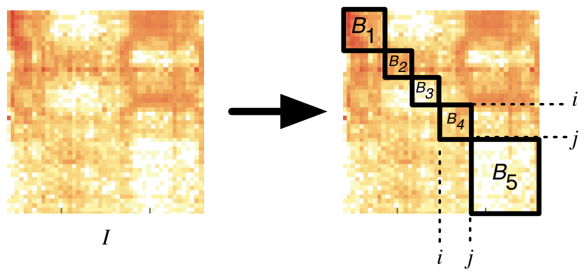
2. You are given a set of classes, each with a start time and end time, and a positive number of units. For convenience, you can consider time to be a positive integer from 0 to k. Return the maximum number of units you can take without having any overlapping classes.

3. In some languages, such as Chinese, words are sometimes not separated by spaces. For another example, in German, numbers are sometimes written together "Dreihundertfünfzigfünftousand" and compound words are often created: "Glückszahl" means "lucky (Glück) number (zahl)." We would like to decompose such strings into the component words that were used to form them. Assume you have a function $word(s)$ that takes a string s and returns a score indicating how likely it is that s is a indivisible word. For example, in German, "zahl" would receive a high score, but "kszahl" would not.

   Design a dynamic programming algorithm to break a given string $a = a_1a_2 \dots a_n$ into words $w_1, \dots, w_k$ to maximize $\sum_i word(w_i)$. Note that you are *not* given k as input.

4. You are given a rooted tree T of n nodes, where every node i is associated with a weight w_i. Note that w_i can be negative. Your task is to select a subset of nodes to maximize the their total weight. However, if you select node u, then you can't select any of u's descendants. Design an $O(n)$-time dynamic programming algorithm to find the maximum total weight.

5. You are given an image I that consists of $n \times n$ pixels of various colors. A *block diagonal partition* $\{B_1, \dots, B_k\}$ of the image is a set of non-overlapping, *square* boxes that cover the diagonal of the matrix. See example below, where the boxes in the right image give a block diagonal partition.



You are also given a function called "$f(B)$" that returns a number between 0 and 1 to indicate how uniform the color within the box B is: it returns 1 if there is only one color in the box and 0 if there are lots of different colors.

Give a dynamic programming solution to find the optimal block diagonal partition $P = \{B_1, \ldots, B_k\}$ of an image I that maximizes the following quantity:

$$-\alpha k + \sum_{i=1}^{k} f(B_i)$$

where α is a given constant and k is the number of blocks in the partition P. Note you are given α, but you are *not* given k.