Homework 3

Due Date: September 23, 2019 at 11:59PM on Gradescope.
Instructor: Jeremiah Blocki

Homework Guideline Reminders

- Assignments must be typed. Submit one pdf file to Gradescope by 11:59PM, or else late penalties will apply. The pdf file can include hand-drawn images of figures.
- Each question needs to start with the resources and collaborator (RC) statement. You will not be penalized for using resources or having collaborators if your answers are expressed in your own words. If you consulted no resources outside of course material or had no collaborators, you must state so. A question without a complete RC statement will not be graded.

Question 1: Divide and Conquer (33 points)

An array $A[1..n]$ has the "**AP-free**" property if it does not contain any arithmetic progression, in other words, for every pair (p, q) of indices with $1 \leq p < q \leq n$ there is no index r with $p < r < q$ such that $A[p] + A[q] = 2 \cdot A[r]$.

(a) Given a fixed positive integer n, devise an efficient divide and conquer algorithm that returns a permutation of $(1, 2, \ldots, n)$ that is AP-free. For example, (2,1,4,3) is an AP-free permutation of (1,2,3,4) for $n = 4$. You may assume that $n = 2^k$ is a power of 2.

(b) Prove the correctness of your algorithm and analyze its running time.

Hint 1: It may be helpful to remember that adding an even number to an odd number always yields an odd number.

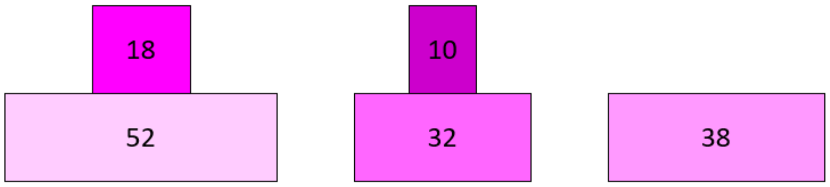
Hint 2: Note that the AP-free property is linear, that is, an array $A[1..n]$ is AP-free if and only if for every integers u, v the array $B[1..n]$ obtained from setting $B[i] = u \cdot A[i] + v$ for $1 \leq i \leq n$ is also AP-free e.g., (2,1,4,3) is AP-free if and only if (4,2,8,6) is AP-free. You may use this observation without proof.

Question 2: Greedy Algorithm (33 points)

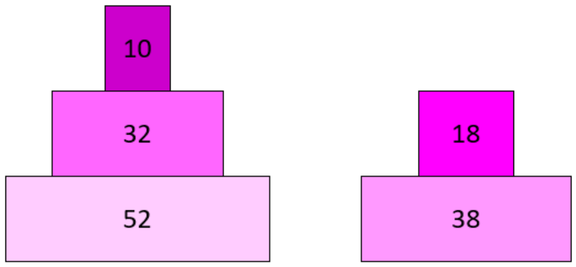
Jane is now a baker at Blocki's Blasted Bakery (BBB), and she bakes cakes of varying sizes for her customers. She has to pack the cakes in boxes and send them out for delivery at the end of the day. She bakes the cakes in order of when the customer places their order, and she must pack the cake before starting the next cake. She can use as many boxes as she needs, but the delivery cost is dependent on the number of boxes she uses. She can stack several cakes in the same box, but she cannot stack a larger cake onto a smaller cake or it will collapse. She seeks out help from the students in CS381 as she wants to minimize the delivery cost.

More precisely the input to our algorithm is an ordered list of cakes' sizes e.g., [52, 32, 38, 10, 18]. The output is a list of valid stacks of cakes. A valid stack should be non-collapsing e.g., the stack $(32, 38)$ would collapse since the larger cake (size 38) is on top of the smaller one (size 32). Similarly, a valid stack should respect the original ordering of cakes e.g., the stack $(38, 32)$ is invalid because the cake with size 38 would have been baked/packed after the cake with size 32. The goal is to minimize the number of stacks of cakes. For this problem we will assume that there is no limit on the height of a stack.

Since cakes must be stacked in order, we could, for example, stack cake 1 on box 1, cake 2 on box 2, cake 3 on box 3, cake 4 on box 2, and cake 5 on box 1 as shown below.



However, the optimal solution is 2 boxes as follows.



(a) Devise a greedy algorithm which returns a packing of the cakes that minimizes her delivery cost. Analyze the time and space complexity of your algorithm.

(b) Prove the correctness of your algorithm i.e. prove that the number of boxes used by your greedy algorithm is no more than number of boxes used by any optimal algorithm.

Question 3: Proof/Counterexample (34 points)

After an economic analysis, BBB has decided that fixed size (say size 50) cakes are the most profitable and has decided to only bake cakes of size 50. BBB also expanded its storage room so Jane is now able to bake all of the cakes before packing the boxes. Once again Jane has to deliver a batch of cakes, however, this time, her delivery boxes come with specific capacities: 1, 5, 10, and 50, i.e a box of capacity 1 would fit 1 cake, a box of capacity 5 would fit 5 cakes, and so on. A box cannot fit more than its capacity, and packing less than the capacity would cause the cakes to be ruined during delivery, so each box must contain exactly the number of cakes of its capacity. Like before, she needs to minimise her delivery cost which is directly proportional to the number of boxes she uses (all boxes have the same cost regardless of capacity). Given a number of cakes, n, she packs the cakes using the following greedy strategy: She repeatedly chooses the biggest box out of the 4 choices that she can completely fill and continues until she has packed all her cakes.

(a) Prove or disprove that Jane's strategy is optimal.

(b) There was a manufacturing mixup with the boxes, and instead, Jane received boxes with different capacities: 1, 5, 7, and 50. Has the optimality changed? Prove or disprove that Jane's strategy is optimal in this situation.