CS 218, Fall 2018
Posted: November 14th, 2018

Name:

Student ID #:

• You are expected to work on this assignment on your own
• Use pseudocode, Python-like or English to describe your algorithms. Absolutely no C++/C/Java
• When designing an algorithm, you are allowed to use any algorithm or data structure we explained in class, without giving its details, unless the question specifically requires that you give such details
• Always remember to analyze the time complexity of your algorithms
• Homework has to be submitted electronically on Gradescope by the deadline. No late assignments will be accepted
Problem 1. (10 points)

We are given a list of $n$ items with sizes $s_1, s_2, \ldots, s_n$. A *sequential bin packing* of these items is an assignment of items to bins, such that in each bins the items are consecutive. (That is, each bin has items $s_i, s_{i+1}, \ldots, s_j$ for some indices $i \leq j$.) Bins have unbounded capacities. The *load* of a bin is the sum of the size of the elements in it.

Give an algorithm that determines a sequential packing of $n$ items into $k$ bins (some of which might end up empty) for which the maximum load of a bin is minimized. Analyze the time complexity of your algorithm.

Answer:
Problem 2. (10 points)

We are given a weighted complete bipartite graph $G = (X, Y, X \times Y)$, where $X = \{x_1, \ldots, x_n\}$, $Y = \{y_1, \ldots, y_m\}$, with $n < m$. By $w_{i,j}$ we denote the weight of edge $(x_i, y_j)$.

A cross-free matching $M$ is a matching between $X$ and $Y$ in which no two edges “cross”. More specifically, if $i < j$ and $(x_i, y_k), (x_j, y_l) \in M$ then $k < l$. The weight of a matching $M$ is the sum of the weights of the edges in $M$.

Give a dynamic programming algorithm that computes a perfect cross-free matching in $G$ of minimum weight (perfect here means that each node in $X$ is matched – some of the nodes in $Y$ will remain unmatched). Analyze the space and time complexity of your solution.

Answer:
Problem 3. (10 points)

You are given \( n \) biased coins described by their probabilities \( p_1, \ldots, p_n \in [0, 1] \), where \( p_i \) is the probability that the \( i \)-th coin comes up heads. Given an integer \( k \), you want to determine the probability of obtaining exactly \( k \) heads when these coins are tossed independently at random.

For instance if \( n = 4 \), the probability of obtaining \( k = 2 \) heads is
\[
p_1p_2(1 - p_3)(1 - p_4) + p_1(1 - p_2)p_3(1 - p_4) + p_1(1 - p_2)(1 - p_3)p_4 + (1 - p_1)p_2(1 - p_3)p_4 + (1 - p_1)(1 - p_2)p_3p_4.
\]

- Give an \( O(n^2) \) dynamic programming algorithm for this task. Assume you can multiply and add two numbers in \([0, 1]\) in constant time.

- Design a degree \( n \) polynomial \( g(x) \) so that the coefficient of \( x^k \) in the polynomial is exactly the probability of obtaining exactly \( k \) heads. Show how one can compute \( g(x) \) using divide-and-conquer in \( O(n \log^2 n) \) time.

Answer: