COP4533
Algorithm Abstraction & Design
Programming Project (Fall 2024)

1 Problem Definition

Consider the problem of curating an art exhibition featuring n paintings that must be displayed in a specific sequence determined by the theme of the show; thus, their order cannot be changed. Each painting s_i, where $1 \leq i \leq n$, has a base width w_i and a height h_i. The gallery space is divided into display platforms, each with a fixed maximum width W. The total width of the paintings placed on a single platform cannot exceed W. We have the flexibility to adjust the height of each platform to match the tallest painting placed upon it. The cost of a particular arrangement is the sum of the heights of the tallest paintings on each platform. Your goal is to determine an assignment of the paintings to platforms that minimizes the total cost.

Example 1: $n = 7$, $W = 10$,
$h_i = [21, 19, 17, 16, 11, 5, 1]$
$w_i = [7, 1, 2, 3, 5, 8, 1]$

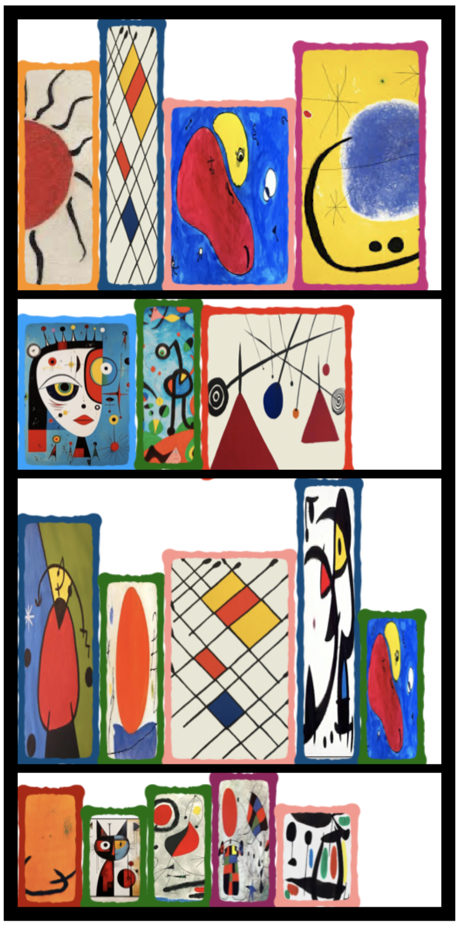
Solution 1: $Platform_1 = [s_1 ... s_3]$;
$Platform_2 = [s_4 ... s_5]$;
$Platform_3 = [s_6 ... s_7]$;
$cost = 21 + 16 + 5 = 42$

Example 2: $n = 4$, $W = 10$,
$h_i = [8,\ 10,\ 9,\ 7]$
$w_i = [8,\ 1,\ 2,\ 2]$

Solution 2: $Platform_1 = [s_1]$;
$Platform_2 = [s_2 \dots s_4]$;
$cost = 8 + 10 = 18$

Example 3: $n = 7$, $W = 10$,
$h_i = [12,\ 10,\ 9,\ 7,\ 8,\ 10,\ 11]$
$w_i = [3,\ 2,\ 3,\ 4,\ 3,\ 2,\ 3]$

Solution 3: $Platform_1 = [s_1 \dots s_3]$;
$Platform_2 = [s_4]$;
$Platform_3 = [s_5 \dots s_7]$;
$cost = 12 + 7 + 11 = 30$



Below is the formal problem definition, starting with the general version, followed by two special cases. The special cases include input restrictions, which may allow for more efficient solutions.

ProblemG Given the heights $h_1, \dots, h_n$ and the base widths $w_1, \dots, w_n$ of n paintings, along with the width W of the display platform, find an arrangement of the paintings on platforms that minimizes the total height.

PROBLEMS1 Given the heights $h_1, \ldots, h_n$, where $h_i \geq h_j \; \forall i < j$, and the base widths $w_1, \ldots, w_n$ of n paintings, along with the width W of the display platform, find an arrangement of the paintings on platforms that minimizes the total height.
(Note: The heights of the paintings form a monotonically non-increasing sequence, as in EXAMPLE 1.*)*

PROBLEMS2 Given the heights $h_1, \ldots, h_n$, where $\exists k$ such that $\forall i < j \leq k,\; h_i \geq h_j$ and $\forall k \leq i < j,\; h_i \leq h_j$, and the base widths $w_1, \ldots, w_n$ of n paintings, along with the width W of the display platform, find an arrangement of the paintings on platforms that minimizes the total height.
(Note: The heights of the paintings follow a unimodal function with a single local minimum, as in EXAMPLE 3.*)*

2 MILESTONE 1 - Greedy Algorithms

For the first part of the project, you will design, analyze and implement greedy algorithms to solve the special cases of the problem. You will conduct an experimental study on the performance of your algorithms. You are also required to construct examples showing how an algorithm developed for a special case of the problem might not work for the general version of the problem.

2.1 Design, Analysis, and Implementation Tasks

ALGORITHM1 Design a $\Theta(n)$ time greedy algorithm for solving PROBLEMS1.

ANALYSIS1 Prove the correctness of your greedy ALGORITHM1 for solving PROBLEMS1.

QUESTION1 Give an input example showing that Algorithm1 does not always solve PROBLEMG.

QUESTION2 Give an input example showing that Algorithm1 does not always solve PROBLEMS2.

PROGRAM1 Give an implementation of your greedy ALGORITHM1.

ALGORITHM2 Design a $\Theta(n)$ time greedy algorithm for solving PROBLEMS2.

ANALYSIS2 Prove the correctness of your greedy ALGORITHM2 for solving PROBLEMS2.

PROGRAM2 Give an implementation of your greedy ALGORITHM2.

2.2 Experimental Comparative Study

Test your implementations extensively for correctness and performance. For this purpose, you should create randomly generated input files of various sizes. The exact size of the experimental data sets that your program can handle depends on the quality of your implementation. For instance, you might want to choose $n = 1000, 2000, 3000, 4000, 5000$ to create at least five data sets for each experiment. Then, you should conduct and present a performance comparison of the following: For each comparison, generate a two dimensional plot of running time (y-axis) against input size (x-axis). These should be included in your report along with additional comments/observations.

PLOT1 Plot the running time of PROGRAM1 with respect to varying input size (as x-axis).

PLOT2 Plot the running time of PROGRAM2 with respect to varying input size (as x-axis).

2.3 Report

Prepare a report that describes design and analysis of your algorithms, presents the results of experimental study, and summarizes your learning experience. Your report should include but not limited to the following sections.

- **Team Members.** You may choose to work alone or in a team of three students on this assignment. If you choose to work in a team, clearly state the name and the contribution of each team member.
- **Algorithm Design and Analysis.** Give a clear description of your algorithm design and as well as its analysis. You can also include the pseudo code of your algorithms. Make sure to provide the answers to QUESTION1 and QUESTION2 as well.
- **Experimental Study.** Present the results of your experimental study. Include all your plots along with any explanation and comments you wish to provide.
- **Conclusion.** Summarize your learning experience on the first component of the project assignment. For each programming task, comment on the ease of implementation and other potential technical challenges.

2.4 Deliverables

The following contents are required for MILESTONE1 submission:

- **Implementation**: Submit the implementation code on Gradescope. Refer to Section 5 for more details. Make sure to include detailed comments next to each non-trivial block of code.
- **Report**: The report, in PDF format, must be submitted on Canvas.

3 MILESTONE 2 - Dynamic Programming Algorithms

For the second part of the project, you will design, analyze and implement dynamic programming algorithms to solve the general cases of the problem. You will conduct an experimental study on the performance of your algorithms.

3.1 Design, Analysis, and Implementation Tasks

ALGORITHM3 Design a $\Theta(n2^{n-1})$ time naive algorithm for solving PROBLEMG.

ANALYSIS3 Give an analysis (correctness & running time) of ALGORITHM3 for solving PROBLEMG.

ALGORITHM4 Design either a $\Theta(n^3)$ time dynamic programming algorithm or a $\Theta(n^2|W|)$ time dynamic programming algorithm for solving PROBLEMG.

ANALYSIS4 Give an analysis (correctness & running time) of ALGORITHM4 for solving PROBLEMG.

ALGORITHM5 Design a $\Theta(n^2)$ time dynamic programming algorithm for solving PROBLEMG.

ANALYSIS5 Give an analysis (correctness & running time) of ALGORITHM5 for solving PROBLEMG.

PROGRAM3 Give an implementation of ALGORITHM3.

PROGRAM4 Give an implementation of ALGORITHM4.

PROGRAM5A Give a top-down recursive implementation of ALGORITHM5 using memoization.

PROGRAM5B Give an iterative bottom-up implementation of ALGORITHM5.

3.2 Experimental Comparative Study

Test your implementations extensively for correctness and performance. For this purpose, you should create randomly generated input files of various sizes. The exact size of the experimental data sets that your program can handle depends on the quality of your implementation. For instance, you might want to choose $n = 1000, 2000, 3000, 4000, 5000$ to create at least five data sets for each experiment. Then, you should conduct and present a performance comparison of the following: For each comparison, generate a two dimensional plot of running time (y-axis) against input size (x-axis). These should be included in your report along with additional comments/observations.

PLOT3 Plot the running time of PROGRAM3 with respect to varying input size (as x-axis).

PLOT4 Plot the running time of PROGRAM4 with respect to varying input size (as x-axis).

PLOT5 Plot the running time of PROGRAM5A with respect to varying input size (as x-axis).

PLOT6 Plot the running time of PROGRAM5B with respect to varying input size (as x-axis).

PLOT7 Overlay PLOTS 3,4,5,6 and contrasting the performance of PROGRAMS 3,4,5A,5B.

PLOT8 Overlay PLOTS 5,6 and contrasting the performance of PROGRAMS 5A, 5B.

In addition to testing the running time performance of the algorithms described above, you are asked to conduct experiments to test the quality of the output of PROGRAM1 which implements greedy ALGORITHM1, when it is run on the general case of the problem PROBLEMG. Note that such solution is not guaranteed to be optimal. Your task is to determine how different it is from the optimal. Make sure to conduct enough experiments with many random data sets. For x-axis, simply use n as the varying input size. On y-axis you can simply plot $(h_g - h_o)/h_o$, where h_o is the optimal height determined by any of PROGRAMS 3,4,5A,5B, and h_g is the height determined by PROGRAM1.

PLOT9 Plot the output quality comparison $(h_g - h_o)/h_o$ of PROGRAM1 and any of PROGRAMS 3,4,5A,5B.

3.3 Report

Prepare a report that describes design and analysis of your algorithms, presents the results of experimental study, and summarizes your learning experience. Your report should include but not limited to the following sections.

- **Team Members.** You may choose to work alone or in a team of three students on this assignment. If you choose to work in a team, clearly state the name and the contribution of each team member.
- **Algorithm Design and Analysis.** Give a clear description of your algorithm design and as well as its analysis. Make sure to include the recursive formulation expressing optimal substructure for the dynamic programming algorithms. You can also include the pseudo code of your algorithms.

- **Experimental Study.** Present the results of your experimental study. Include all your plots along with any explanation and comments you wish to provide.
- **Conclusion.** Summarize your learning experience on the first component of the project assignment. For each programming task, comment on the ease of implementation and other potential technical challenges.

3.4 Deliverables

The following contents are required for Milestone2 submission:

- **Implementation**: Submit the implementation code on Gradescope. Refer to Section 5 for more details. Make sure to include detailed comments next to each non-trivial block of code.
- **Report**: The report, in PDF format, must be submitted on Canvas.

4 MILESTONE 3 - Video Presentation

Prepare a video of length between 5 to 7 minutes, that presents all your results (design, analysis, experiments, and learning experience). Following are the expected components. Summarize your results on algorithm design and analysis. Note that you have already detailed these in your report. Given the time limit for the presentation, it is best to focus on the challenging components. For instance of all the design and analysis tasks, I recommend you spend more time presenting the following: Algorithm2, Analysis2, Algorithm4, Analysis4, Algorithm5, Analysis5. Also, contrast Algorithm4 and Algorithm5, given that both are dynamic programming algorithms solving the same problem but they have different time complexity. [recommended time: upto 2-4 minutes] Comment on your implementations (e.g., whether you used any other resources/libraries) and make a demo of your implementations clearly showing that they work. [recommended time: 2-3 minutes] Summarize your experimental study results, using your plots. [recommended time: 1 minute] In conclusion briefly comment on your learning experience. For group projects, all members must take part in the video presentation.

5 Language/Input/Output Specifications

You may use C++, Java, or Python. Encapsulation code for each language will be provided (they differ only in filenames and class names). Add your implementation to the provided encapsulation code in the language of your choice. Do **not** modify the filenames or class names in the given code. You must upload a file for each program on Gradescope (1 file per program). Do **not** include any extra files. You will receive instant feedback for a few example test cases to verify that your code follows the correct specifications.

The autograder on Gradescope runs on an Ubuntu 22.04 image. C++ code will be compiled using g++ 11.4.0 with the '-std=gnu++17' compile flag. Java code will be compiled and evaluated using OpenJDK 17. Python code will be evaluated using Python 3.10.

For convenience, you can assume that $1 \leq n, W < 10^5$, and $\forall i \ 1 \leq h[i], w[i] < 10^5$. Test cases will be generated to accommodate the required time complexity.

Input. Your program will read input from standard input (`stdin`) in the following order:

- Line 1 consists of two integers n and W separated by a single space.
- Line 2 consists of n integers (height of n paintings) separated by a single space.
- Line 3 consists of n integers (base width of n paintings) separated by a single space.

Output. Your program should print to standard output (`stdout`) in the following order:

- Line 1 contains the number of platforms used (denoted as m)
- Line 2 contains the optimal total height.
- The next m lines should consist of the number of paintings on each platform.

6 Grading Policy

Milestones 1, 2, and 3 will weigh **30%**, **50%**, and **20%** of your total project grade, respectively. On each Milestone you will be assigned 20% bonus points.

For Milestones 1 and 2, grades will be based on the correctness & efficiency of algorithms and the quality of your experimental study and report:

- **Program 60%.** Correct/efficient design and implementation/execution. Also make sure to include comments with your code for clarity.
- **Report 40%.** Quality (clarity, details) of the write up on your design, analysis, programming experience, and experimental study.