Programming Assignment 1
Advanced Algorithms

Deadline: 6 October 2019 23:59

1 Introduction

The first and second programming assignments are closely related. In both, you will work on the auctions with budget constraints problem. For the first programming assignment, we expect you to implement an approximation algorithm for this problem, as well as a short report.

This is a pair assignment (group of size two).

2 Approximate Auction Revenue Maximization

Problem Statement. The 'auctions with budget constraints' problem, as well as applicable algorithms, are described extensively in Andelman and Mansour [1]. In this auction, k different items are sold to n different bidders. Each bidder i has a bid b_{ij} for item j. Additionally, each bidder i has a budget limit d_i. Due to this budget limit, given an allocation of the items among the bidders, the seller's revenue for bidder i becomes $\min\left(d_i, \sum_{j=1}^{k} x_{ij} b_{ij}\right)$, where

$$x_{ij} = \begin{cases} 1 \text{ if item } j \text{ is allocated to bidder } i. \\ 0 \text{ otherwise.} \end{cases}$$

In other words, if a bidder is allocated items whose combined cost exceeds his budget limit, he pays his budget limit. The objective is to find an allocation of the items that maximizes the seller's revenue.

As an example, suppose we have 3 items that are auctioned to 2 bidders. Their budget limits are 10 and 20, respectively. Their bids for the three items are as follows:

$$\begin{aligned} b_{11} &= 2 \qquad & b_{21} &= 5 \\ b_{12} &= 5 \qquad & b_{22} &= 3 \\ b_{13} &= 8 \qquad & b_{23} &= 10 \end{aligned}$$

Given an allocation of item 1 to the second bidder and items 2 and 3 to the first bidder, the seller's revenue becomes $\sum_{i=1}^{2} \min\left(d_i, \sum_{j=1}^{3} x_{ij} b_{ij}\right) = 10 + 5 = 15$.

Approximation Algorithm. The problem of maximizing the seller's revenue exactly is NP-hard. Instead, you are asked to design and implement an FPTAS approximation algorithm for dealing with this problem. The algorithm is described in Section 4.1 of [1]. Of course you are also allowed to use any other source. Just make sure that you include proper references in your report.

3 Implementation Requirements

Your implementations of the algorithm should meet the following constrains:

- The algorithm must be written by your group. Discussing or sharing thoughts about the assignment with other groups is allowed, sharing solutions or code is not.
- The pseudocode should fit on one A4 paper.
- Your code should meet the requirements for neat programming from earlier courses.
- The executable should accept two arguments: the path to an instance file and ϵ (float).
- The executable should print a single integer: the seller's revenue.

3.1 Input Format & Sample Instances

We have included some sample instances of the problem that are comparable in complexity to the ones we will use for the automated testing during the grading procedure. Of course, you are free to create your own test cases.

Note that since we use automatic testing, your program should handle the input format of the attached files:

The first line of the file contains a single integer: n the number of bidders. The second line contains another integer: k, the number of items. The third line contains n integers: the budget limits of each bidder (d_1, d_2, ..., d_n), separated by a single ' ' (space) character. The following n lines contain k integers: the values of the items for each bidder (b_{ij}).

For the example above, the input file would look like the following:

```
2
3
10 20
2 5 8
5 3 10
```

You may assume that all input files are valid according to this specification.

3.2 Environment

The algorithm must be implemented in Java. We provide a skeleton with a sample brute force algorithm. You can use it as a template and replace the brute force algorithm with your implementation.

We use an automated grading pipeline to evaluate the efficiency of your algorithm. Therefore, you have to submit a compressed folder (*zip*) containing your implementation in CPM, meeting the following requirements:

- You should keep the file name of the Main class as it was provided in the template
- The uploaded *zip* file should have no directory structure inside it. Of course, you can split your implementation into separate files, but do not add any packages or dependencies.

We compile and run your implementation with the following commands:

```
find -name "*.java" > sources.txt
javac @sources.txt
java Main sample_instances.txt 0.1
```

4 Report Requirements

We expect you to write a brief report containing the following:

1. A brief description (including pseudo-code and time-analysis) of the FPTAS. For the pseudocode, we recommend to use one of LaTeX' pseudocode packages like `algorithmicx`.
2. Your expectations before running the algorithm on various test problems.
3. A comparison of run time of your algorithm for the various instance parameters n, k, d_{max}, as well as ε.
4. A discussion of some important results of this comparison.

Your report on the algorithm should meet the following constraints:

1. The report must be a single PDF file.
2. It must contain your names, student numbers, the Brightspace group number and the assignment number.
3. It should be at most two sides of an A4 (excluding an optional front page and space used by figures or pseudocode).
4. It should be written in clear, readable and correct **English**.
5. All the presented figures must be high quality (optimally vector graphical).

5 Assessment and Feedback

The runtime of your algorithm will be measured for various instances similar to the ones we have provided, with $\epsilon = 5$. As a guideline, you should be able to solve all provided instances with $\varepsilon = 5$ within 3 minutes on an Intel(R) Core(TM) i7 CPU.

We will check both your report as well as your code on correctness, clarity, quality and completeness with respect to the requirements. A breakdown of the grading is given below.

- **(1p)** Pseudocode of your FPTAS algorithm.
- **(1p)** Theoretical runtime analysis of your pseudocode.
- **(4p)** Implementation of the algorithm. Fewer points in case of mismatches between the pseudocode and implementation, or inefficient choices.
- **(2.5p)** Experimental analysis of the algorithm's runtime. How do the various problem instance parameters (n, k, d_{max} and ε) influence the runtime?
- **(1.5p)** Expectation of the above experiments, and a discussion of the actual results.

Good luck!

References

[1] Nir Andelman and Yishay Mansour. Auctions with budget constraints. In *Scandinavian Workshop on Algorithm Theory*, pages 26–38. Springer, 2004.