CS2500 Algorithms
First Project

Due dates: First Report March 7th, Second Report March 16th

Project Demo: Schedule with TA after 2nd report submission

Project goal: Investigate on performance of Brute Force, Dynamic Programming, and Greedy Algorithms to manage applications on your smart device

General information

Maximum group size 3.

Any programming language can be used for the implementation.

The project reports should be submitted through the blackboard by the due date.

The project should be presented to the TA within 1 week from the deadline of the second report.

Problem description: Application Management System

There are a variety of apps we use on our smartphones. Though we only see one app running while we are using the application, there are other apps running in the background. These applications are active in the main memory. This allows for the apps to be reopened faster, as they're waiting in memory for you to return.

However, smart devices are typically resource-constrained. Hence, it is likely to run out of memory when you keep every app in memory. To run a new app, the operating system needs to choose and remove some apps from the memory. This process is called "*deactivation*".

When the system runs out of memory, it is not a good way to remove apps by randomly choosing them. To re-run the apps, the data should be re-transferred from the storage back into memory, if the apps were removed from the memory. Hence, it would take much longer time to re-open the apps.

In this project, you need to develop three algorithms to solve "app deactivation" problems and compare their performances.

Problem formulation

N: the number of activated apps

$\mathbf{A_i}$: Each application (where i=1,…,n)

$\mathbf{M_i}$: the memory space for the app $\mathbf{A_i}$.

$\mathbf{C_i}$: additional cost when you re-run the removed app $\mathbf{A_i}$

Our goal is to deactivate some apps in $\{\mathbf{A_1},\ldots,\mathbf{A_n}\}$ to secure $\boldsymbol{M}$ memory spaces while minimizing the summation of the re-run cost.

As you may imagine, you cannot choose fraction of any app $\mathbf{A_i}$.

Algorithm design

You need to develop three algorithms.

1. Brute force approach: Search all possible subsets and picks the one with minimum costs.
2. Dynamic programming: Save the solutions and look up the table
3. Greedy algorithm: Heuristic solutions

Evaluation

The approaches should be tested on various inputs, with different size and heterogeneity. Randomly choose the $\boldsymbol{M}$, $\boldsymbol{M_i}$ and $\boldsymbol{C_i}$ for each $\boldsymbol{A_i}$ within the appropriate range.

Evaluations should be carried out in terms of two measures:

(1) Execution time.
(2) The costs for the solution.
(3) Accuracy

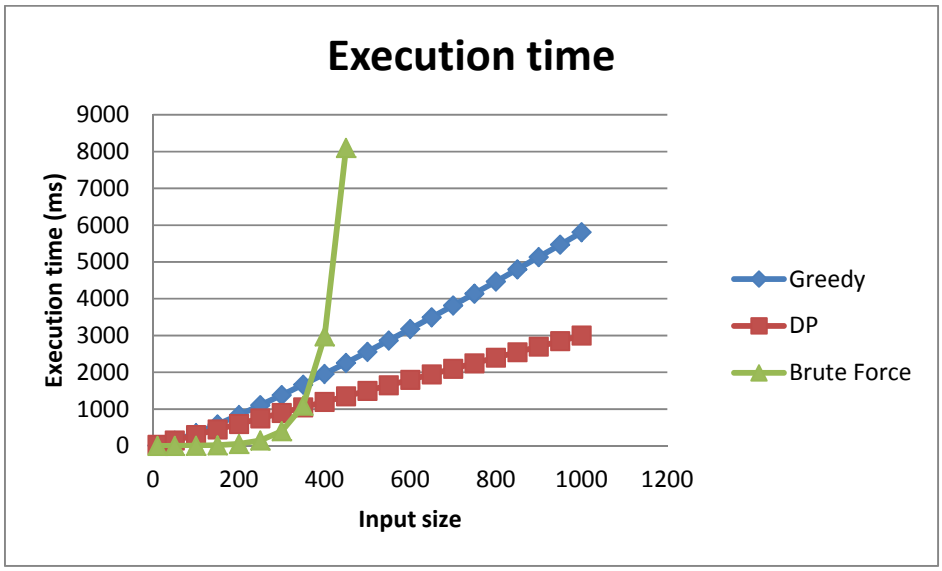
At least 2 graphics including execution time and costs for the solution should be generated to compare three approaches.

In each graphic, you will have three trend lines: one for brute force approach, one for dynamic programming approach, and one for greedy approach.

Each point in the graphic should be the average of the result for at least 10 inputs. For example, in order to measure the execution time for an input of size n = 50, it is necessary to generate at least 10 different inputs whose sizes are 50. Then the execution time should be evaluated on each input, and the corresponding point in the graphic will be the average of these values.

The X-axis will be the input size. The Y-axis will be average execution time or average costs for the solution.

The example graphic can be as follows.



Note that this example graphic is drawn randomly. The inputs size can be changed to test meaningful values in the project (i.e., values for which the algorithms show a observable differences).

Compare the accuracy of greedy algorithm against the brute force approach and the dynamic programming approach. Here, the accuracy can be defined in many different ways. For example, you may compare the cost of greedy algorithm with dynamic programming approach. You may compare the selected set of apps.

Optional: If you come up with other interesting results, you can add them to the report. If it is significant, you will get extra credits.

Reports

Two reports should be prepared. The first report does not require implementation and should include at least the following sections:

- Abstract: Summary of the following sections
- Introduction and motivation: Overview of application management system

- Proposed solutions: Pseudo-code, description and complexity analysis of the proposed algorithms
- Plan of experiments: Description of the methodology that you plan to use in the experiments, significance and expected results.
- Each team member's role in this project

The second report requires implementation and describes the implementation and results. It should include at least the following sections:

- Abstract: Summary of the following sections and results
- Implementation: Description of the methodology used for implementing the proposed solutions (e.g. relevant classes and data structures, structure of the program, etc.).
- Experiments: Description of the experiments performed and obtained results. Results should be represented in a graphical form as well as discussed in the writing.
- Each team member's role in this project
- Conclusions: Summary of the work and final considerations.

Source Code

You need to submit your source code along with 2nd report. **DO NOT** include your source code in the report.

Source code should be in the separate directory. Create .zip or .tar file including your 2nd report and source code directory.