April 29th, 2022

Dokuz Eylül University Department of Computer Engineering

CME 2204 Assignment 2

Dynamic Programming & Greedy Approach

Assigned: April 29th, 2022 TA: Ibrahim Berber
Due: May 22nd, 2022 (Sunday 23:55)

Neat and concise code solutions are required to receive full credit for your solutions. If you cannot solve a particular step, state this clearly in your code and report. The submissions will be checked for the code similarity and maximum action will be taken for any plagiarism. Therefore, any resources utilized must absolutely be referenced.

Introduction

In this assignment you will implement a program which simulates a value game using customized chess pieces. In our case, there will be many available pieces for each type of piece, each having a historical name, along with an arbitrary attack point. Each piece type will have a level. You will select your pieces depending on maximum allowed level (see Table 1). For instance, if we set the maximum allowed level to 3, you could only have one pawn, one rook and one archer. If you fall short of gold, you may also prefer not to buy a piece from the allowed level (i.e., leaving that level empty) to have a higher valued piece. For example, instead of buying one pawn, one rook and one archer, you could skip rook (in level 2) and get a better archer, if that makes you more advantageous in total attack point. In other words, you don't need to fill the piece for each allowed level.

You may assume that there will be only one type of piece **at most** from each level, e.g., no two pawns are allowed. In the beginning of the game, you will start with a specified amount of gold, which you will use it to raise your army. Your objective is to assemble an army with as high attack point as possible.

Max level	Pawn	Rook	Archer	Knight	Bishop	War ship	Siege	Queen	King
1	✓	✗	✗	✗	✗	✗	✗	✗	✗
2	✓	✓	✗	✗	✗	✗	✗	✗	✗
3	✓	✓	✓	✗	✗	✗	✗	✗	✗
…	...	...	...	...	...	...	...	...	...
9	✓	✓	✓	✓	✓	✓	✓	✓	✓

Table 1: *Name of the pieces and their levels (piece types).*

When the game starts, it must take the necessary arguments given in Code 1. from the user.

```
int GOLD_AMOUNT;
int MAX_LEVEL_ALLOWED;
int NUMBER_OF_AVAILABLE_PIECES_PER_LEVEL;
```

Code 1: *Required user inputs*

Now, let us consider individual types. For example, you may have the option of selecting following knights (or not selecting any of them, leaving level 4 blank) when *number of allowed pieces per level* is set to 3 and *max level allowed* is greater than or equal to 4. You will select at most one of them.

Knight #1:	Knight #2:	Knight #3:
El Cid	Richard the Lionheart	Attila
120 attack points	150 attack points	175 attack points
80 gold	105 gold	120 gold

Figure 1: *An example where there are three options for selecting the knight.*

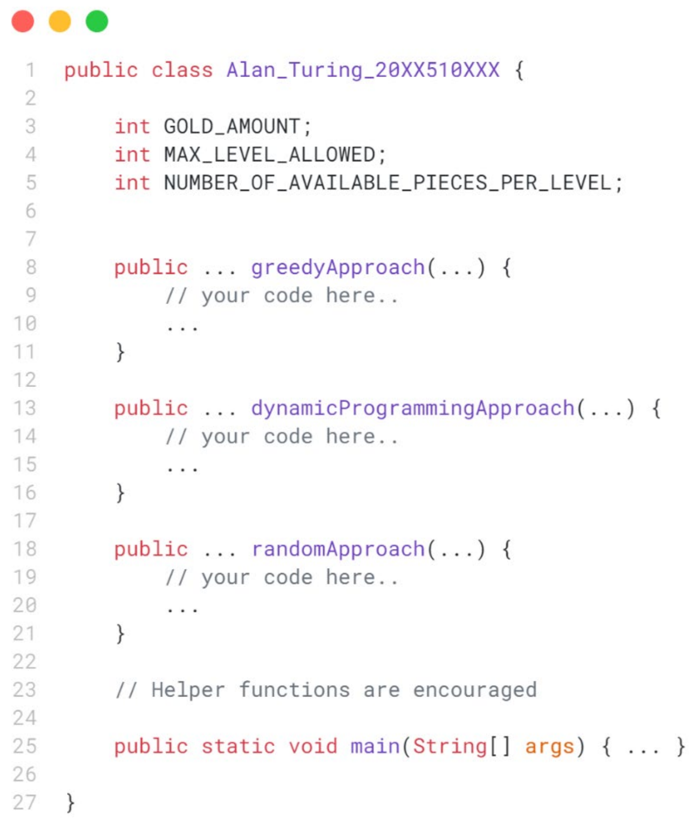
Again, keep in mind that there might be a piece from lower levels that has higher attack points. You should pick pieces in a way such that maximize the **total attack points of entire army**. Make your decisions wisely.

Problem Statement

Consider the following scenario. Let assume that you (as a user) play against an opponent (computer). The computer performs its selection based on *greedy approach (GA)* or *random approach (RA)*. On the other hand, you select your hero based on *dynamic programming approach (DP)*. There will be two trials in the game. While the Computer uses the greedy approach in the first trial and the random approach in the second trial, User will adapt dynamic programming in both trials. For these three approaches, you are instructed to measure their execution times. More specifically, whenever GA, RA or DP is called, their individual runtimes should be measured.

Submission

You will submit your solutions with a zip file <*Name_Surname_StudentID.zip*>, containing your code <*Name_Surname_StudentID.java*> and your report <*Name_Surname_StudentID.pdf*>. Please use a *single* Java file, though you are encouraged to use multiple classes. As an example, refer to Code 2 below.



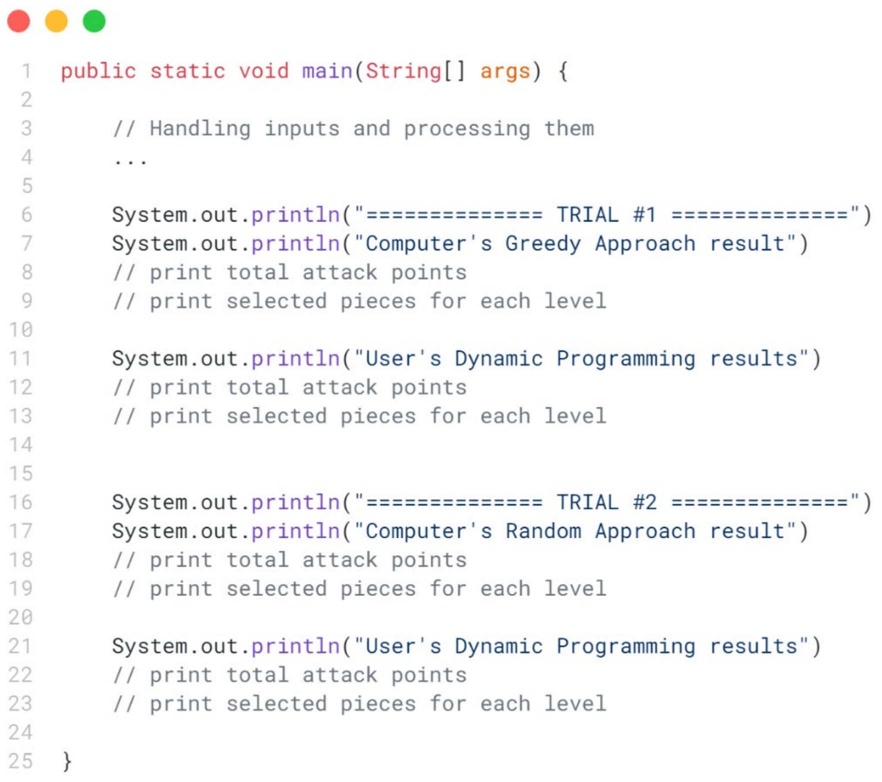
```java
public class Alan_Turing_20XX510XXX {

    int GOLD_AMOUNT;
    int MAX_LEVEL_ALLOWED;
    int NUMBER_OF_AVAILABLE_PIECES_PER_LEVEL;


    public ... greedyApproach(...) {
        // your code here..
        ...
    }

    public ... dynamicProgrammingApproach(...) {
        // your code here..
        ...
    }

    public ... randomApproach(...) {
        // your code here..
        ...
    }

    // Helper functions are encouraged

    public static void main(String[] args) { ... }

}
```

Code 2: *The code file Alan_Turing_20XX510XXX.java to be uploaded*

You can use auxiliary text files to save your results if convenient, but make sure you include them in your zip file. A typical main method may look as shown in Code 3 on the following page.

Please ensure that your codes are working. Again, if they are not, then indicate them both in report and code as a comment, such as *not working* or *partially working*. Clear explanation of critical parts of your codes via comments are strongly recommended.

Note that you will not require any other libraries other than what JDK provides. You cannot use any external library for direct computation of greedy approach or dynamic programming approach. However, if you use any other external library, please include them in your zip file along with your report and code.



```java
public static void main(String[] args) {

    // Handling inputs and processing them
    ...

    System.out.println("============== TRIAL #1 ==============")
    System.out.println("Computer's Greedy Approach result")
    // print total attack points
    // print selected pieces for each level

    System.out.println("User's Dynamic Programming results")
    // print total attack points
    // print selected pieces for each level


    System.out.println("============== TRIAL #2 ==============")
    System.out.println("Computer's Random Approach result")
    // print total attack points
    // print selected pieces for each level

    System.out.println("User's Dynamic Programming results")
    // print total attack points
    // print selected pieces for each level

}
```

Code 3: *A sample main function. Also, print the execution time of each algorithms, too.*

Constrains

For this assignment, you will have the following constraints:

- $5 \leq$ *gold amount* ≤ 1200
- $1 \leq$ *maximum allowed level* ≤ 9
- $1 \leq$ *number of pieces for each level (i.e., piece type)* ≤ 25

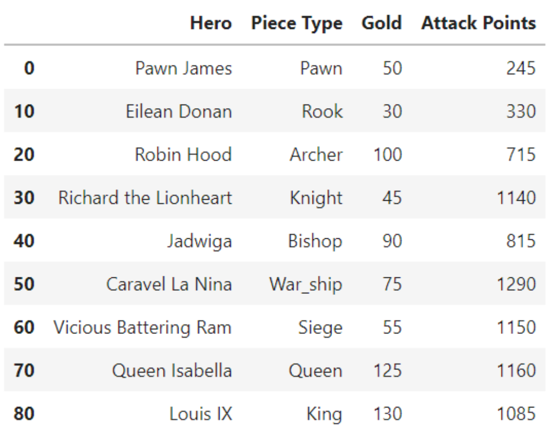
A sample data is attached in a text format for you to experiment with.

Report

You are expected to prepare a brief report where you will include your code explanations and a short discussion of your approaches. If any parts of your code are missing, state then clearly and attempt to describe your intuition. Also, a few sentences regarding the runtime of the approaches you have implemented will suffice.

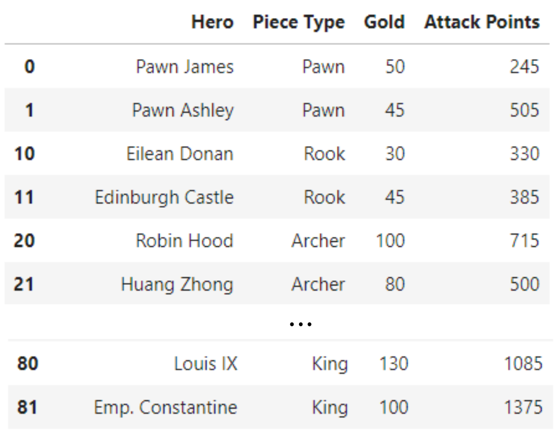
Using Input Data

The attached example input data contains 10 examples from each type. If number of pieces for each level is set to *n* where *n* is less than 10, that means you should use first *n* from each category.



	Hero	Piece Type	Gold	Attack Points
0	Pawn James	Pawn	50	245
10	Eilean Donan	Rook	30	330
20	Robin Hood	Archer	100	715
30	Richard the Lionheart	Knight	45	1140
40	Jadwiga	Bishop	90	815
50	Caravel La Nina	War_ship	75	1290
60	Vicious Battering Ram	Siege	55	1150
70	Queen Isabella	Queen	125	1160
80	Louis IX	King	130	1085

(a)



	Hero	Piece Type	Gold	Attack Points
0	Pawn James	Pawn	50	245
1	Pawn Ashley	Pawn	45	505
10	Eilean Donan	Rook	30	330
11	Edinburgh Castle	Rook	45	385
20	Robin Hood	Archer	100	715
21	Huang Zhong	Archer	80	500
		…		
80	Louis IX	King	130	1085
81	Emp. Constantine	King	100	1375

(b)

Figure 2: *Illustration of the selection of top `n` from each category.*
(a) *n=1, first heroes are selected;* (b) *n=2, first two heroes are selected.*

Grading Policy

Items	Percentage
Correct file naming	5%
Input file processing	10%
Dynamic programming approach	30%
Greedy approach	20%
Random approach	5%
Code design, naming conventions, good commenting, readability of the code	15%
Report	15%
Total	**100%**

Disclaimer

The hero names and their stats are chosen purely in a randomized fashion; hence these scores do not reflect any information regarding the historical significance.

Good luck!