Middle East Technical University   Department of Computer Engineering

CENG 315

Algorithms

Fall 2025-2026

Take-Home Exam 3

Due date: 9 November 2025, Sunday, 23.59

1 Problem Definition

A warrior is preparing for a final battle against a great boss. The warrior has **n** turns to acquire power-enhancing items and gather resources. The goal is to maximize their total attack power by the end of the n-th turn.

The warrior begins with an initial amount of gold, G_0.

In each turn t (from $t = 1$ to n), the warrior must choose exactly one of the following four actions:

1. **Buy a Potion**: This action costs $P[t][0]$ gold and adds $W[t][0]$ to the warrior's total attack power.
2. **Buy an Elixir**: This action costs $P[t][1]$ gold and adds $W[t][1]$ to the warrior's total attack power.
3. **Buy a Scroll**: This action costs $P[t][2]$ gold and adds $W[t][2]$ to the warrior's total attack power.
4. **Wait and Scavenge**: The warrior buys nothing and instead gains $G[t]$ gold. This action adds 0 to the warrior's total attack power.

A warrior can only choose to buy a potion, elixir, or scroll if their current gold is **greater than or equal to** its price. The attack power from all purchased items is cumulative.

Figure 1: The warrior preparing for the final battle.

Your task is to determine the maximum total attack power the warrior can achieve after all n turns are complete.

```
int find_max_attack(int n, int G0,
                    const vector<vector<int>>& P,
                    const vector<vector<int>>& W,
                    const vector<int>& G)
```

This function should return the maximum total attack power the warrior can achieve after n turns by optimally choosing which items to buy or when to scavenge for gold.

Parameters:

- **int n**: The total number of turns.
- **int G0**: The initial amount of gold the warrior starts with.
- **const vector<vector<int>>& P**: A 2D vector of size $n \times 3$ where $P[t][i]$ is the price of item i on turn t (0-indexed, so turn 0 to turn $n-1$). Index 0 represents a potion, index 1 represents an elixir, and index 2 represents a scroll.
- **const vector<vector<int>>& W**: A 2D vector of size $n \times 3$ where $W[t][i]$ is the attack power boost of item i on turn t.
- **const vector<int>& G**: A 1D vector of size n where $G[t]$ is the gold gained on turn t if the warrior chooses to wait and scavenge.

Returns:

- An integer representing the maximum total attack power achievable after all n turns.

2 Example I/O

Example 1: Simple 1-turn, buy best affordable

```
n = 1, G0 = 10
P = {{5, 8, 12}}
W = {{6, 10, 15}}
G = {3}
Expected output: 10
```

Example 2: 2-turn, Buy then Buy

```
n = 2, G0 = 10
P = {{5, 7, 100}, {5, 8, 100}}
W = {{6, 8, 100}, {7, 10, 100}}
G = {1, 1}
Expected output: 13
```

Example 3: 2-turn, Scavenge then Buy

```
n = 2, G0 = 5
P = {{100, 100, 100}, {8, 10, 100}}
W = {{1, 1, 1}, {10, 12, 100}}
G = {4, 1}
Expected output: 10
```

3 Specifications and Hints

- This is a **dynamic programming** problem. With a well-designed implementation, you don't need to track all turns simultaneously.
- For each turn and each possible gold amount, consider all 4 actions (buy potion, buy elixir, buy scroll, or scavenge).
- Be careful about the gold constraint: you can only buy an item if `current_gold` >= `item_price`.
- The gold amount can be quite large if the warrior scavenges many times. Consider the maximum possible gold for your solution.
- You will implement your solution in the `the3.cpp` file.
- You are free to add helper functions to `the3.cpp`.
- Do not change the first line of `the3.cpp`, which is `#include "the3.h"`
- Do not change the arguments and the return value of the function `find_max_attack()` in the file `the3.cpp`.
- Do not include any other library or write include anywhere in your `the3.cpp` file (not even in comments).
- You are given a `test.cpp` file to test your work on ODTUCLASS or your locale. You can and you are encouraged to modify this file to add different test cases.
- You can test your `the3.cpp` on the virtual lab environment. If you click run, your function will be compiled and executed with `test.cpp`. If you click evaluate, you will get feedback for your current work and your work will be temporarily graded with a limited number of inputs.
- The grade you see in VPL is not your final grade, your code will be reevaluated with more inputs after the exam.
- If you want to test your work and see your outputs on your locale you can use the following commands:

```
> g++ test.cpp the3.cpp -Wall -std=c++11 -o test
> ./test
```

4 Constraints and Limits

- The maximum number of turns is 1000.
- Item prices (potions, elixirs, scrolls), attack power boosts, and gold gains are non-negative integers.
- Maximum values for prices and gold are reasonable (typically ≤ 100).
- Initial gold G_0 is a non-negative integer (typically ≤ 100).
- **Recursive solutions are not allowed.** You must implement an iterative dynamic programming approach.
- The system has the following limits to test your solution:
  - a maximum execution time of 2 minutes

  - a 256 MB maximum memory limit
  - a maximum execution file size of 128 MB

- Solutions with longer running times, excessive memory usage, or stack overflow errors will not be graded.
- An efficient iterative dynamic programming solution should easily pass all test cases within the time and memory limits.
- If you are sure that your solution works in the expected complexity but your evaluation fails due to limits in the lab environment, please send an email.

5 Regulations

- **Implementation and Submission:** The template files are available in the Virtual Programming Lab (VPL) activity called "THE3" on ODTUCLASS. At this point, you have two options:
  - You can download the template files, complete the implementation, and test it with the given sample I/O on your local machine. Then submit the same file through this activity.
  - You can directly use the editor of the VPL environment by using the auto-evaluation feature of this activity interactively. Saving the code is equivalent to submitting a file.

  Please make sure that your code runs on ODTUCLASS. There is no limitation in running your code online. The last save/submission will determine your final grade.
- **Programming Language:** You must code your program in C++11. Your submission will be tested on the VPL environment in ODTUCLASS, hence you are expected to make sure your code runs successfully there.
- **Cheating: This assignment is designed to be worked on individually.** Additionally, the use of any LLMs (chatgpt, copilot, the other one that you are thinking about...) and copying code directly from the internet for implementations is strictly forbidden. Your work will be evaluated for cheating, and disciplinary action may be taken if necessary.
- **Evaluation:** Your program will be evaluated automatically using "black-box" testing, so make sure to obey the specifications. No erroneous input will be used. Therefore, you don't have to worry about invalid cases.
  Important Note: The given sample I/O's are only to ease your debugging process and NOT official. Furthermore, it is not guaranteed that they cover all the cases of required functions. As a programmer, it is your responsibility to consider such extreme cases for the functions. Your implementations will be evaluated by the official test cases to determine your final grade after the deadline.