Quiz 5C

CS/CE 412/471 Algorithms: Design and Analysis, Spring 2025

16 Apr, 2025. 4 questions, 25 points, 4 printed sides

Instructions:

1. Write your full name and your Hogwarts ID (or Muggle student ID) below.
2. You have 60 minutes to complete this enchanted assessment.
3. Wands away! Only your quill (pen), parchment (paper), and an optional calculator are permitted. Any magical artifacts, enchanted mirrors, or talking notebooks must be submitted with your bag at the front of the classroom.
4. Work independently — no use of Polyjuice Potion, Invisibility Cloaks, or Legilimency will be tolerated. Academic honesty is enforced by the strictest of Ministry decrees.
5. Present your spells (solutions) clearly and concisely. Illegible runes may result in lost points.
6. For your submission to be marked by the Council of Elders (your instructor), return this sheet along with your written solutions.
7. Have courage like a Gryffindor, wisdom like a Ravenclaw, diligence like a Hufflepuff, and ambition like a Slytherin. Good luck!

Student Name: ______________________________

Student ID: ______________________________

1. Best Beast Bonding with Limited Mana [5 points]

You're choosing magical creatures to bond with during your elective in Care of Magical Creatures. Each creature requires a certain amount of bonding mana and grants house points based on how powerful or rare it is. For creature i, let m_i be the mana required and h_i the house points awarded. Your total available mana is M. Choose a subset of creatures to maximize your total house points.

(a) $\boxed{\text{2 points}}$ For this problem, describe the solution and the value of the solution.

Solution: This is the 0-1 knapsack problem. The solution is the set of creatures, and the value of the solution is the corresponding house points.

(b) $\boxed{\text{3 points}}$ Give a recursive formulation for computing the maximum value. Clearly indicate the base case(s) and define any notation you introduce.

Solution: Let us number the creatures from 1 to n. Let $H(m, i)$ denote the maximum house points that can be obtained by consuming mana up to m and including creatures up to i. Then the required answer is $H(M, n)$ and

$$H(m,i) = \begin{cases} 0 & i = 0 \text{ or } m = 0 \\ -\infty & m < 0 \\ \max\{H(m - m_i, i-1) + h_i, H(m, i-1)\} & 1 \le i \le n \end{cases}$$

2. Upgrade or Overspend? [5 points]

You want to upgrade your broomstick for Quidditch. There are different magical broomstick parts available, each with a price in Sickles and an aerodynamic improvement score. You may buy multiple copies of any part (unlimited supply). Starting with B Sickles, Ron Weasley follows the strategy to keep buying the part with the best improvement that he can afford.

Is this strategy guaranteed to give the optimal improvement? Justify.

Solution: This is the unbounded knapsack problem. We prove the strategy sub-optimal through a counterexample.

Proof. The strategy does not always provide the maximum improvement.

Consider $B = 10$ and two parts, one with price and improvement both equal to 9 and the other with price and improvement both equal to 5.

The strategy leads to a purchase of part 1 and a total improvement of 9.

A more optimal solution, 2 purchases of part 2 with a total improvement of 10, exists. □

3. Do You Need a Time-Turner? [5 points]

Below are magical computing problems. State whether dynamic programming is suitable for each. Briefly justify your choice (do not solve).

(a) $\boxed{\text{1 point}}$ Wand Core Fusion: Given a desired power level P and a list of wand cores, each with a power level, can you fuse a subset to get exactly P?

Solution: Dynamic programming is a suitable approach. For each core, we decide whether to include or exclude it in the subset. Multiple sequences of decisions can lead to the same subproblem, so subproblems overlap.

(b) $\boxed{\text{1 point}}$ Dark Maze Escape: You are in a grid with curses and traps. From each cell, you can move right or down. Maximize the health you retain when reaching the bottom-right.

Solution: Dynamic programming applies. This is an optimization problem. Multiple paths from the start position can lead to a given position in the grid, so the solution will involve overlapping subproblems.

(c) $\boxed{\text{1 point}}$ Spellbook Sorting: You are given a list of spell names. Sort them lexicographically.

Solution: Dynamic programming is not a suitable approach. Sorting does not involve overlapping subproblems.

(d) $\boxed{\text{1 point}}$ Magical Duel Simulator: Given a fixed elimination tree of wizards, simulate the entire duel to find the winner.

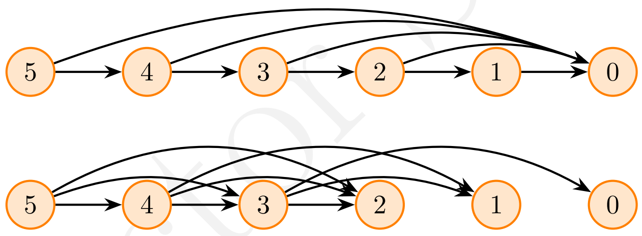
Solution: Dynamic programming is not a suitable approach. The duels can be directly simulated to find the winner. There are no overlapping subproblems.

(e) $\boxed{\text{1 point}}$ Magic Path Weaving: Given musical incantations of fixed durations, how many ways can you combine them into a chant of length L?

Solution: Dynamic programming applies. The explanation is similar to the first part.

4. Subproblem Symphonies [10 points]

Two directed acyclic graphs are shown below, each representing subproblem dependencies for some magical task.



For each graph:

(a) Provide a recurrence formulation that fits the dependency graph.

(b) Estimate the time complexity of the solution.

(c) Argue whether dynamic programming is an appropriate approach.

(d) Provide a bottom-up pseudocode implementation for general n.

Solution: Left graph.

(a) a plausible recurrence matching the graph:

$$f(n) = \begin{cases} a_0 & n = 0 \\ \min\{a_i + f(i) : i \in \{0, n-1\}\} & \text{otherwise} \end{cases}$$

where $a = \langle a_0, a_1, a_2, \ldots, a_n \rangle$ is an input array.

(b) an estimate of the time complexity of the solution:
All but 1 of the n nodes connect to 2 nodes each. The time complexity is therefore $\Theta(n)$.

(c) a justification whether dynamic programming is a suitable approach to solve the problem:
A dynamic programming approach is suitable. There is overlap, albeit superficial, among the subproblems.

(d) a bottom up pseudocode implementation of your recurrence for general n:

```
F($n, a$)
1  initialize $dp = \langle dp_0, dp_1, dp_2, \dots, dp_n \rangle$
2  $dp_0 = a_0$
3  for $i = 1$ to $n$
4      $dp_i = \min(a_{i-1} + dp_{i-1}, a_0 + dp_0)$
```

Solution: Right graph.

(a) a plausible recurrence matching the graph:

$$f(n) = \begin{cases} a_n & n \leq 2 \\ \min\{a_i + f(i) : n-3 \leq i < n\} & \text{otherwise} \end{cases}$$

where $a = \langle a_0, a_1, a_2, \dots, a_n \rangle$ is an input array.

(b) an estimate of the time complexity of the solution:
All but 3 of the n nodes connect to 3 nodes each. The time complexity is therefore $\Theta(n)$.

(c) a justification whether dynamic programming is a suitable approach to solve the problem:
A dynamic programming approach is not suitable. The subproblems do not overlap,

(d) a bottom up pseudocode implementation of your recurrence for general n:

```
F($n, a$)
1  initialize $dp = \langle dp_0, dp_1, dp_2, \dots, dp_n \rangle$
2  $dp_0, dp_1, dp_2 = a_0, a_1, a_2$
3  for $i = 3$ to $n$
4      $dp_i = \min(a_{i-1} + dp_{i-1}, a_{i-2} + dp_{i-2}, a_{i-3} + dp_{i-3})$
```