Quiz 5A

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. Maximize House Points with Magical Energy [5 points]

Course enrollment for next term is open at Hogwarts University! Each course consumes a certain amount of *Magical Energy* and grants a *House Points Bonus* depending on how exciting or rewarding it is. That is, for each course i, the magical energy required is e_i, and the house point bonus is p_i. You cannot expend more than E magical energy in total. You want to choose a subset of courses to maximize your total house point bonus next semester.

(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 courses, 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 courses from 1 to n. Let $P(e, i)$ denote the maximum house points that can be obtained by expending energy up to e and including courses up to i. Then the required answer is $P(E, n)$ and

$$P(e,i) = \begin{cases} 0 & i = 0 \text{ or } e = 0 \\ -\infty & e < 0 \\ \max\{P(e - e_i, i-1) + p_i, P(e, i-1)\} & 1 \le i \le n \end{cases}$$

2. Greedy Gobbles or Dynamic Dilemmas? [5 points]

Rahimov Rahmania's *Phoenix Puffs* are the new craze at Hogwarts! Each type of Phoenix Puffs is priced in Sickles and carries a magical *satiation score* — a measure of how satisfied it makes a hungry witch feel.

You may buy any number of each type of Phoenix Puff (unlimited supply). Each type i has a price p_i (in Sickles) and a satiation s_i.

Luna Lovegood wants to maximize her total satiation without exceeding her total Sickles. Starting with T sickles, she will keep buying the most satiating type of Phoenix Puff that she can still afford.

Prove or disprove that this strategy always provides her the maximum satiation.

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 satiation.

Consider $T = 10$ and two types with $p_1 = s_1 = 9$ and $p_2 = s_2 = 5$.

The strategy leads to a purchase of item 1 and a total satiation of 9.

A more optimal solution, 2 purchases of item 2 leading to a total satiation of 10, exists. □

3. Which Spell Needs Memoization? [5 points]

Below are several magical computation problems. For each, state whether dynamic programming is a suitable approach. Justify briefly (do not solve the problem).

(a) $\boxed{\text{1 point}}$ Magical Spell Combination: Given a target spell power S, and a list of spell fragments, each with a power value, determine if it is possible to combine some fragments to reach exactly S.

Solution: Dynamic programming applies. For each fragment, we decide whether to include or exclude it in the subset, and repeat for the remaining fragments. Multiple sequences of decisions can lead to the same subproblem. So subproblems overlap.

(b) $\boxed{\text{1 point}}$ Potion Gathering Quest: You are at the top-left cell of a $n \times n$ grid. At each step, you can move right or down. Each cell contains a potion with a value. Find a path to the bottom-right cell that collects the maximum total value.

> **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) $\boxed{\text{1 point}}$ Sorting Owls: Given a list of owl delivery times, sort them in ascending order.

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

(d) $\boxed{\text{1 point}}$ Wizard Duel Elimination: In a tournament of n wizards, each duel eliminates one. Given the tournament tree, simulate all duels and report the winner.

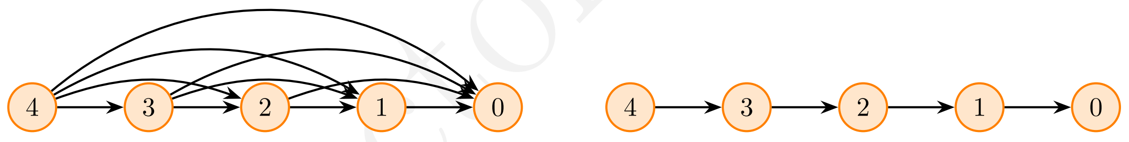
> **Solution:** Dynamic programming is not a suitable approach. The duels can be directly simulated to obtain the winner. There are no overlapping subproblems.

(e) $\boxed{\text{1 point}}$ Magical Melody Construction: Given a set of musical phrases, find the number of ways to arrange them into a melody of exactly T beats.

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

4. Graphical Spells: Cast or Skip? [10 points]

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



For each graph, provide:

(a) a plausible recurrence matching the graph,

(b) an estimate of the time complexity of the solution,

(c) a justification whether dynamic programming is a suitable approach to solve the problem, and

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

> **Solution:** Left graph.
>
> (a) a plausible recurrence matching the graph:
>
> $$f(n) = \begin{cases} a_0 & n = 0 \\ \min\{a_i + f(i) : 0 \le i < n\} & \text{otherwise} \end{cases}$$
>
> 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:
> Each of the n nodes connects to $n - 1$ nodes. The time complexity is therefore $\Theta(n^2)$.
>
> (c) a justification whether dynamic programming is a suitable approach to solve the problem:
> A dynamic programming approach is suitable. The subproblems overlap,

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

```
F($n, a$)
1  initialize $dp = \langle dp_0, dp_1, dp_2, \ldots, dp_n \rangle$
2  $dp_0 = a_0$
3  for $i = 1$ to $n$
4      $dp_i = \infty$
5      for $j = i - 1$ downto 0
6          $dp_i = \min(dp_i, a_j + dp_j)$
```

Solution: Right graph.

(a) a plausible recurrence matching the graph:

$$f(n) = \begin{cases} a_0 & n = 0 \\ a_n + f(n-1) & \text{otherwise} \end{cases}$$

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:
Each of the n nodes connects to 1 node. 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 $dp = \langle dp_0, dp_1, dp_2, \ldots, dp_n \rangle$
2  $dp_0 = a_0$
3  for $i = 1$ to $n$
4      $dp_i = a_i + dp_{i-1}$
```