

Question?

[PollEv.com/erickhiu063](https://PollEv.com/erickhiu063)



# Midterm Review Session 2

Eric Khiu

3/4/24 6-8pm @ BBB 1670

Handout available at course drive/Exam/Midterm Review

Annotated version will be uploaded tonight

# Exam Logistics

- ▶ Midterm: Wednesday March 6th, 7-9pm
- ▶ Room assignments on [drive](#)
- ▶ You may bring one double-sided 8.5 x 11 study sheet
- ▶ No calculators
- ▶ We will provide scratch papers

Question?

[PollEv.com/erickhiu063](https://pollEv.com/erickhiu063)



# Exam Format

- ▶ 5 multiple answer multiple choice
  - ▶ 3 single answer multiple choice
  - ▶ 4 short answer
  - ▶ 2 long answer
- 
- ▶ See Piazza @1003 for details

Question?

[PollEv.com/erickhiu063](https://pollEv.com/erickhiu063)

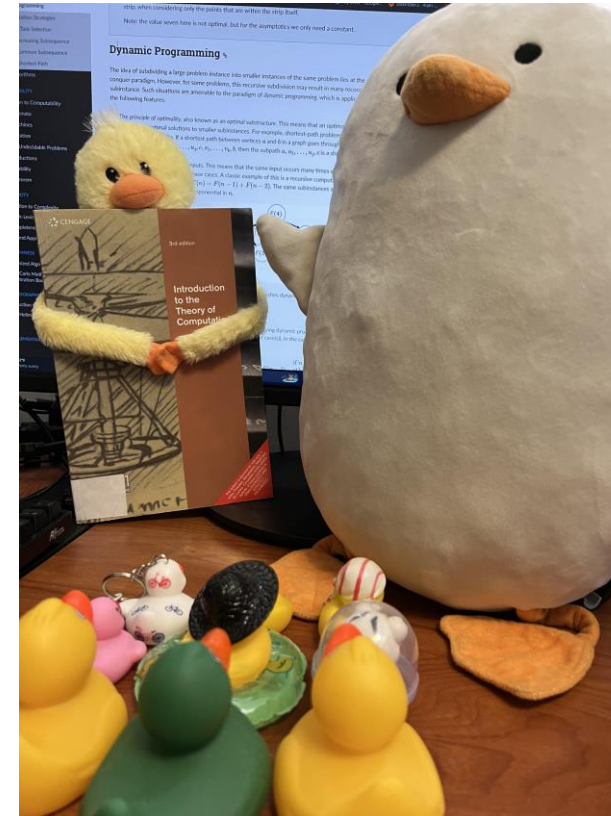


# Selected Problems

- ▶ WN 22 Short 1- Decidability
- ▶ WN 22 Long 1- DP: 2ASP (See Review 1 with Daphne)
- ▶ WN 22 Long 2- Turing Reduction:  $L_{BOB}$
- ▶ WN 22 Long 3- Potential Method
- ▶ WN 23 Short 1- Asymptotic Bound
- ▶ WN 23 Long 1- DP: Max sum decreasing subsequence

Question?

[PollEv.com/erickhiu063](https://poll-ev.com/erickhiu063)



# DP Recipe

Question?

[PollEv.com/erickhiu063](https://pollEv.com/erickhiu063)



- ▶ Write recurrence
  - ▶ Choose the subject of recurrence
  - ▶ Base case(s)
  - ▶ Form optimal sub-solution (“up to this point”)
- ▶ Size of table (Dimensions? Range of each dimensions?)
- ▶ To fill in cell, which other cells do I look at?
- ▶ Which cell(s) contain the final answer?
- ▶ Reconstructing solution: Follow arrows from final answer to base case



# DP Recurrence Overview

- ▶ **Step 1: Define the subject of recurrence**
  - ▶ Is this a 1D DP problem? 2D? What **dimension** do we want to do it in?
  - ▶ In which/ how many **direction(s)** do we want to **reduce** the problem?
- ▶ **Step 2: Identify the base case(s)**
  - ▶ In what situation(s) we **can't reduce** the problem **further**?
  - ▶ Is there any **special cases**?
- ▶ **Step 3: Construct (optimal) sub-solution**
  - ▶ Sub-solution:
    - ▶ [Sub] How to **reduce** the problem to **smaller** version of the same problem?
    - ▶ [Solution] How to **combine** the result so that the **overall result is correct**?
    - ▶ If [some condition] is/ isn't satisfied, what options do we have?
  - ▶ Optimal: (only for optimization problem)
    - ▶ Is it a **maximization** or **minimization** problem?
    - ▶ What is/ are the **variable(s)** we're taking max/ min over?
    - ▶ What is **the objective function** to be maximized/ minimized?



# Turing Reductions Overview

- ▶ Suppose we want to show that  $A \leq_T B$
- ▶ **Step 1: Identify the inputs of  $D_A$  and  $D_B$** 
  - ▶ Is the input a number? A string? Multiple strings? A machine?
- ▶ **Step 2: Draft Desired Behavior of  $D_A$** 
  - ▶ Choose between “return same” and “return opposite”
  - ▶ **Return same:**  $x \in A \Leftrightarrow \dots \Leftrightarrow x' \in B \Leftrightarrow D_B(x') \text{ accepts} \Leftrightarrow D_A(x) \text{ accepts}$
  - ▶ **Return opposite:**  $x \in A \Leftrightarrow \dots \Leftrightarrow x' \notin B \Leftrightarrow D_B(x') \text{ rejects} \Leftrightarrow D_A(x) \text{ accepts}$
- ▶ **Step 3: Generate input(s) for  $D_B$** 
  - ▶ **Return same:** How to generate  $x'$ , possibly using  $x$ , such that  $x \in A \Rightarrow x' \in B$  and  $x \notin A \Rightarrow x' \notin B$ ?
  - ▶ **Return opposite:** How to generate  $x'$ , possibly using  $x$ , such that  $x \in A \Rightarrow x' \notin B$  and  $x \notin A \Rightarrow x' \in B$ ?

Note: Here we condense the two cases using iff

Question?

[PollEv.com/erickhiu063](https://PollEv.com/erickhiu063)



# Selected Problems



UNIQNAME (print): WN 22 Short 1

---

### Shorter Answer – 30 points

1. (8) Briefly prove or disprove the following statement:  
If  $L_1, L_2 \subseteq \{0, 1\}^*$  are undecidable languages and  $L_1 \neq L_2$ , then their symmetric difference,  $(L_1 \setminus L_2) \cup (L_2 \setminus L_1)$ , is also an undecidable language.

## Proofs and longer questions – 40 points

Answer two of the following three questions. **Clearly cross out the question you do not want graded.** If it isn't clear what problem you don't want graded, we will grade the first two. Each of the two questions are worth 20 points.

1. The **2-Arithmetic Subsequence Problem** is defined as follows: Given a sequence of  $n$  positive integers  $a_1, \dots, a_n$ , find the length of the longest subsequence where each value in the subsequence is increasing by 2. For example, in the sequence 6, 1, 2, 3, 12, 4, 5, 6, the length of the longest increasing-by-two subsequence is 3 and is achieved by the subsequence 2, 4, 6 (also by 1, 3, 5).
  - (a) Find a recurrence relation that can be used to solve the 2ASP.
  - (b) Write pseudocode which solves the problem using dynamic programming. It should have a run time that is  $O(n^2)$ .

**UNIQNAME** (print): WN 22 Long 2

---

2. Let  $L_{bob} = \{\langle M \rangle \mid \langle \text{"bob"} \rangle \notin L(M)\}$ . Show that  $L_{ACC} \leq_T L_{bob}$  or show that  $L_{HALT} \leq_T L_{bob}$ .  
(Do whichever of the two you would prefer.)

3. Consider the following two-player game on a  $12 \times 12$  grid. At the beginning of the game, there is at most one stone occupying each square (so each square either has 1 or 0 stones). On their turn, each player must pick a stone and remove it. Subsequent to removing the stone, the player can place or remove stones as desired, on any of the squares to the **right** of that stone (on the same row). No square may ever have more than one stone. The game ends when there are no more stones left on the board.

Are there any initial board states **and** sets of moves that can make the game continue indefinitely, or will the game always end **regardless** of the initial position and the moves of the players? If so, describe them. If not, provide a proof that demonstrates this.

**Hint:** Try simulating the game on a  $4 \times 4$  grid.

### Shorter Answer – 24 points

Each of the three questions in this section is worth 8 points.

1. Give the **tightest correct asymptotic** (big- $O$ ) bound, as a function of  $n$ , on the **worst-case number of additions** done by the following algorithm, along with **a value of  $k$  that induces the worst case**. Also **state whether this is polynomial in the input size** or not. No explanation or proof is needed.

```

1: function FUNK( $n, k$ )                                 $\triangleright n$  is a positive integer, and  $k \in \{1, 2, \dots, n\}$ 
2:    $x = 0$ 
3:   for  $i = 1, 2, \dots, k$  do
4:     for  $j = 1, 2, \dots, n - k$  do
5:        $x = x + 1$ 
6:   return  $x$ 

```

## Proofs and Longer Answers – 40 points

Each of the two questions in this section is worth 20 points.

1. Let  $A[1, \dots, n]$  be an array of  $n \geq 1$  positive real numbers. A decreasing subsequence of  $A$  is a sequence of array elements  $A[i_1] > A[i_2] > \dots > A[i_m]$ , where  $i_1 < i_2 < \dots < i_m$  are some (not necessarily contiguous) array indices.

We are interested in the maximum sum obtainable by decreasing subsequences of  $A$ , i.e.,

$$\max\{A[i_1] + \dots + A[i_m] : A[i_1] > A[i_2] > \dots > A[i_m] \text{ is a decreasing subsequence of } A\}.$$

For example, for  $A = [4, 2, 2, 3, 5, 1]$ , both  $S_1 = [4, 3, 1]$  and  $S_2 = [5, 1]$  are decreasing subsequences. The sum over  $S_1$  is  $8 = 4 + 3 + 1$ , whereas the sum over  $S_2$  is  $6 = 5 + 1$ . It can be verified that  $S_1$  has the maximum sum overall, i.e., no decreasing subsequence of  $A$  has a sum greater than 8. (Note that the subsequence  $[4, 2, 2, 1]$  has a sum of 9, but it is not decreasing, because  $2 \not> 2$ .)

- (a) Let  $H(i)$  denote the maximum sum obtainable by a decreasing subsequence of  $A$  that ends at index  $i$ , i.e.,

$$H(i) = \max\{A[i_1] + \dots + A[i_m] : A[i_1] > \dots > A[i_m] \text{ is a decreasing subsequence of } A \text{ with } i_m = i\}.$$

Give a correct recurrence relation for  $H(i)$ , including base case(s), that is suitable for an efficient dynamic-programming algorithm. Briefly justify your answer.

Hint: an optimal decreasing subsequence ending at index  $i$  is either a single element, or has an immediate predecessor (a second-to-last element). What are the possible indices for this predecessor? What is true about the part of the subsequence that ends at this predecessor?

UNIQNAME (print): WN 23 Long 1

---

- (b) Using your answer to part (a), **describe a dynamic-programming algorithm** that outputs the maximum sum over all decreasing subsequences of an input array  $A$ . Also **give the tightest correct asymptotic (big- $O$ ) running time** for your algorithm, as a function of  $n$ . Assume that addition, comparisons, and similar basic operations take constant time.