Question?
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

Question?
PollEv.com/erickhiu063

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

Exam Format

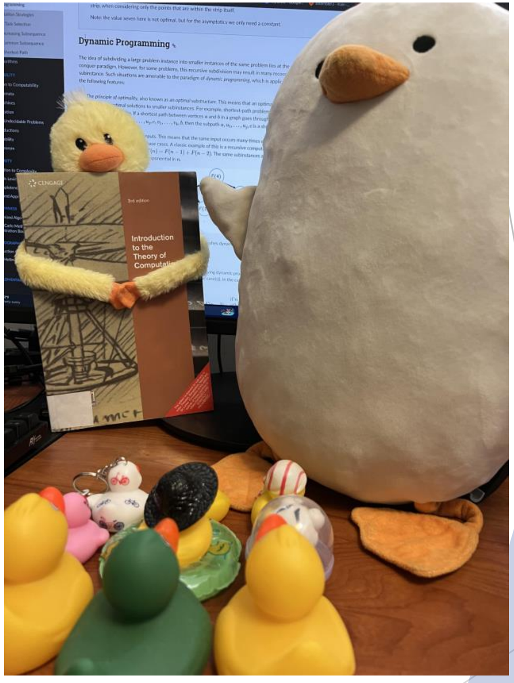
- 5 **multiple answer** multiple choice
- 3 **single answer** multiple choice
- 4 **short answer**
- 2 **long answer**

- See Piazza @1003 for details

Question?
PollEv.com/erickhiu063

Selected Problems

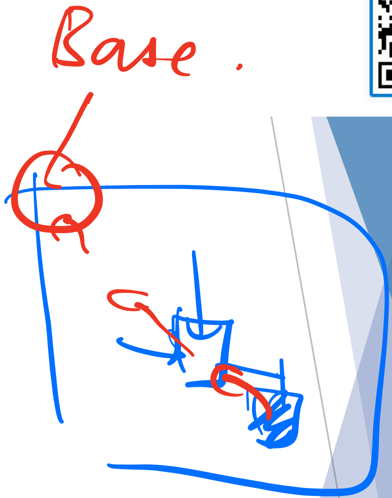
- WN 22 Short 1- Decidability
- WN 22 Long 1- DP: 2ASP (See Review 1 with Daphne)
- WN 22 Long 2- Turing Reduction: L_{BOB}
- WN 22 Long 3- Potential Method
- WN 23 Short 1- Asymptotic Bound
- WN 23 Long 1- DP: Max sum decreasing subsequence



Question?
PollEv.com/erickhiu063

DP Recipe

- 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** (English)
  - 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?

Into

max $\boxed{\text{obj value}}$???

Turing Reductions Overview

Question?
PollEv.com/erickhiu063

- 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: $x \in A \Leftrightarrow \cdots \Leftrightarrow x' \in B \Leftrightarrow D_B(x')$ accepts $\Leftrightarrow D_A(x)$ accepts
  - Return opposite: $x \in A \Leftrightarrow \cdots \Leftrightarrow x' \notin B \Leftrightarrow D_B(x')$ rejects $\Leftrightarrow D_A(x)$ 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

Selected Problems

UNIQNAME (print): WN 22 Short 1

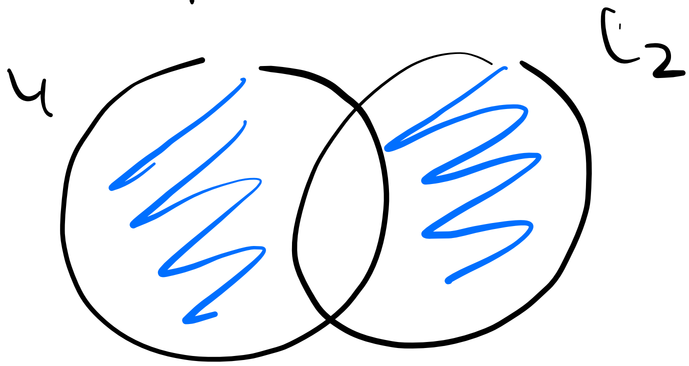
Shorter Answer – 30 points

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.

Dec: Σ^*, ϕ, finite lang.

Undec: L_{ACC}, L_{HALT}, L_{BARBER}.

Sym. diff



Idea: Think of a case where sym. diff is dec.
If possible $\Rightarrow$ False.

Reminder: $L_1 \neq L_2$

Intuitive approach: $L_2 = L_1^C$

$\{0,1\}^* \in$ Dec.

$L_2 = L_1^c$ $\qquad$ $L_1, L_2 \subseteq \{0,1\}^*$

$L_1 \setminus L_2 = L_1$ $\qquad$ $L_1 \cup L_2 = \{0,1\}^*$

$L_2 \setminus L_1 = L_2$ $\qquad$ bee.

UNIQNAME (print): WN 22 Long 1

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 $a_1, \ldots, a_n$, 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 $1, 3, 5$).

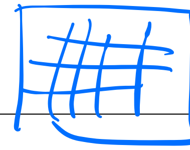
   (a) Find a recurrence relation that can be used to solve the 2ASP.

   (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.)

UNIQNAME (print): WN 22 Long 3



3. Consider the following two-player game on a 12×12 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 4×4 grid.

101

Potential Method:

potential func: strictly decreasing after every time interval

0 0 . . . | 1 |

previous 0 ↓ will be 0 ↑ Remove throughout.

1 0 0 0 1 → bin. string.

Each row: binary string. bin → dec.

+

P(Σ val. represented by each bin string)

Let $b_{i,j}$ = binary string on row i on the jth move. $i = 1, \ldots, 12$

Let potential function on move j be:

$$p(j) = \sum_{i=1}^{12} b_{i,j}$$

since on every round, $\exists\, i \in \{1, \ldots, 12\}$ s.t. the val. represented by b_i decreased. say: $\boxed{b_{1,j+1} < b_{1,j}}$

WLOG

Therefore,

$$p(j+1) = b_{1,j+1} + \sum_{i=2}^{12} b_{i,j+1}$$

$$< b_{1,j} + \sum_{i=2}^{12} b_{i,j+1}$$

$p(j+1) = \frac{2}{3} p(j)$

$$= \sum_{i=1}^{12} b_{i,j+1} = p(j)$$

since $p(j+1) < p(j) \quad \forall j$

$\rightarrow$ strictly decreasing.

Every row: $2^{12}-1$ max val. repr. by bin. string length n:

bin. string

max val. repr. $= 2^n - 1$

Since: $P(j)$ will decrease on every round

worst: Decrease by 1.

Consider: max. possible initial value for $p(j) = 12$. max. val. on each row.

$= 12(2^{12}-1)$

UNIQNAME (print): WN 23 Short 1

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)          ▷ n is a positive integer, and k ∈ {1, 2, ..., n}
2:     x = 0
3:     for i = 1, 2, ..., k do
4:         for j = 1, 2, ..., n − k do
5:             x = x + 1
6:     return x
```

Input size, $s = \log_2 n + \log_2 k$
$= \log_2 nk$

$\Rightarrow \boxed{nk = 2^{\text{input\_size}}}$

Obs: As $k\uparrow$, $n-k$ $\downarrow$ #inner iter $\downarrow$

But if $k=0$: won't even have outer iter.

$\boxed{k = \frac{n}{2}}$

outer: $k = \frac{n}{2}$ iter.

Inner: $n-k = n - \frac{n}{2} = \frac{n}{2}$

$O\left(\frac{n}{2}\left(\frac{n}{2}\right)\right) = O\left(\frac{n^2}{4}\right) = O(n^2)$

$n\left(\frac{n}{2}\right) = \frac{n^2}{2} \Rightarrow \frac{n^2}{2} = 2^{\text{input\_size}}$

$n^2 = 2^{\text{input\_size}+1}$

$|V|$

$|E|$

$O(n^2) = O\left(2^{\text{input\_size}}\right) \to$ exp.

UNIQNAME (print): WN 23 Long 1

Proofs and Longer Answers – 40 points

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

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

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

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

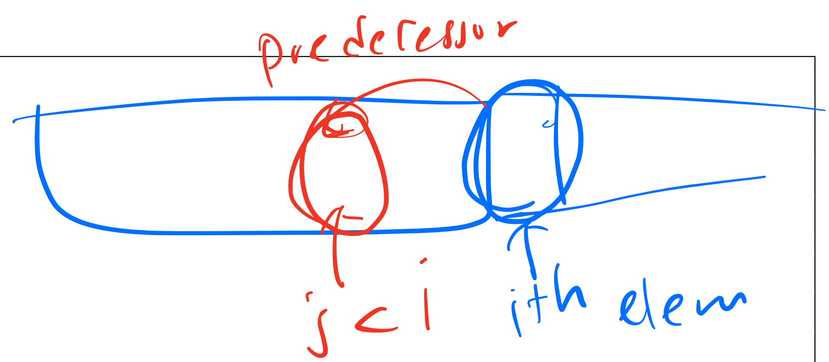
   For example, for $A = [4, 2, 2, 3, 5, 1]$, both $S_1 = [4, 3, 1]$ and $S_2 = [5, 1]$ are decreasing subsequences. The sum over S_1 is $8 = 4 + 3 + 1$, whereas the sum over S_2 is $6 = 5 + 1$. It can be verified that S_1 has the maximum sum overall, i.e., no decreasing subsequence of A has a sum greater 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.,

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

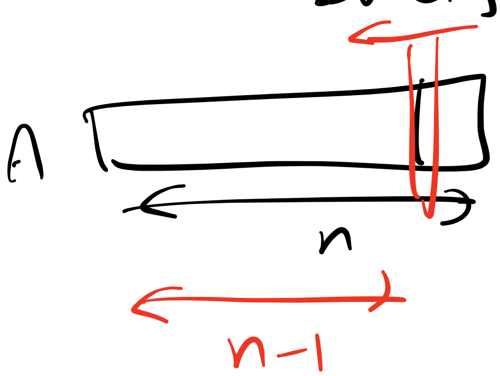
   Give a correct recurrence relation for $H(i)$, including base case(s), that is suitable for an efficient dynamic-programming algorithm. Briefly justify your answer.

   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?



Step 1:

H(i) = max sum .. decreasing subseq
ending at i



Reduce in 1 direction: 1D DP

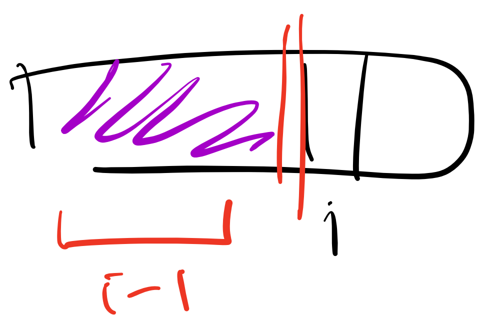
Step 2

Base: ~~when~~ we can't reduce
further when length = 1

Return A[i]

Step 3:

[sub]



smaller ver. of
same prob → max sum.
decreasing
subseq.

[Tol] How to use info we
get from

We know: All max sum decreasing subseq ending at each $j < i$

[combine] $H(i)$ = max sum d s ending at i.

But: we can't simply pick.

must choose: decreasing subseq ending at i

Guaranteed by previous optimality.

$A[j] > A[i]$

$H(j)$

[OPT] maximization prob: we max

max over?

$j < i$ s.t. $A[j] > A[i]$

obj. func?

$H(j)$

Add to $A[i]$ → updated sum (ending at i)

Recurrence:

$$H(i) = \begin{cases} A[i] & \text{if } i=1 \text{ or } \forall j<i, A[j]<A[i] \\ A[i] + \max\limits_{\substack{j<i: \\ A[j]>A[i]}} A[j] & \text{o.w.} \end{cases}$$

Edgecase: what if we don't have such j?

$\forall j < i,\ A[j] < A[i]$

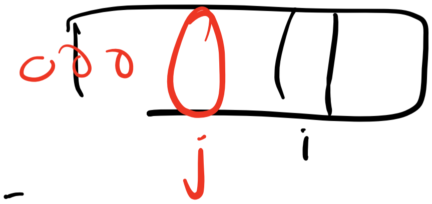
Take $A[i]$ ← ith start of subseq

$$H(i) = \begin{cases} A[i] & \text{if } i=1 \text{ or } \forall j<i, A[j]<A[i] \\ A[i] + \max\limits_{\substack{j<i: \\ A[j]>A[i]}} A[j] & \text{o.w.} \end{cases}$$

Correctness Pf

S_i = decreasing subseq. w/ max sum ending at i.

$S_i' = S_i \setminus A[i]$



decreasing subseq ending at j.

<u>claim</u> S_i' must be optimal decreasing subseq. ending at j

Consider $S_i'' \neq S_i'$

suppose S_i'' has greater sum that S_i'

1. same j, we would have pick S_i'' for that j $\quad H(j)$ = val of S_i''

2. different j would have chose that j $\uparrow$ S_i''

   $\Rightarrow S_i'$ wasn't the optimum.

In either case, $H(i)$ = max sum. decreasing sub ending at i. 7:34

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.

$$H(i) = \begin{cases} A[i] & \text{if } i = 1 \text{ or } \forall j < i, A[j] < A[i] \\ A[i] + \max\limits_{j<i \,:\, A[j] > A[i]} A[j] & \text{o.w.} \end{cases}$$

Alg

MSDS(A[1, ..., n]):

memo ← empty array length n

memo[1] ← A[1]

for i = 2, ..., n:

// Check if we have valid j

$\boxed{\text{if } A[j] < A[i]\ \forall j < i \quad \text{memo} \leftarrow A[i]}$ equivalent

O(n) O(n²) O(n)

has-predecessor ← False

for j = 1, ..., i-1

if A[j] > A[i] then

has_predecessor ← True.

break

if has-predecessor = False:

memo ← A[i]

else:

memo[i] ← A[i] + $\max\limits_{j<i}$: A[j]

$A[j] > A[i]$

equivalent

$O(n)$

else:

temp_max $\leftarrow -\infty$

for $j = 1, \ldots, i-1$

if $H[j] >$ temp_max :

temp_max = $H[j]$

memo$[i] \leftarrow A[i] +$ temp_max

$O(n)$ Return $\max_i$ memo$[i]$

// Note: can do for loop similarly

Runtime Analysis :

$O(n^2) + O(n) = O(n^2)$