CSL 356: Analysis and Design of Algorithms

Ragesh Jaiswal

CSE, IIT Delhi

Techniques

- Greedy Algorithms
- Divide and Conquer
- Dynamic Programming
- Network Flows
- Computational Intractability

Dynamic Programming

Dynamic Programming: Main Ideas

- *Break the given problem into a* ***few*** *sub-problems and combine the optimal solution of the smaller sub-problems to get optimal solutions to larger ones.*
  - How is it different from Divide-and-Conquer?
    - Here you are allowed to have overlapping sub-problems.

- *Suppose your recursive algorithm gives a recursion tree that has many common sub-problems (e.g., recursion for computing fibonacci numbers), then it helps to save the solution of sub-problems and use this solution whenever the same sub-problem is called.*
  - Dynamic programming algorithms are also called **table-filling** algorithms.

Dynamic Programming: Examples

- Problem (longest increasing subsequence): You are given a sequence of integers $A[1], \dots, A[n]$ and you are asked to find the *longest increasing subsequence* of integers.
  - Example: The longest increasing subsequence of the sequence $(7, 2, 8, 6, 3, 6, 9, 7)$ is $(2, 3, 6, 7)$.

- Let $L(i)$ denote the length of the longest increasing subsequence that ends with the number $A[i]$.
- What is $L(1)$?

Dynamic Programming: Examples

- Problem (longest increasing subsequence): You are given a sequence of integers $A[1], \dots, A[n]$ and you are asked to find the *longest increasing subsequence* of integers.
  - Example: The longest increasing subsequence of the sequence $(7, 2, 8, 6, 3, 6, 9, 7)$ is $(2, 3, 6, 7)$.

- Let $L(i)$ denote the length of the longest increasing subsequence that ends with the number $A[i]$.
- What is $L(1)$?
  - $L(1) = 1$.

Dynamic Programming: Examples

- Problem (longest increasing subsequence): You are given a sequence of integers $A[1], \ldots, A[n]$ and you are asked to find the *longest increasing subsequence* of integers.
  - Example: The longest increasing subsequence of the sequence $(7, 2, 8, 6, 3, 6, 9, 7)$ is $(2, 3, 6, 7)$.

- Let $L(i)$ denote the length of the longest increasing subsequence that ends with the number $A[i]$.
- What is $L(1)$?
  - $L(1) = 1$.
- What is the value of $L(i)$ in terms of $L(1), \ldots, L(i-1)$?

Dynamic Programming: Examples

- Problem (longest increasing subsequence): You are given a sequence of integers $A[1], \ldots, A[n]$ and you are asked to find the *longest increasing subsequence* of integers.
  - Example: The longest increasing subsequence of the sequence $(7, 2, 8, 6, 3, 6, 9, 7)$ is $(2, 3, 6, 7)$.

- Let $L(i)$ denote the length of the longest increasing subsequence that ends with the number $A[i]$.
- What is $L(1)$?
  - $L(1) = 1$.
- What is the value of $L(i)$ in terms of $L(1), \ldots, L(i-1)$?
  - $L(i) = \max_{j<i, A[j] \leq A[i]} (1 + L(j))$.

Dynamic Programming: Examples

- Let $n = 9$ and $(A[1], \dots, A[9]) = (7, 2, 8, 6, 3, 1, 9, 7, 10)$.
  - $L(1) = 1$
  - $L(2) = 1$
  - $L(3) = 2$
  - $L(4) = 2$
  - $L(5) = 2$
  - $L(6) = 1$
  - $L(7) =$

Dynamic Programming: Examples

- Let $n = 9$ and $(A[1], \ldots, A[9]) = (7, 2, 8, 6, 3, 1, 9, 7, 10)$.
  - $L(1) = 1$
  - $L(2) = 1$
  - $L(3) = 2$
  - $L(4) = 2$
  - $L(5) = 2$
  - $L(6) = 1$
  - $L(7) = \max(2, 2, 3, 3, 3, 2) = 3$
  - $L(8) = \max(2, 2, 3, 3, 2) = 3$
  - $L(9) = \max(2, 2, 3, 3, 3, 2, 4, 4) = 4$

Dynamic Programming: Examples

- Let $n = 9$ and $(A[1], \dots, A[9]) = (7, 2, 8, 6, 3, 1, 9, 7, 10)$.
  - $L(1) = 1$
  - $L(2) = 1$
  - $L(3) = 2$
  - $L(4) = 2$
  - $L(5) = 2$
  - $L(6) = 1$
  - $L(7) = \max(2, 2, 3, 3, 3, 2) = 3$
  - $L(8) = \max(2, 2, 3, 3, 2) = 3$
  - $L(9) = \max(2, 2, 3, 3, 3, 2, 4, 4) = 4$
- What is the length of the longest increasing subsequence?

Dynamic Programming: Examples

- Let $n = 9$ and $(A[1], \ldots, A[9]) = (7, 2, 8, 6, 3, 1, 9, 7, 10)$.
  - $L(1) = 1$
  - $L(2) = 1$
  - $L(3) = 2$
  - $L(4) = 2$
  - $L(5) = 2$
  - $L(6) = 1$
  - $L(7) = \max(2, 2, 3, 3, 3, 2) = 3$
  - $L(8) = \max(2, 2, 3, 3, 2) = 3$
  - $L(9) = \max(2, 2, 3, 3, 3, 2, 4, 4) = 4$
- What is the length of the longest increasing subsequence?
  - $\max_{1 \le j \le n} L(j)$

Dynamic Programming: Examples

Length-LIS(A, n)
- if ($n = 1$) return(1)
- $max \leftarrow 1$
- for $j = (n-1)$ to 1
  - if $(A[j] \leq A[n])$
    - $s \leftarrow$Length-LIS(A, j)
    - if ($\max < s + 1$) $max \leftarrow s + 1$
- return($\max$)

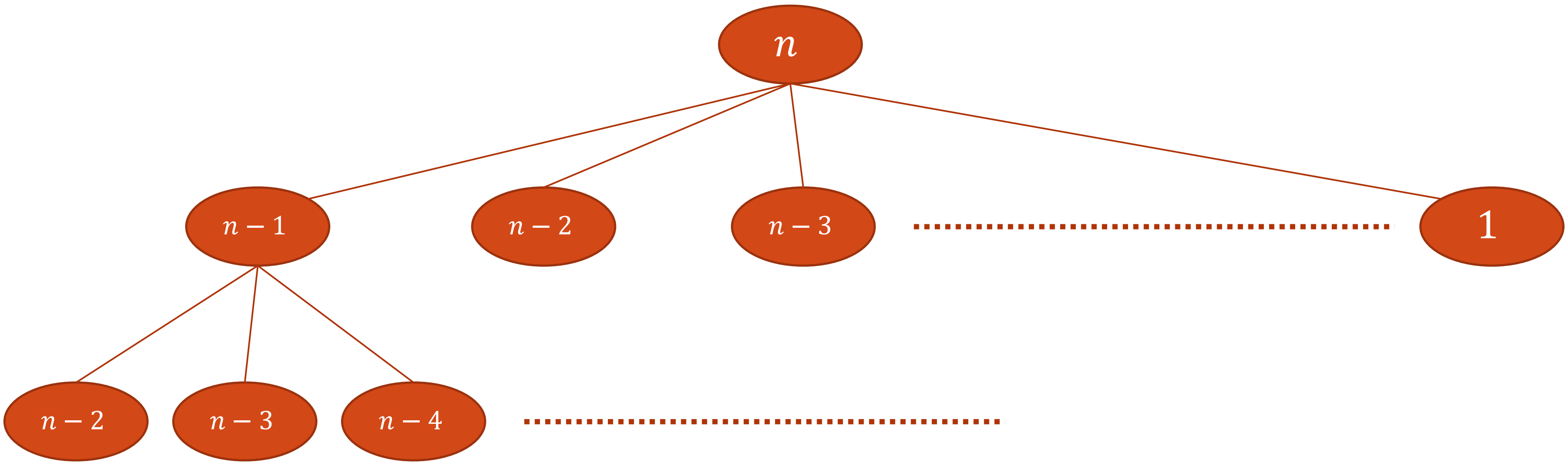
- What is the running time for the above algorithm?
  - $T(n) = T(n-1) + T(n-2) + \ldots + T(1)$

Dynamic Programming: Examples

```
Length-LIS(A, n)
    - if (n = 1) return(1)
    - max ← 1
    - for j = (n − 1) to 1
        - if (A[j] ≤ A[n])
            - s ←Length-LIS(A, j)
            - if (max < s + 1) max ← s + 1
    - return(max)
```

- What is the running time for the above algorithm?
  - $T(n) = T(n-1) + T(n-2) + \ldots + T(1)$
  - $T(n) = 2^{O(n)}$

Dynamic Programming: Examples



- Lot of nodes are repeated.

Dynamic Programming: Examples

Length-LIS(A)

- for $i = 1$ to n
  - $max \leftarrow 1$
  - for $j = 1$ to $(i - 1)$
    - if $(A[j] \leq A[i])$
      - if $(max < L[j] + 1)$
        - $max \leftarrow L[j] + 1$
  - $L[i] \leftarrow max$
- return the maximum of $L[i]$'s

- What is the running time for the above algorithm?

Dynamic Programming: Examples

Length-LIS(A)
- for $i = 1$ to n
  - $max \leftarrow 1$
  - for $j = 1$ to $(i - 1)$
    - if $(A[j] \leq A[i])$
      - if $(max < L[j] + 1)$
        - $max \leftarrow L[j] + 1$
  - $L[i] \leftarrow max$
- return the maximum of $L[i]$'s

- What is the running time for the above algorithm?
  - $T(n) = O(n^2)$

Dynamic Programming: Examples

Length-LIS(A)
- for $i = 1$ to n
  - $max \leftarrow 1$
  - for $j = 1$ to $(i - 1)$
    - if $(A[j] \leq A[i])$
      - if $(max < L[j] + 1)$
        - $max \leftarrow L[j] + 1$
  - $L[i] \leftarrow max$
- return the maximum of $L[i]$'s

- What is the running time for the above algorithm?
  - $T(n) = O(n^2)$
- But the problem was to find the longest increasing subsequence and not the length!

Dynamic Programming: Examples

LIS(A)

- for $i = 1$ to n
  - $max \leftarrow 1$
  - $P[i] = i$
  - for $j = 1$ to $(i - 1)$
    - if $(A[j] \leq A[i])$
      - if $(max < L[j] + 1)$
        - max $\leftarrow L[j] + 1$
        - $P[i] = j$
  - $L[i] \leftarrow max$
- *//P stores the longest increasing subsequence output this*

- But the problem was to find the longest increasing subsequence and not the length!
- For each number, we just note down the index of the number preceding this number in a longest increasing subsequence.

Dynamic Programming: Examples

	1	2	3	4	5	6	7	8	9
A	7	2	8	6	3	1	9	7	10
L	1	1	2	2	2	1	3	3	4
P	1	2	1	2	2	6	3	4	7

LIS(A)

- for $i = 1$ to n
  - $max \leftarrow 1$
  - $P[i] = i$
  - for $j = 1$ to $(i - 1)$
    - if $(A[j] \leq A[i])$
      - if $(max < L[j] + 1)$
        - $\max \leftarrow L[j] + 1$
        - $P[i] = j$
  - $L[i] = max$
- *//P stores the longest increasing subsequence output this*

Dynamic Programming: Examples

	1	2	3	4	5	6	7	8	9
A	7	2	8	6	3	1	9	7	10
L	1	1	2	2	2	1	3	3	4
P	1	2	1	2	2	6	3	4	7

- So one of the longest increasing subsequence is (7, 8, 9, 10).

Dynamic Programming: Examples

Longest Common Subsequence

Dynamic Programming: Examples

- Problem(longest common subsequence): Let S and T be strings of characters. S is of length n and T is of length m. Find the *longest common subsequence*. This is the longest sequence of characters that appear in both S and T. The characters are not necessarily contiguous.
- Example: S = XYXZPQ, T =YXQYXP.
  - The longest common subsequence is XYXP
  - S=**XYX**Z**P**Q, T=Y**X**Q**YXP**

End

Problems to think about:

1. We saw an example where there were exponentially large number of increasing subsequences such that the length of these sequences was equal to the length of the longest increasing subsequence. Can you construct a similar example for the longest common subsequence problem?