COL106: Data Structures and Algorithms

Ragesh Jaiswal, IIT Delhi

Course Overview

- Graph Algorithms
- Algorithm Design Techniques:
  - Greedy Algorithms
  - Divide and Conquer
  - Dynamic Programming
  - Network Flows
- Computational Intractability

Dynamic Programming

- Main idea: *Break the given problem in to a few sub-problems and combine the solutions of the smaller sub-problems to get solutions to larger ones.*
- How is it different than Divide and Conquer?
  - Here you are allowed overlapping sub-problems.

- Main idea: *Break the given problem in to a few sub-problems and combine the solutions of the smaller sub-problems to get solutions to larger ones.*
- How is it different than Divide and Conquer?
  - Here you are allowed 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

Problem

Longest increasing subsequence: You are given a sequence of integers $A[1], A[2], ..., A[n]$ and you are asked to find a longest increasing subsequence of integers.

- Example: The longest increasing subsequence of the sequence $(7, 2, 8, 6, 3, 6, 9, 7)$ is ?

Problem

Longest increasing subsequence: You are given a sequence of integers $A[1], A[2], ..., A[n]$ and you are asked to find a 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 $L(1)$?

Problem

Longest increasing subsequence: You are given a sequence of integers $A[1], A[2], ..., A[n]$ and you are asked to find a 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 $L(1), ...L(i-1)$?

Problem

Longest increasing subsequence: You are given a sequence of integers $A[1], A[2], ..., A[n]$ and you are asked to find a 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 $L(1), ...L(i-1)$?

$$L(i) = 1 + \max_{j<i \text{ and } A[j]<A[i]} \{L(j)\}$$

  - Note that if the set $\{j : j < i \text{ and } A[j] < A[i]\}$ is empty, then the second term on the RHS is 0.

- Let $n = 9$ and $(A[1], ..., A[9]) = (7, 2, 8, 6, 3, 1, 10, 9, 11)$
  - $L(1) =?$
  - $L(2) =?$
  - $L(3) =?$
  - $L(4) =?$
  - $L(5) =?$
  - $L(6) =?$
  - $L(7) =?$
  - $L(8) =?$
  - $L(9) =?$

- Let $n = 9$ and $(A[1], ..., A[9]) = (7, 2, 8, 6, 3, 1, 10, 9, 11)$
  - $L(1) = 1$
  - $L(2) = 1$
  - $L(3) = 2$
  - $L(4) = 2$
  - $L(5) = 2$
  - $L(6) = 1$
  - $L(7) = 1 + \max\{1, 1, 2, 2, 2, 1\} = 3$
  - $L(8) = 1 + \max\{1, 1, 2, 2, 2, 1\} = 3$
  - $L(9) = 1 + \max\{1, 1, 2, 2, 2, 1, 3, 3\} = 4$
- What is the length of the longest increasing subsequence?

- Let $n = 9$ and $(A[1], ..., A[9]) = (7, 2, 8, 6, 3, 1, 10, 9, 11)$
  - $L(1) = 1$
  - $L(2) = 1$
  - $L(3) = 2$
  - $L(4) = 2$
  - $L(5) = 2$
  - $L(6) = 1$
  - $L(7) = 1 + \max\{1, 1, 2, 2, 2, 1\} = 3$
  - $L(8) = 1 + \max\{1, 1, 2, 2, 2, 1\} = 3$
  - $L(9) = 1 + \max\{1, 1, 2, 2, 2, 1, 3, 3\} = 4$
- What is the length of the longest increasing subsequence?

$$\max_{1 \leq j \leq n} L(j)$$

Algorithm

`Length-LIS-recursive`(A, n)
- If $(n = 1)$ return(1)
- $max \leftarrow 1$
- For $j = (n-1)$ to 1
  - If $(A[j] < A[n])$
    - $s \leftarrow$ `Length-LIS-recursive`(A, j)
    - If $(max < s + 1)$ $max \leftarrow s + 1$
- return(max)

- What is the running time of this algorithm?

Algorithm

`Length-LIS-recursive`(A, n)
- If $(n = 1)$ return(1)
- $max \leftarrow 1$
- For $j = (n-1)$ to 1
  - If $(A[j] < A[n])$
    - $s \leftarrow$ `Length-LIS-recursive`(A, j)
    - If $(max < s+1)$ $max \leftarrow s+1$
- return(max)

- What is the running time of this algorithm?
  - $T(n) \leq T(n-1) + T(n-2) + ... + T(1) + cn$; $T(1) \leq c$

Algorithm

`Length-LIS-recursive`(A, n)

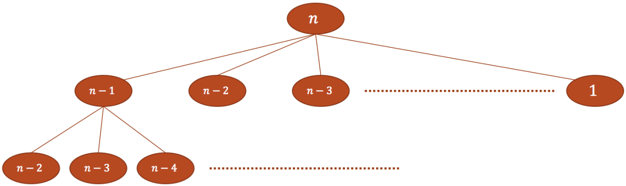
- If $(n = 1)$ return(1)
- $max \leftarrow 1$
- For $j = (n-1)$ to 1
  - If $(A[j] < A[n])$
    - $s \leftarrow$ `Length-LIS-recursive`(A, j)
    - If $(max < s + 1)$ $max \leftarrow s + 1$
- return(max)

- What is the running time of this algorithm?
  - $T(n) \leq T(n-1) + T(n-2) + ... + T(1) + cn$; $T(1) \leq c$
  - $T(n) = O(n \cdot 2^n)$

Algorithm

Length-LIS-recursive(A, n)
- If $(n = 1)$ return(1)
- $max \leftarrow 1$
- For $j = (n - 1)$ to 1
  - If $(A[j] < A[n])$
    - $s \leftarrow$ Length-LIS-recursive(A, j)
    - If $(max < s + 1)$ $max \leftarrow s + 1$
- return(max)

- What is the running time of this algorithm?
  - $T(n) =\leq T(n-1) + T(n-2) + ... + T(1) + cn; \; T(1) \leq c$
  - $T(n) = O(n \cdot 2^n)$
- Lot of nodes in the recursion tree are repeated.



Algorithm

`Length-LIS`(A, n)
- For $i = 1$ to n
  - $max \leftarrow 1$
  - For $j = 1$ to $(i - 1)$
    - If $(A[j] < A[i])$
      - 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 of this algorithm?

Algorithm

`Length-LIS`(A, n)
- For $i = 1$ to n
  - $max \leftarrow 1$
  - For $j = 1$ to $(i - 1)$
    - If $(A[j] < A[i])$
      - 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 of this algorithm? $O(n^2)$
- But the problem was to find a longest increasing subsequence and not the length!

Algorithm

LIS(A, n)
- For $i = 1$ to n
  - $max \leftarrow 1$
  - $P[i] \leftarrow i$
  - For $j = 1$ to $(i - 1)$
    - If $(A[j] < A[i])$
      - If $(max < L[j] + 1)$
        - $max \leftarrow L[j] + 1$
        - $P[i] \leftarrow j$
  - $L[i] \leftarrow max$
- ... // *Use P to output a longest increasing subsequence*

- But the problem was to find a 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.

	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

Algorithm

LIS(A, n)

- For $i = 1$ to n
  - $max \leftarrow 1$
  - $P[i] \leftarrow i$
  - For $j = 1$ to $(i - 1)$
    - If $(A[j] < A[i])$
      - If $(max < L[j] + 1)$
        - $max \leftarrow L[j] + 1$
        - $P[i] \leftarrow j$
  - $L[i] \leftarrow max$
- OutputSequence(A, L, P, n)

Dynamic Programming
Longest increasing subsequence

	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

Algorithm

OutputSequence(A, L, P, n)
- Let j be the index such that $L[j]$ is maximized
- $i \leftarrow 1$
- While $(P[j] \neq j)$
  - $B[i] \leftarrow A[j]$
  - $j \leftarrow P[j]$
  - $i \leftarrow i + 1$
- $B[i] \leftarrow A[j]$
- Print B in reverse order

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

Problem

Let S and T be strings of characters. S is of length n and T is of length m. Find a *longest common subsequence* in S and T. This is a longest sequence of characters (not necessarily contiguous) that appear in both S and T.

- Example $S = \text{XYXZPQ}$, $T = \text{YXQYXP}$

Problem

Let S and T be strings of characters. S is of length n and T is of length m. Find a *longest common subsequence* in S and T. This is a longest sequence of characters (not necessarily contiguous) that appear in both S and T.

- Example S = XYXZPQ, T = YXQYXP
  - The longest common subsequence is XYXP
  - S = XYXZPQ, T = YXQYXP
- How do we define the subproblems?

End