Dynamic Programming

Hengfeng Wei

hfwei@nju.edu.cn

June 11 – June 19, 2017

Dynamic Programming

1. 1D DP
2. 2D DP
3. DP on Graphs
4. The Knapsack Problem

$f^{(S(n))} = 1$

$f^{(S(n))} = 1$ (Problem 7.2)

$$f(n) = \begin{cases} n - 1 & \text{if } n \in \mathbb{Z}^+ \\ n/2 & \text{if } n\%2 = 0 \\ n/3 & \text{if } n\%3 = 0 \end{cases}$$

$S(n)$: minimum number of steps taking n to 1.

$f^{(S(n))} = 1$

$f^{(S(n))} = 1$ (Problem 7.2)

$$f(n) = \begin{cases} n - 1 & \text{if } n \in \mathbb{Z}^+ \\ n/2 & \text{if } n\%2 = 0 \\ n/3 & \text{if } n\%3 = 0 \end{cases}$$

$S(n)$: minimum number of steps taking n to 1.

$S(i)$: minimum number of steps taking i to 1

$f^{(S(n))} = 1$

$f^{(S(n))} = 1$ (Problem 7.2)

$$f(n) = \begin{cases} n - 1 & \text{if } n \in \mathbb{Z}^+ \\ n/2 & \text{if } n\%2 = 0 \\ n/3 & \text{if } n\%3 = 0 \end{cases}$$

$S(n)$: minimum number of steps taking n to 1.

$S(i)$: minimum number of steps taking i to 1

$$S(i) = 1 + \min\{N(i-1), N(i/2)(\text{if } n\%2 = 0), N(i/3)(\text{if } n\%3 = 0)\}$$

$$S(1) = 0$$

$f^{(S(n))} = 1$

Collatz $(3n+1)$ conjecture:

$$f(n) = \begin{cases} n/2 & \text{if } n\%2 = 0 \\ 3n+1 & \text{if } n\%2 = 1 \end{cases}$$

$$f^*(n) = 1?$$

$f^{(S(n))} = 1$

Collatz $(3n+1)$ conjecture:

$$f(n) = \begin{cases} n/2 & \text{if } n\%2 = 0 \\ 3n+1 & \text{if } n\%2 = 1 \end{cases}$$

$$f^*(n) = 1?$$

"Mathematics may not be ready for such problems."

— Paul Erdős

Longest increasing subsequence

Longest increasing subsequence (Problem 7.3)

- Given an integer array $A[1 \dots n]$
- To find (the length of) a longest increasing subseqence.

$$5, 2, 8, 6, 3, 6, 9, 7 \implies 2, 3, 6, 9$$

Longest increasing subsequence

Subproblem: $L(i)$: the length of the LIS of $A[1 \ldots i]$

Goal: $L(n)$

Longest increasing subsequence

Subproblem: $L(i)$: the length of the LIS of $A[1 \dots i]$

Goal: $L(n)$

Make choice: whether $A[i] \in LIS[1 \dots i]$?

Recurrence:

$$L(i) = \max\{L(i-1), 1 + \max_{j < i \wedge A[j] \leq A[i]} L(j)\}$$

Longest increasing subsequence

Subproblem: $L(i)$: the length of the LIS of $A[1 \dots i]$

Goal: $L(n)$

Make choice: whether $A[i] \in LIS[1 \dots i]$?

Recurrence:

$$L(i) = \max\{L(i-1), 1 + \max_{j<i \wedge A[j] \le A[i]} L(j)\}$$

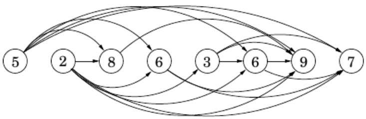
Init:

$$L(0) = 0$$

Time:

$$O(n^2) = \Theta(n) \cdot O(n)$$

Longest increasing subsequence



Longest path distance in the DAG!

Maximum-sum subarray

Maximum-sum subarray (Google Interview)

- Array $A[1 \cdots n], a_i >=< 0$
- To find (the sum of) a maximum-sum subarray of A
  - mss $= 0$ if all negative

$$A[-2, 1, -3, 4, -1, 2, 1, -5, 4] \implies [4, -1, 2, 1]$$

Maximum-sum subarray

Maximum-sum subarray (Google Interview)

- Array $A[1 \cdots n], a_i >=< 0$
- To find (the sum of) a maximum-sum subarray of A
  - mss $= 0$ if all negative

$$A[-2, 1, -3, 4, -1, 2, 1, -5, 4] \implies [4, -1, 2, 1]$$

Subproblem: MSS$[i]$: the sum of the MS$[i]$ of $A[1 \cdots i]$

Goal: mss $=$ MSS$[n]$

Maximum-sum subarray

Maximum-sum subarray (Google Interview)

- Array $A[1 \cdots n], a_i >=< 0$
- To find (the sum of) a maximum-sum subarray of A
  - mss $= 0$ if all negative

$$A[-2, 1, -3, 4, -1, 2, 1, -5, 4] \implies [4, -1, 2, 1]$$

Subproblem: MSS$[i]$: the sum of the MS$[i]$ of $A[1 \cdots i]$

Goal: mss $=$ MSS$[n]$

Make choice: Is $a_i \in$ MS$[i]$?

Recurrence:

$$\text{MSS}[i] = \max\{\text{MSS}[i-1], ???\}$$

Maximum-sum subarray

Subproblem: MSS$[i]$: the sum of the MS$[i]$ *ending with* a_i or 0

Goal: mss $= \max_{1 \le i \le n}$ MSS$[i]$

Maximum-sum subarray

Subproblem: MSS$[i]$: the sum of the MS$[i]$ *ending with* a_i or 0

Goal: mss $= \max_{1 \le i \le n}$ MSS$[i]$

Make choice: where does the MS$[i]$ start?

Recurrence:

$$\text{MSS}[i] = \max\{\text{MSS}[i-1] + a_i, 0\}$$ (prove it!)

Maximum-sum subarray

Subproblem: MSS$[i]$: the sum of the MS$[i]$ *ending with* a_i or 0

Goal: mss $= \max_{1 \le i \le n}$ MSS$[i]$

Make choice: where does the MS$[i]$ start?

Recurrence:

$$\text{MSS}[i] = \max\{\text{MSS}[i-1] + a_i, 0\}$$ (prove it!)

Init:

$$\text{MSS}[0] = 0$$

Maximum-sum subarray

Subproblem: MSS$[i]$: the sum of the MS$[i]$ *ending with* a_i or 0

Goal: mss $= \max_{1 \le i \le n}$ MSS$[i]$

Make choice: where does the MS$[i]$ start?

Recurrence:

$$\text{MSS}[i] = \max\{\text{MSS}[i-1] + a_i, 0\}$$ (prove it!)

Init:

$$\text{MSS}[0] = 0$$

Time: $\Theta(n)$

Maximum-sum subarray

Maximum-product subarray

Maximum-product subarray (Problem 7.4)

Reconstructing string

Reconstructing string (Problem 7.9)

- String $S[1 \cdots n]$
- Dict for *lookup*:

$$\text{dict}(w) = \begin{cases} \text{true} & \text{if } w \text{ is a valid word} \\ \text{false} & \text{o.w.} \end{cases}$$

- Is $S[1 \cdots n]$ valid (reconstructed as a sequence of valid words)?

Reconstructing string

Subproblem: $V[i]$: is $S[1 \cdots i]$ valid?

Goal: $V[n]$

Reconstructing string

Subproblem: $V[i]$: is $S[1 \cdots i]$ valid?

Goal: $V[n]$

Make choice: where does the last word start?

Recurrence:

$$V[i] = \bigvee_{j=1\ldots i} (V[j-1] \land \mathsf{dict}(S[j \cdots i]))$$

Reconstructing string

Subproblem: $V[i]$: is $S[1 \cdots i]$ valid?

Goal: $V[n]$

Make choice: where does the last word start?

Recurrence:

$$V[i] = \bigvee_{j=1\ldots i} (V[j-1] \wedge \mathsf{dict}(S[j \cdots i]))$$

Init:

$$V[0] = \mathsf{true}$$

Time: $O(n^2)$

Hotel along a trip

Hotel along a trip (Problem 7.15)

- Hotel sequence (distance): $a_0 = 0, a_1, \cdots, a_n$
- $a_0 \rightsquigarrow a_n$
- Stop at only hotels
- Cost: $(200 - x)^2$
- To minimize overall cost

Hotel along a trip

Subproblem: $C[i]$: minimum cost when the destination is a_i

Goal: $C[n]$

Hotel along a trip

Subproblem: $C[i]$: minimum cost when the destination is a_i

Goal: $C[n]$

Make choice: what is the last but one hotel a_j to stop?

Recurrence:

$$C[i] = \min_{0 \le j < i} \{C[j] + (200 - (a_i - a_j))^2\}$$

Hotel along a trip

Subproblem: $C[i]$: minimum cost when the destination is a_i

Goal: $C[n]$

Make choice: what is the last but one hotel a_j to stop?

Recurrence:

$$C[i] = \min_{0 \le j < i} \{C[j] + (200 - (a_i - a_j))^2\}$$

Init:

$$C[0] = 0$$

Time:

$$O(n^2) = \Theta(n) \cdot O(n)$$

Highway restaurants

Highway restaurants (Problem 7.16)

- Locations: $L[1 \ldots n]$
- Profits: $P[1 \ldots n]$
- Any two hotels should be $\geq k$ miles apart
- To maximize the total profit

Subproblem:

Goal:

Make choice:

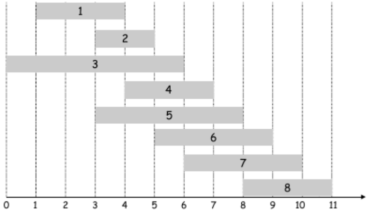
Recurrence:

Time:

Weighted interval/class scheduling

Weighted interval/class scheduling (Problem 7.14)

- Classes: $\mathcal{C} = \{c_1, c_2, \cdots, c_n\} \quad c_i \triangleq \langle g_i, s_i, f_i \rangle$
- Choosing non-conflicting classes to maximize your grades



sort $\mathcal{C}$ by finishing time.

Weighted interval/class scheduling

Greedy algorithms by finishing time or weights fail.

Weighted interval/class scheduling

Subproblem: $G[i]$: the maximal grades obtained from $\{c_1, c_2, \cdots, c_i\}$

Goal: $G[n]$

Weighted interval/class scheduling

Subproblem: $G[i]$: the maximal grades obtained from $\{c_1, c_2, \cdots, c_i\}$

Goal: $G[n]$

Make choice: choose c_i or not in $G[i]$?

Recurrence:

$$G[i] = \max\{G[i-1], G[p(i)] + g_i\}$$

$p(i)$: the largest index $j < i$ *s.t.* c_i and c_j are disjoint

Weighted interval/class scheduling

Subproblem: $G[i]$: the maximal grades obtained from $\{c_1, c_2, \cdots, c_i\}$

Goal: $G[n]$

Make choice: choose c_i or not in $G[i]$?

Recurrence:

$$G[i] = \max\{G[i-1], G[p(i)] + g_i\}$$

$p(i)$: the largest index $j < i$ *s.t.* c_i and c_j are disjoint

Init:

$$G[0] = 0$$

Weighted interval/class scheduling

Subproblem: $G[i]$: the maximal grades obtained from $\{c_1, c_2, \cdots, c_i\}$

Goal: $G[n]$

Make choice: choose c_i or not in $G[i]$?

Recurrence:

$$G[i] = \max\{G[i-1], G[p(i)] + g_i\}$$

$p(i)$: the largest index $j < i$ *s.t.* c_i and c_j are disjoint

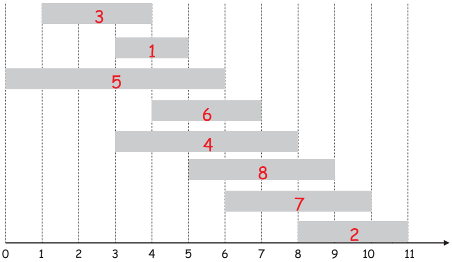
Init:

$$G[0] = 0$$

Time: $O(n \log n) + T(p(i)) + O(n) \cdot O(1)$

Weighted interval/class scheduling

Why is ordering necessary?

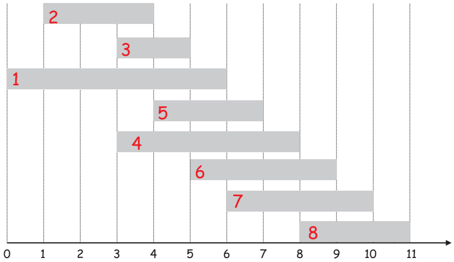


$$G[7] = \max\{G[6], G[\{1,3,5\}] + g_7\}$$

subproblems changed: all $O(2^n)$ subsets

Weighted interval/class scheduling

What about sorting by starting time?



$$G[6] = \max\{G[5], G[\{2,3\}] + g_6\}$$

subproblems changed: all $O(2^n)$ subsets

Dynamic Programming

1. 1D DP
2. 2D DP
3. DP on Graphs
4. The Knapsack Problem

Longest common subsequence

LCS: longest common subsequence (Problem 7.5)

$$X = X_1 \cdots X_m \quad Y = Y_1 \cdots Y_n$$

(1) Find (the length of) an LCS of X and Y

$$X = \langle A, B, C, B, D, A, B \rangle$$
$$Y = \langle B, D, C, A, B, A \rangle$$
$$Z = \langle B, C, B, A \rangle$$

Longest common subsequence

Subproblem: $L[i, j]$: the length of an LCS of $X[1 \cdots i]$ and $Y[1 \cdots j]$

Goal: $L[m, n]$

Longest common subsequence

Subproblem: $L[i, j]$: the length of an LCS of $X[1 \cdots i]$ and $Y[1 \cdots j]$

Goal: $L[m, n]$

Make choice: Is $X_i = Y_j$?

Recurrence: (Proof!)

$$L[i,j] = \begin{cases} L[i-1, j-1] + 1 & \text{if } X_i = Y_j \\ \max\{L[i-1, j], L[i, j-1]\} & \text{if } X_i \neq Y_j \end{cases}$$

Longest common subsequence

Subproblem: $L[i, j]$: the length of an LCS of $X[1 \cdots i]$ and $Y[1 \cdots j]$

Goal: $L[m, n]$

Make choice: Is $X_i = Y_j$?

Recurrence: (Proof!)

$$L[i, j] = \begin{cases} L[i-1, j-1] + 1 & \text{if } X_i = Y_j \\ \max\{L[i-1, j], L[i, j-1]\} & \text{if } X_i \neq Y_j \end{cases}$$

Init:

$$L[0, j] = 0, \; 0 \leq j \leq n$$
$$L[i, 0] = 0, \; 0 \leq i \leq m$$

Time: $\Theta(mn)$

Longest common subsequence

Longest common subsequence (Problem 7.5)

$$X = X_1 \cdots X_m \quad Y = Y_1 \cdots Y_n$$

(2) Allowing repetition of X

(3) Allowing repetition $\leq k$ of X

Longest common subsequence

Longest common subsequence (Problem 7.5)

$$X = X_1 \cdots X_m \quad Y = Y_1 \cdots Y_n$$

(2) Allowing repetition of X

(3) Allowing repetition $\leq k$ of X

$$L[i,j] = \begin{cases} L[i, j-1] + 1 & \text{if } X_i = Y_j \\ \max\{L[i-1,j], L[i,j-1]\} & \text{if } X_i \neq Y_j \end{cases}$$

Longest common subsequence

Longest common subsequence (Problem 7.5)

$$X = X_1 \cdots X_m \quad Y = Y_1 \cdots Y_n$$

(2) Allowing repetition of X

(3) Allowing repetition $\leq k$ of X

$$L[i,j] = \begin{cases} L[i, j-1] + 1 & \text{if } X_i = Y_j \\ \max\{L[i-1,j], L[i,j-1]\} & \text{if } X_i \neq Y_j \end{cases}$$

$$X \implies X^{(k)} \triangleq X_1^{(k)} \cdots X_m^{(k)}$$

Longest common substring

What about longest common *substring*?

Shortest common supersequence

Shortest common supersequence (Problem 7.6)

$$X = X_1 \cdots X_m \quad Y = Y_1 \cdots Y_n$$

- Find (the length of) a shortest common subsequence of X and Y

Shortest common supersequence

Subproblem: $L[i,j]$: the length of an SCS of $X[1 \cdots i]$ and $Y[1 \cdots j]$

Goal: $L[m,n]$

Shortest common supersequence

Subproblem: $L[i,j]$: the length of an SCS of $X[1 \cdots i]$ and $Y[1 \cdots j]$

Goal: $L[m,n]$

Make choice: Is $X_i = Y_j$?

Recurrence:

$$L[i,j] = \begin{cases} L[i-1,j-1]+1 & \text{if } X_i = Y_j \\ \max\{L[i-1,j]+1, L[i,j-1]+1\} & \text{if } X_i \neq Y_j \end{cases}$$

Shortest common supersequence

Subproblem: $L[i,j]$: the length of an SCS of $X[1 \cdots i]$ and $Y[1 \cdots j]$

Goal: $L[m,n]$

Make choice: Is $X_i = Y_j$?

Recurrence:

$$L[i,j] = \begin{cases} L[i-1,j-1]+1 & \text{if } X_i = Y_j \\ \max\{L[i-1,j]+1, L[i,j-1]+1\} & \text{if } X_i \neq Y_j \end{cases}$$

Init:

$$L[0,j] = j, \ 0 \le j \le n$$
$$L[i,0] = i, \ 0 \le i \le m$$

Shortest common supersequence

Subproblem: $L[i, j]$: the length of an SCS of $X[1 \cdots i]$ and $Y[1 \cdots j]$

Goal: $L[m, n]$

Make choice: Is $X_i = Y_j$?

Recurrence:

$$L[i, j] = \begin{cases} L[i-1, j-1] + 1 & \text{if } X_i = Y_j \\ \max\{L[i-1, j] + 1, L[i, j-1] + 1\} & \text{if } X_i \neq Y_j \end{cases}$$

Init:

$$L[0, j] = j, \ 0 \leq j \leq n$$
$$L[i, 0] = i, \ 0 \leq i \leq m$$

Remark

$$\max(m, n) \leq L(m, n) \leq m + n$$

Variants of LCS

Variants of LCS (Problem 7.7)

Longest contiguous substring both forward and backward

Longest contiguous substring both forward and backward (Problem 7.8)

- String $T[1 \cdots n]$
- Find a longest contiguous substring (LCS) both forward and backward

dynamicprogrammingmanytimes

- try subproblem $L[i]$: the length of an LCS in $T[1 \cdots i]$
- try subproblem $L[i, j]$: the length of an LCS in $T[i \cdots j]$

Longest contiguous substring both forward and backward

Subproblem: $L[i, j]$: the length of an LCS starting with T_i and ending with T_j

Goal: $\max_{1 \leq i \leq j \leq n} L[i, j]$

Longest contiguous substring both forward and backward

Subproblem: $L[i, j]$: the length of an LCS starting with T_i and ending with T_j

Goal: $\max_{1 \le i \le j \le n} L[i, j]$

Make choice: Is $T_i = T_j$?

Recurrence:

$$L[i, j] = \begin{cases} 0 & \text{if } T_i \neq T_j \\ L[i+1, j-1] + 1 & \text{if } T_i = T_j \end{cases}$$

Longest contiguous substring both forward and backward

Subproblem: $L[i, j]$: the length of an LCS starting with T_i and ending with T_j

Goal: $\max_{1 \leq i \leq j \leq n} L[i, j]$

Make choice: Is $T_i = T_j$?

Recurrence:

$$L[i, j] = \begin{cases} 0 & \text{if } T_i \neq T_j \\ L[i+1, j-1] + 1 & \text{if } T_i = T_j \end{cases}$$

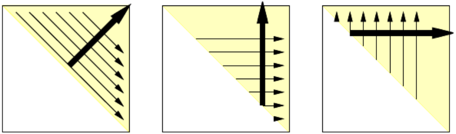
Init:

$$L[i, i] = 0, \ 0 \leq i \leq n$$

$$L[i, i+1] = \begin{cases} 1 & \text{if } T_i = T_{i+1} \\ 0 & \text{if } T_i \neq T_{i+1} \end{cases}$$

Longest contiguous substring both forward and backward

Code: three ways of filling the table



```
for d = 2 to n-1
  for i = 1 to n-d
    j = i + d
    ...
return max_{1 <= i <= j <= n} L[i,j]
```

Longest palindrome subsequence

Longest palindrome subsequence (Problem 7.10)

(1) Find (the length of) a longest palindrome subsequence of $S[1 \cdots n]$

Subproblem: $L[i, j]$: the length of an LSP of $S[i \cdots j]$

Goal: $L[1, n]$

Longest palindrome subsequence

Longest palindrome subsequence (Problem 7.10)

(1) Find (the length of) a longest palindrome subsequence of $S[1 \cdots n]$

Subproblem: $L[i,j]$: the length of an LSP of $S[i \cdots j]$

Goal: $L[1,n]$

Make choice: Is $S[i] = S[j]$?

Recurrence:

$$L[i,j] = \begin{cases} L[i+1, j-1] + 2 & \text{if } S[i] = S[j] \\ \max L[i+1, j], L[i, j-1] & \text{if } S[i] \neq S[j] \end{cases}$$

Longest palindrome subsequence

Longest palindrome subsequence (Problem 7.10)

(1) Find (the length of) a longest palindrome subsequence of $S[1 \cdots n]$

Subproblem: $L[i,j]$: the length of an LSP of $S[i \cdots j]$

Goal: $L[1,n]$

Make choice: Is $S[i] = S[j]$?

Recurrence:

$$L[i,j] = \begin{cases} L[i+1, j-1] + 2 & \text{if } S[i] = S[j] \\ \max L[i+1,j], L[i,j-1] & \text{if } S[i] \neq S[j] \end{cases}$$

Init:

$$L[i,i] = 1, \; \forall 1 \leq i \leq n$$

Palindrome splitting

Palindrome splitting (Problem 7.10)

(2) Split a string $S[1 \dots n]$ into minimum number of palindromes ($\#$ cuts)

Subproblem: $C[i, j]$: minimum number of cuts for string $S[i \dots j]$

Goal: $C[1, n] + 1$

Palindrome splitting

Palindrome splitting (Problem 7.10)

(2) Split a string $S[1 \ldots n]$ into minimum number of palindromes ($\#$ cuts)

Subproblem: $C[i, j]$: minimum number of cuts for string $S[i \ldots j]$

Goal: $C[1, n] + 1$

Make choice: Where is the first cut?

Recurrence:

$$C[i, j] = \begin{cases} 0 & \text{if } S[i \ldots j] \text{ is a palindrome} \\ \min\limits_{i+1 \le k \le j-1} C[i, k] + 1 + C[k+1, j] & \text{o.w.} \end{cases}$$

Palindrome splitting

Palindrome splitting (Problem 7.10)

(2) Split a string $S[1 \dots n]$ into minimum number of palindromes ($\#$ cuts)

Subproblem: $C[i,j]$: minimum number of cuts for string $S[i \dots j]$

Goal: $C[1,n] + 1$

Make choice: Where is the first cut?

Recurrence:

$$C[i,j] = \begin{cases} 0 & \text{if } S[i \dots j] \text{ is a palindrome} \\ \min\limits_{i+1 \le k \le j-1} C[i,k] + 1 + C[k+1,j] & \text{o.w.} \end{cases}$$

Init: $C[i,i] = 0$

Time: $O(n^3)$

Palindrome splitting

Palindrome splitting (Problem 7.10)

(2) Split a string $S[1 \dots n]$ into minimum number of palindromes

Subproblem: $P[i]$: minimum number of palindromes for $S[1 \cdots i]$

Goal: $P[n]$

Palindrome splitting

Palindrome splitting (Problem 7.10)

(2) Split a string $S[1 \dots n]$ into minimum number of palindromes

Subproblem: $P[i]$: minimum number of palindromes for $S[1 \cdots i]$

Goal: $P[n]$

Make choice: Where does the last palindrome start from?

Recurrence:

$$P[i] = \min_{\substack{1 \le k \le i \\ S[k \dots i] \text{ is a palindrome}}} P[k-1] + 1$$

Palindrome splitting

Palindrome splitting (Problem 7.10)

(2) Split a string $S[1 \dots n]$ into minimum number of palindromes

Subproblem: $P[i]$: minimum number of palindromes for $S[1 \cdots i]$

Goal: $P[n]$

Make choice: Where does the last palindrome start from?

Recurrence:

$$P[i] = \min_{\substack{1 \le k \le i \\ S[k \dots i] \text{ is a palindrome}}} P[k-1] + 1$$

Init: $P[0] = 1$

Time: $O(n^2)$

String splitting

String splitting (Problem 7.11)

- Split a string S into many pieces
- Cost $|S| = n \implies n$
- Given locations of m cuts: $C_0, C_1, \cdots, C_m, C_{m+1}$
- Find the minimum cost of splitting S into $m+1$ pieces $S_0 \cdots S_m$

String splitting

Subproblem: $C[i, j]$: the minimum cost of splitting substring $S_i \cdots S_{j-1}$ using cuts $C_{i+1} \cdots C_{j-1}$

Goal: $C[0, m+1]$

String splitting

Subproblem: $C[i,j]$: the minimum cost of splitting substring $S_i \cdots S_{j-1}$ using cuts $C_{i+1} \cdots C_{j-1}$

Goal: $C[0, m+1]$

Make choice: What is the first cut in $C_{i+1} \cdots C_{j-1}$?

Recurrence:

$$C[i,j] = \min_{i<k<j} \left(C[i,k] + C[k,j] + l(S_i \cdots S_{j-1}) \right)$$

String splitting

Subproblem: $C[i,j]$: the minimum cost of splitting substring $S_i \cdots S_{j-1}$ using cuts $C_{i+1} \cdots C_{j-1}$

Goal: $C[0, m+1]$

Make choice: What is the first cut in $C_{i+1} \cdots C_{j-1}$?

Recurrence:

$$C[i,j] = \min_{i<k<j} \left(C[i,k] + C[k,j] + l(S_i \cdots S_{j-1}) \right)$$

Init: $C[i, i+1] = 0$

Subproblem:

Subproblem:

Goal:

Make choice:

Subproblem:

Goal:

Make choice:

Recurrence:

Init:

Time:

Dynamic Programming

1. 1D DP
2. 2D DP
3. DP on Graphs
4. The Knapsack Problem

Dynamic Programming

1. 1D DP
2. 2D DP
3. DP on Graphs
4. The Knapsack Problem


THANK YOU