Dynamic Programming

Hengfeng Wei

Institute of Computer Software, NJU

December 25, 2013

Outline

Outline

Overview

Dynamic programming is *not* about filling in tables!
It is about smart recursion.
Smart recursion is about the *structure of subproblems*.

Common Subproblems

- 1-dimension subproblems
  - input: string (or array) $x[1 \ldots n]$;
    subproblem: prefix ($x[1 \ldots i]$), suffix ($x[i \ldots n]$);
    examples: Maximum Subarray Sum, Typesetting Problem, Longest Increasing Subsequence, Hotel Placement
- 2-dimension subproblems
  - input: two strings $x[1 \ldots m], y[1 \ldots n]$;
    subproblem: prefixes $x[1 \ldots i], y[1 \ldots j]$;
    examples: Edit Distance, Longest Common Subsequence
  - input: a string $x[1 \ldots n]$;
    subproblem: interval $x[x \ldots j]$;
    examples: Chain Matrix Multiplication, Optimal BST, Longest Palindrome

Common Subproblems

- 3-dimension subproblems
  - Floyd-Warshall algorithm
- tree-like subproblems
  - input: tree; subproblem: rooted subtree; example: Independent Set in Tree
- two mind games
  - Google Egg Game, Hungry-Lion Game

Common Subproblems

How to identify subproblems?

Make your choice:

- binary choice (whether)
  - coin-changing, ...
- multi-way choices (where, which ...)
  - optimal BST, ...

Outline

Outline

Longest Increasing Subsequence

Problem (Longest Increasing Subsequence (LIS))

- *given an integer array* $A[1 \dots n]$
- *to find (the length of) a longest increasing subseq.*

$$A : 5, 2, 8, 6, 3, 6, 7, 9$$

$$IS : 5, 8, 9;\ 2, 3, 6, 9$$

Longest Increasing Subsequence

$$\boxed{L(i) : \text{the length of the LIS of } A[1 \dots i].}$$

Theorem (Recurrence)

$\boxed{\textit{Make choice:}}$ *whether* $A[i] \in LIS[1 \dots i]$.

$$L(i) = \max \begin{cases} L(i-1) & \textit{if } A[i] \notin LIS[1 \dots i] \\ 1 + \max\{L(j) : j < i \wedge A[j] < A[i]\} & o.w. \end{cases}$$

$\boxed{\text{Base cases:}}$ $L(0) = 0$.

Longest Increasing Subsequence

Filling the table:

X :								
	5	2	8	6	3	6	7	9

Time complexity: $\Theta(n^2)$
Space complexity: $\Theta(n)$

Outline

Hotel Placement

Problem (Hotel Placement)

- *open hotels along a highway*
- *possible locations:* $X[1 \dots n] = x_1, \dots, x_n$
- *profit:* $P[1 \dots n] = p_1, \dots, p_n$
- *any two hotels should be at least k miles apart*
- $\boxed{\textit{Goal:}}$ *to maximize the total profit*

Hotel Placement

$M(i)$: max profit when considering only the first i locations.

Theorem (Recurrence)

Make choice: *whether to build a hotel at location i.*

$$M(i) = \max \begin{cases} M(i-1) & \textit{do not build at } i \\ p_i + \max\{M(j) : j < i \wedge x_i - x_j \geq k\} & o.w. \end{cases}$$

Base cases: $M(0) = 0$

Time complexity: $\Theta(n^2)$
Space complexity: $\Theta(n)$

Outline

Typesetting Problem

Problem (Typesetting Problem)

- *text: n words of widths* $W : w_1, \ldots, w_n$
- *line width* L
- *penalty:* $f(w_i \ldots w_j)$
- $\boxed{\textit{Goal:}}$ *to minimize the typesetting penalty*

Typesetting Problem

$P(i)$: the minimum penalty to typeset $W[i \dots n]$.

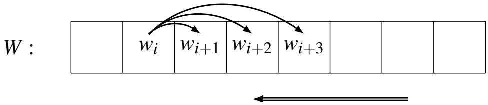
Theorem (Recurrence)

Make choice: *how many words to put in the first line.*

$$P(i) = \begin{cases} 0 & \textit{if } \sum(w_i, \dots, w_n) \leq \\ \min_{w_i + \dots w_{i+k-1} \leq L} f(w_i \dots w_k) + P(i+k) & o.w. \end{cases}$$

Typesetting Problem

Filling the table:



Time complexity: $\Theta(nW)$
Space complexity: $\Theta(n)$

Outline

Outline

Edit Distance

Problem (Edit Distance)

- *transform string $A[1 \ldots m]$ to another one $B[1 \ldots n]$*
- *allowed edits: insertion, deletion, substitution*

S	-	N	O	W	Y		-	S	N	O	W	-	Y
S	U	N	N	-	Y		S	U	N	-	-	N	Y

- $\boxed{\textit{Goal:}}$ *to find the edit distance (similarity) — the minimum number of edits*

Edit Distance

$E(i, j)$: the edit distance of $A[1 \dots i]$ and $B[1 \dots j]$.

Theorem (Recurrence)

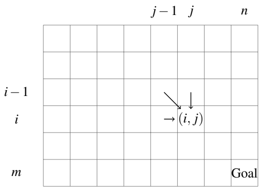
Make choices: *three cases for the rightmost column*

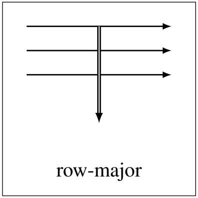
$$E(i,j) = \min \begin{cases} E(i-1,j)+1 & (x[i], \_) \\ E(i,j-1)+1 & (\_, y[j]) \\ E(i-1,j-1) & (x[i], y[j]) \textit{ and } A[i] = B[j] \\ E(i-1,j-1)+1 & (x[i], y[j]) \textit{ and } A[i] \neq B[j] \end{cases}$$

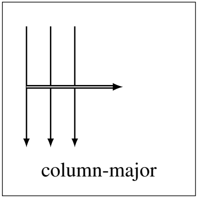
Base cases: $E(i, 0) = i$ (*deletion*); $E(0, j) = j$ (*insertion*).

Edit Distance

Filling the table:







Time complexity: $\Theta(nm)$
Space complexity: $\Theta(nm)$

Outline

Longest Common Subsequence

Problem (Longest Common Subsequence (LCS))

- *another similarity measurement*
- *given two sequences* $X[1 \ldots m]$ *and* $Y[1 \ldots n]$.
- *to find (the length of) a longest subseq. common to both.*

$$X : A\ B\ C\ B\ D\ A\ B$$
$$Y : B\ D\ C\ A\ B\ A$$
$$LCS : B\ D\ A\ B;\ B\ C\ A\ B;\ B\ C\ B\ A;$$

Longest Common Subsequence

$\boxed{L(i,j)\text{: the length of the LCS of } X[1\ldots i] \text{ and } Y[1\ldots j].}$

$\boxed{\text{Make choice:}}$ Let $Z[1\ldots k] = LCS(X[1\ldots i], Y[1\ldots j]$, whether $(X[i], Y[j]) \in Z[1\ldots k]$?

Lemma (Make Choice)

$\boxed{\textit{If}}$ $(X[i], Y[j]) \notin Z[1\ldots k]$, *then either* $X[i] \notin Z$ *or* $X[j] \notin Z$.

Longest Common Subsequence

$L(i, j)$: the length of the LCS of $X[1 \dots i]$ and $Y[1 \dots j]$.

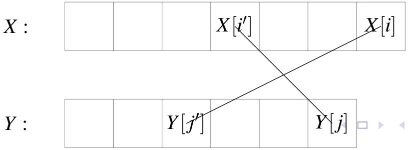
Make choice: Let $Z[1 \dots k] = LCS(X[1 \dots i], Y[1 \dots j])$, whether $(X[i], Y[j]) \in Z[1 \dots k]$?

Lemma (Make Choice)

If $(X[i], Y[j]) \notin Z[1 \dots k]$, *then either* $X[i] \notin Z$ *or* $X[j] \notin Z$.

Proof.

By contradiction. Or, case by case.



Longest Common Subsequence

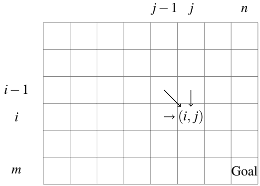
Theorem (Recurrence)

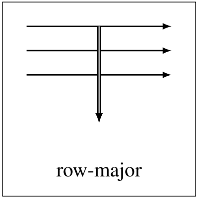
$$L(i,j) = \begin{cases} L(i-1,j-1)+1 \ \textit{(why?)} & \textit{if } X[i] = Y[j] \\ \max\{L(i,j-1), L(i-1,j)\} & \textit{o.w.} \end{cases}$$

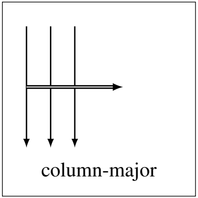
$\boxed{\textit{Base cases:}}$

$$L(i,0) = 0; L(0,j) = 0;$$

Longest Common Subsequence







Time complexity: $\Theta(nm)$
Space complexity: $\Theta(nm)$

Outline

Palindrome

Problem (Palindrome)

- *given a seq.* $X[1 \dots n]$
- *to find (the length of) a longest palindrome subseq.*

$$X : A\ C\ G\ T\ G\ T\ C\ A\ A\ A\ T\ C\ G$$
$$P : A\ C\ G\ C\ A; \quad A\ C\ G\ T\ G\ C\ A$$

Palindrome

$\boxed{L[i \dots j] : \text{the length of the LP subseq. of } X[i \dots j].}$

$\boxed{\text{Make choice:}}$ Let $Z[1 \dots k] = LP(X)$, whether $X[i] = Z[1] = X[j] = Z[k]$.

Theorem (Recurrence)

$\boxed{\textit{If}}$ *$X[i] \neq X[j]$, then either $X[i] \notin Z$ or $X[j] \notin Z$.*

Proof.

$\boxed{\text{By contradiction.}}$ □

Palindrome

Theorem (Recurrence)

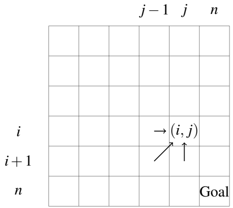
$$L(i,j) = \begin{cases} L(i+1, j-1) + 2 \ (\textit{why?}) & \textit{if } X[i] = Y[j] \\ \max\{L(i+1, j), L(i, j-1)\} & \textit{o.w.} \end{cases}$$

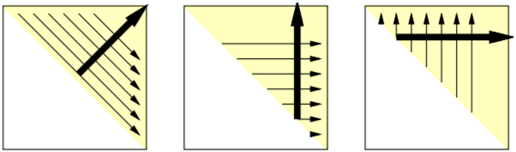
$\boxed{\textit{Base cases:}}$

$$L(i,i) = 1, \forall i = 1 \ldots n$$

Palindrome

Filling the table:





diagonal-major, row-major, column-major

Palindrome

Diagonal-major:

```
for i = 1 → n do
    L(i, i) = 1
end for

for l = 1 → n do
    for i = 1 → n − l do
        j = i + l
        L(i, j) = ...
    end for
end for
```

Questions:

- ED vs. LCS
- LCS vs. LP

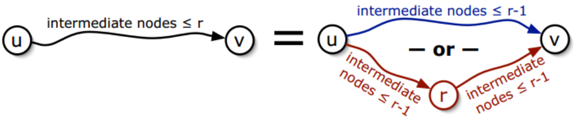
Outline

Outline

Floyd-Warshall Algorithm Revisited

$\boxed{D^k(i,j) \equiv D(i,j,k):}$ the length of the shortest path from i to j where intermediate vertex is numbered at most k.

Theorem (Recurrence)



$$D^k(i,j) = \min\{D^{k-1}(i,k) + D^{k-1}(k,j), D^{k-1}(i,j)\}$$

$\boxed{\textit{Base cases:}}$

$$D^0(i,j) = w(i \to j).$$

Outline

Outline

Independent Set in Tree

Problem

Independent Set in Tree

- $\boxed{\textit{Goal:}}$ *to find the largest independent set in tree*

Independent Set in Tree

$I(u)$: the size of the LIS of the subtree rooted at u.

Theorem (Recurrence)

$\boxed{\textit{Make choice:}}$ *whether u is in LIS.*

$$I(u) = \max\left\{\sum_{children\ w\ of\ u} I(w), 1 + \sum_{grandchildren\ w\ of\ u} I(w)\right\}.$$

$\boxed{\text{Base cases:}}$ $I(v) = 1, \forall$ leave in tree.

Outline

Outline

Google Eggs Game

Problem (Google Eggs Game)

- *n floors, m eggs*
- *test the "quality" of egg: critical floor c*
- *cases of $c = 0, c = n$*
- $\boxed{\textit{Goal:}}$ *to determine c while minimizing the number of throws*

Google Eggs Game

$T(i,j)$: minimum number of throws with i floors and j eggs

Theorem (Recurrence)

Make choice: *which floor to throw; broken or not.*

$$T(i,j) = \min_{1\le k\le i} \big(T(i,j \mid k)\big)$$

$$T(i,j \mid k) = \max\{T(i,j \mid k, Y), T(i,j \mid k, N)\}$$

$$T(i,j \mid k, Y) = 1 + T(k-1, j-1)$$

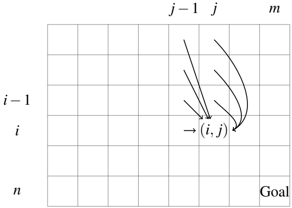
$$T(i,j \mid k, N) = 1 + T(i-k, j)$$

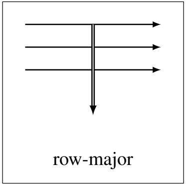
$$T(i,j) = \min_{1\le k\le i} \{1 + \max\{T(k-1,j-1), T(i-k,j)\}\}.$$

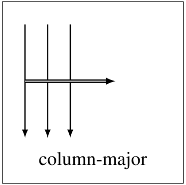
Base cases: $T(n,0) = 0$, $T(0,m) = 0$; $T(i,1) = i$, $T(1,j) = 1$

Google Eggs Game

Filling the table:







Strongly recommend: numerical experiments

Outline

Hungry-Lion Game

Problem (Hungry-Lion Game)

- *a sheep in danger*
- *hungry lions*