CS 170

Lecture 11

- use iClicker

Sanjam Garg.

join.iclicker.com/RGWR

DP Approach

1) Define appropriate subproblems
2) Write a recurrence relation
3) Determine order of computation.
   ↳ induces a DAG structure

Edit Distance

Input : $x[1 \ldots n]$ & $y[1 \ldots m]$

Goal : Find minimum # of keystrokes needed to edit x to y

1 keystroke is needed to add a character
remove a character
substitute a character

Example #

CAP → CUP → 1

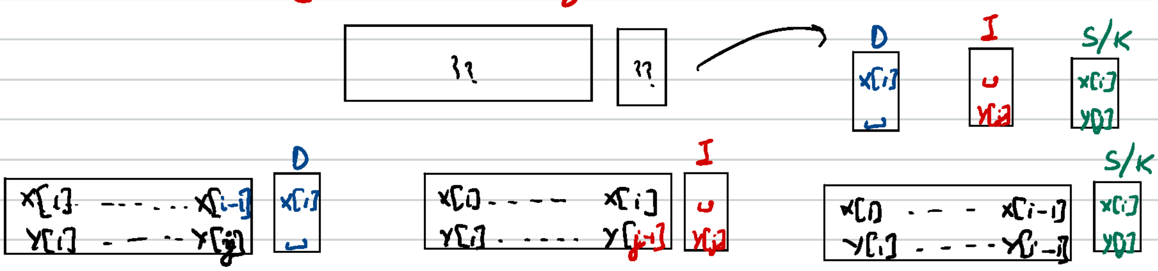
AAPPL → APPLE → 2

SUNNY → SNOWY → 3

SUNNY $\xrightarrow[U]{\text{del}}$ SNNY $\xrightarrow[N \to O]{\text{sub}}$ SNO_Y $\xrightarrow[W]{\text{insert}}$ SNOWY

Step 2: Recurrence relation

edit $x[1 \ldots i] \rightarrow y[1 \ldots j]$



$$E(i,j) \xrightarrow{\min} \begin{cases} 1 + E(i-1, j) \\ 1 + E(i, j-1) \\ \text{diff}(x[i], y[j]) + E(i-1, j-1) \end{cases}$$

$$\text{diff}(a,b) \rightarrow \begin{cases} 1 \text{ if } a \neq b \\ 0 \text{ if } a = b \end{cases}$$

Base Case $E(0,0) = 0 \quad \forall j \;\; E(0,j) = j = E(j,0)$

Algorithm:

Init: For all $i = 1 \dots n$ $\quad E(i,0) = i$
For all $j = 1 \dots m$ $\quad E(0,j) = j$

For all $i = 1 \dots n$
For all $j = 1 \dots m$

$$E(i,j) = \min \begin{cases} 1 + E(i-1,j) \\ 1 + E(i,j-1) \\ \text{diff}(x_i,y_j) + E(i-1,j-1) \end{cases}$$

Run_time: $O(nm)$

Subproblem Structure

(i) input $x_1 \dots x_n$ and a subproblem is $x_1 \dots x_i$

$\boxed{x_1 \dots x_i}\ x_{i+1} \dots x_n$

(ii) input $x_1 \dots x_n$ & $y_1 \dots y_m$ and a subproblem is $x_1 \dots x_i$ & $y_1 \dots y_j$

$\boxed{x_1 \dots x_i}\ x_{i+1} \dots x_n$

$\boxed{y_1 \dots y_j}\ y_{j+1} \dots y_m$

Knapsack Problem

Input: Total weight capacity of a knapsack W
List of n items with weights $w_1 \quad w_2 \quad \dots \quad w_n$ (integers)
values $v_1 \quad v_2 \quad \quad v_n$ (integers)

Goal: Find set of items that will maximize total value
with total weight $\leq W$

Example:

Item	Weight	Value
1	6	\$30
2	3	\$14
3	4	\$16
4	2	\$ 9

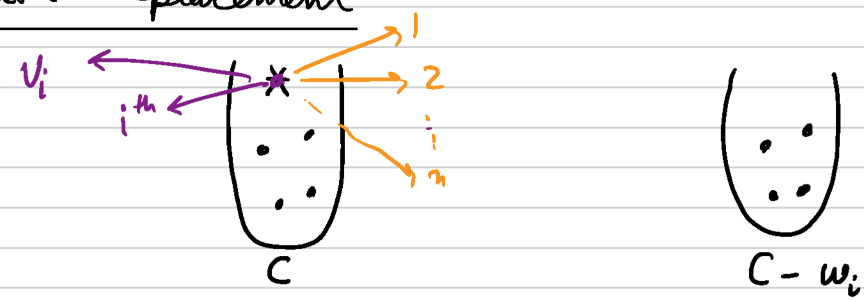
$W = 10$

Two variations: with repetition & without repetition.

With replacement : 1, 4, 4
↳ \$48

Without replacement : 1, 3
\$46

Knapsack with Replacement



Step 1: Subproblem

$K(C)$ = max value that can be packed in a bag of capacity C

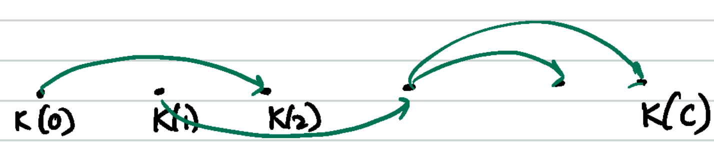
Step 2: Base

$K(0) = 0$

RR

$$K(C) = \max_{i : C \geq w_i} \{ v_i + K(C - w_i) \}$$

Step 3:



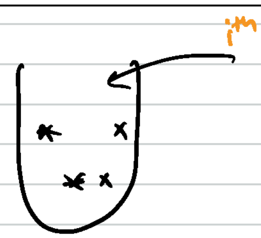
Algorithm

Input: W, $v[1\ldots n]$, $w[1\ldots n]$
$K(0)=0$ For $C=1$ to W $K(C) = \max\limits_{i:w_i \le C} \{ v_i + K(C-w_i) \}$ Output $K(W)$

$$O(\underbrace{W}_{\log(W)} \times n)$$

$$\underbrace{| \; | \; | \; | \; | \ldots |}_{100 \text{ times}} \to \underbrace{100}$$

Knapsack without repetition



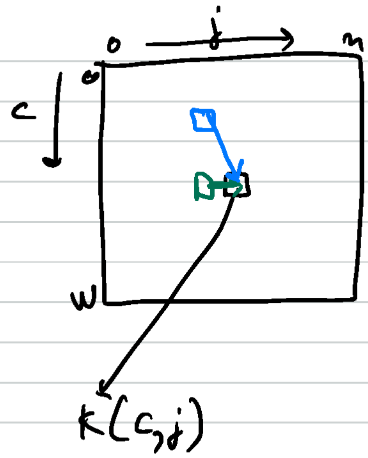
$K(W, n)$

Step1: $K(C, j) =$ Max value that can be packed in a bag of capacity C using items $1 \ldots j$ without repetition.

Step2: Base: $K(0, j) = 0 \quad \forall j$

RR:
$$K(C, j) \begin{cases} K(C, j-1) & w_j > C \\ \max\{\underbrace{\boxed{K(C, j-1)}}_{j^{th}\text{ item is unused}}, \underbrace{\boxed{v_j + K(C-w_j, j-1)}}_{j^{th}\text{ item is used}}\} & w_j \le C \end{cases}$$

Step :



$O(Wn)$

Algorithm

Input : W, $v[1 \ldots n]$, $w[1 \ldots n]$

For $C = 0 \ldots W$
    $K(C, 0) = 0$
For $\ell = 1 \ldots n$
    For $C = 1 \ldots W$
        $K(C, \ell) = \max \{ K(C, \ell - 1), \overbrace{v_\ell + K(C - w_\ell, \ell - 1)}^{\text{only if } C - w_\ell \geq 0} \}$

Output $K(W, n)$

Memoization

Initialize: hash table, initially empty, holds $K(w)$ indexed by w

Knapsack(w)

if w is in hash table: return $K(W)$

$K(w) = \max_i \{ \text{knapsack}(w - w_i) + v_i : w_i \leq w \}$

insert $K(w)$ into hash table with key w

return $K(w)$.

Chain Matrix Multiplication

Multiply $\overset{m_0 \times m_1}{A} \quad \overset{m_1 \times m_2}{B}$

$$(A \times B)_{ij} = \sum_{j=1}^{m_1} A_{ij} B_{jk}$$

$m_0 m_1 m_2$ multiplications.

How to multiply? $\underset{50\times 20}{A} \times \underset{20\times 1}{B} \times \underset{1\times 10}{C} \times \underset{10\times 100}{D}$

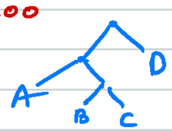
Goal: Find the best paranthezation?

$\underbrace{(A \times \overbrace{(B \times C}^{20\times 1\times 10}))}_{50\times 20\times 10} \times D$

$20\times 1\times 10 + 50\times 20\times 10 + 50\times 10\times 100$

$= 200 + 10{,}000 + 50{,}000$

$= 60{,}200$

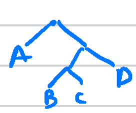


$A \times ((B \times C) \times D)$

$20\times 1\times 10 + 20\times 10\times 100 + 50\times 20\times 100$

$= 200 + 20{,}000 + 100{,}000$

$= 120{,}200$

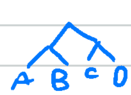


$\overbrace{(A \times B)}^{50\times 20\times 1} \times \overbrace{(C \times D)}^{1\times 10\times 100}$

$50\times 20\times 1 + 1\times 10\times 100 + 50\times 1\times 100$

$= 1000 + 1000 + 5000$

$= 7000$



Subproblem Structure

(i) input $x_1 \ldots x_n$ and a subproblem is $x_1 \ldots x_i$

$$\boxed{x_1 \ldots x_i}\ x_{i+1} \ldots x_n$$

(ii) input $x_1 \ldots x_n$ & $y_1 \ldots y_m$ and a subproblem is $x_1 \ldots x_i$ & $y_1 \ldots y_j$

$$\boxed{x_1 \ldots x_i}\ x_{i+1} \ldots x_n$$

$$\boxed{y_1 \ldots y_j}\ y_{j+1} \ldots y_m$$

Input: $\overset{m_0 \times m_1}{A_1}\ \overset{m_1 \times m_2}{A_2} \ldots \overset{m_{n-1} \times m_n}{A_n}$

Output: Minimum number of multiplications needed.

$M(i)$ = the min number of multiplications needed to multiply $A_1 \times A_2 \ldots A_i$

$$A_1 \ldots A_i$$

$$(A_1\ A_{i_1})\quad (A_{i_1+1} \ldots A_i)$$

$$(A_1\ A_{i_2})\quad (A_{i_2+1}\ A_{i_1}) \ldots$$

Subproblem Structure

(i) input $x_1 \dots x_n$ and a subproblem is $x_1 \dots x_i$

$$\boxed{x_1 \dots x_i}\ x_{i+1} \dots x_n$$

(ii) input $x_1 \dots x_n$ & $y_1 \dots y_m$ and a subproblem is $x_1 \dots x_i$ & $y_1 \dots y_j$

$$\boxed{x_1 \dots x_i}\ x_{i+1} \dots x_n$$

$$\boxed{y_1 \dots y_j}\ y_{j+1} \dots y_m$$

(iii) input $x_1 \dots x_n$ and a subproblem is $x_i \dots x_j$

$$x_1 \dots \boxed{x_i \dots x_j} \dots x_n$$

Input: $\overset{m_0 \times m_1}{A_1}\ \overset{m_1 \times m_2}{A_2} \dots \overset{m_{n-1} \times m_n}{A_n}$

Output: Minimum number of multiplications needed.

Step 1: $M(i,j)$ = the min # of multiplications needed to multiply $A_i \times A_{i+1} \dots A_j$

$i \le j$

Step 2: Base : $M(i,i) = 0 \ \forall i = 1 \dots n$

$A_i \dots A_j \rightarrow j-i$

$$M(i,j) = \min_{i \le k < j} \{ M(i,k) + M(k+1,j) + m_{i-1} m_k m_j \}$$

$j > i$

① $A_i \times (A_{i+1} \dots A_j) \rightarrow 1, \ j-i-1$

② $(A_i \times A_{i+1}) \times (A_{i+2} \dots A_j) \rightarrow 2, \ j-i-2$

$\vdots$

$(A_i \dots A_{j-1}) \times A_j \rightarrow j-i-1, \ 2$

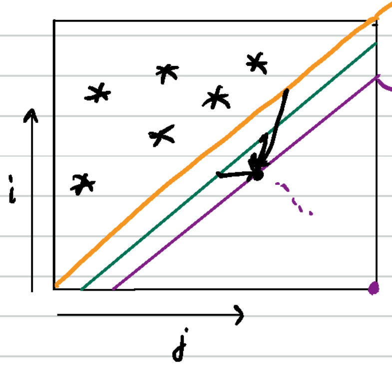
$$\overset{m_{i-1} \times m_k}{(A_i \dots A_k)} \times \overset{m_k \times m_j}{(A_k \dots A_j)}$$

$i, j \longrightarrow i, k \ ; \ k+1, j$

$k - i$ & $j - (k+1) < j - i$

Step3: Order of computation

$A_1 \times A_2 \times A_3 \times A_4 \cdots A_n$



Base Case $(i-j)=0$

$j-i=1$

$j-i=2$

$j-i=n$

Algorithm

For $i = 1, \dots n$ $\quad M(i,i) = 0$

For $s = 1, \dots n-1$

For $i = 1 \dots n-s$

$j = i+s$

$$M(i,j) = \min_{k=i \dots j} \{ M(i,k) + M(k+1,j) + m_{i-1}\, m_k\, m_j \}$$

Return $M(1,n)$

$$O(\underbrace{n^2}_{\#\text{ of subproblems}} \times \underbrace{n}_{\text{time spent to solve each subproblem}})$$