CS6033 Lectures 9-10 Slides/Notes

Dynamic Programming
(Notes, Ch 14)

By Prof. Yi-Jen Chiang
CSE Dept., Tandon School of Engineering
New York University

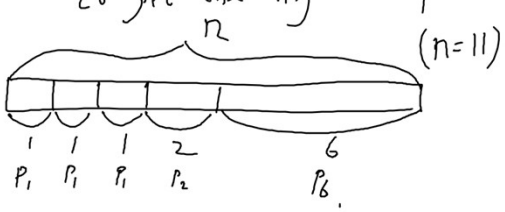
1

Dynamic Programming (ch 14)

Ex. Rod Cutting

We have a rod of length n (positive integer) and given prices $p_1, p_2, \cdots p_n$ where p_i is the price for rod with length i.

Goal: cut the rod of length n into pieces and sell them to get the highest price.



2

Let r_n be the max price for rod of length n.

For any general length j. r_j: max price for length j

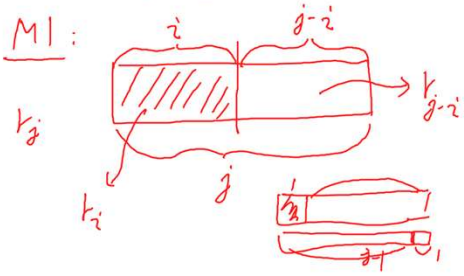
Derive a recursive solution for r_j:

Main step for solving a D.P. problem

Requires creativity

Express the solution in terms of the sol of the subproblems

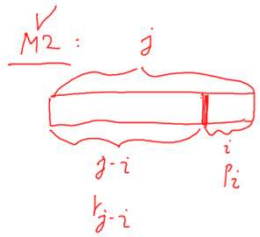
M1:



$$r_j = \max_{1 \le i \le j-1} \{ r_i + r_{j-i}, p_j \} \quad j \ge 1$$

$$r_0 = 0 \quad j = 0$$

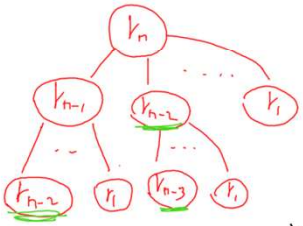
M2:



$$r_j = \max_{1 \le i \le j} \{ p_i + r_{j-i} \} \quad \begin{pmatrix} i = j \\ p_j + r_0 \end{pmatrix} \quad j \ge 1$$

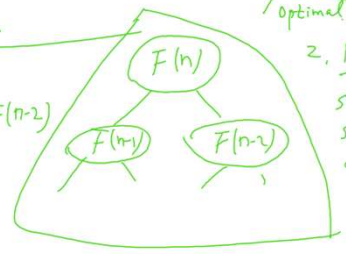
$$r_0 = 0$$

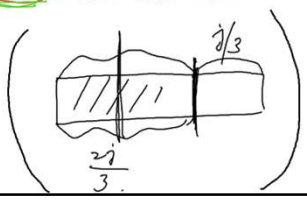
$$\Rightarrow r_n \quad \left(r_n = \max_{1 \le i \le n} \{ p_i + r_{n-1} \} \right)$$



Exponential size in recursion tree.

$$F(n) = F(n-1) + F(n-2)$$





Structure of DP

1. Optimal subproblems

opt. sol. of the current problem includes the sols for subproblems

optimal

2. Repeated subproblems

store the results of subproblems from previous computation to avoid repeated computations

3

4

2. Repeated Subproblems

Store results from previous computations to avoid repeated computations.

① Top-down recursion with "memoization"

② Build up results bottom up. (∵ s values of r_j iteratively).

r_n

$r_j = \max_{1 \le i \le j} \{ p_i + r_{j-i} \}$

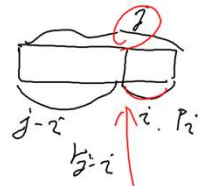
$1 \to n$

$r_0 = 0$

$r_1 = p_1$

$r_2 = \{ p_2, \; p_1 + r_1 \}$

$\vdots$



$r[0] \leftarrow 0.$

For $j = 1$ to n.

  $r[j] \leftarrow -\infty.$

  For $i = 1$ to j

    // compute r_j.

    temp $\leftarrow p_i + r[j-i]$

    if (temp > $r[j]$)

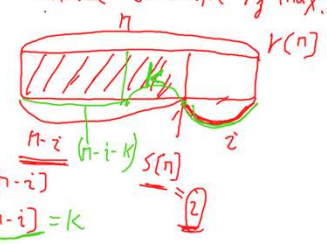
      $r[j] \leftarrow$ temp;

      $s[j] \leftarrow i$;

$\boxed{O(n^2) \text{ time}}$

$s[j] \leftarrow i$

$s[j]$ stores the optimal choice to make r_j max.



$r(n-i)$

$s[n-i] = k$

5

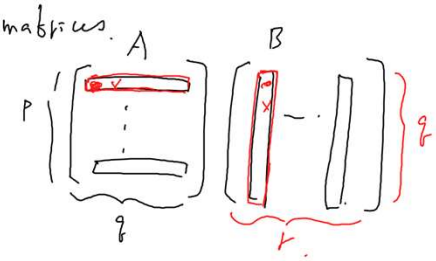
EX: Matrix Chain Multiplication

A, B are matrices.

$C = A \cdot B.$

$A: p \times q$

$B: q \times r$



$C: p \times r.$

$c_{ij} = \sum_{k=1}^{q} A_{ik} \cdot B_{kj}$] q terms addition.

cost: $p \cdot q \cdot r.$

Each entry c_{ij} needs q time.

there are $p \cdot r$ entries in C

cost of multiplication of A · B is $p \cdot q \cdot r$.

$(p \times q)(q \times r)$

Matrices $A_1 \; A_2 \; A_3$

$A_1 \cdot A_2 \cdot A_3$

$(10 \times 50) \; (50 \times 5) \; (5 \times 100)$

M1: $(A_1 \cdot A_2) \cdot A_3$.

$\boxed{10 \cdot 50 \cdot 5}$

(10×5) $\quad 5 \times 100$

$\boxed{10 \times 5 \times 100}$

Faster

cost of M1:

$10 \cdot 50 \cdot 5 + 10 \times 5 \times 100$

M2: $A_1 \cdot (A_2 \cdot A_3)$

$(50 \times 5) \cdot (5 \times 100)$ → $50 \times 5 \times 100.$

$(10 \times 50) \cdot (50 \times 100)$

$10 \times 50 \times 100$

cost of M2

$= 50 \times 5 \times 100 + 10 \times 50 \times 100$

6

Problem: Given a sequence of n matrices: $A_1\ A_2\ A_3 \cdots A_n$ $\quad (A_i : p_{i-1} \times p_i)$

$(p_0 \times p_1)\ (p_1 \times p_2)\ (p_2 \times p_3) \quad (p_{n-1} \times p_n)$

Goal: Place parentheses (i.e. decide the order of multiplication) to minimize the cost of computing $A_1 \cdot A_2 \cdot A_3 \cdots A_n$.

Define: $m[1,n]$ be the min cost of multiplying $A_1 \cdots A_n$

$$m[1,n] = \min_{1 \le k \le n-1} \{ m[1,k] + m[k+1,n] + p_0 p_k p_n \}$$

$A[k+1, j]$ $\quad j$

$(A_{k+1} \cdots A_n)$

$m[k+1, n]$

$(A_1)(\quad)$

$i \quad (\quad j \quad A_{n-1})(A_n)$

$(A_1 A_2 \cdots A_k)(A_{k+1} \cdots A_n)$

$\boxed{m[1,k]} + \boxed{m[k+1,n]}$

$(p_0 \times p_1) \quad (p_{k-1} \times p_k) \quad (p_k \times p_{k+1}) \cdots (p_{n-1} \times p_n)$

$(p_0 \times p_k) \quad (p_k \times p_n)$

$\boxed{p_0 p_k p_n}$

The general subproblem: $m[i,j]$ $\quad i, j$ are general positions.

7

$$m[i,j] = \begin{cases} \min_{i \le k \le j-1} \{ m[i,k] + m[k+1,j] + p_{i-1} p_k p_j \} & j > i \\ 0 & j = i \end{cases}$$

(A_i)

$(A_i \cdots A_k)(A_{k+1} \cdots A_j)$

$m[i,k] \quad m[k+1,j]$

$(p_{i-1} \times p_i) \cdots (p_{k-1} \times p_k) \quad (p_k \times p_{k+1}) \cdots (p_{j-1} \times p_j)$

$(p_{i-1} \times p_k) \quad (p_k \times p_j)$

cost: $p_{i-1} \cdot p_k \cdot p_j$

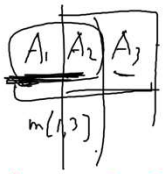
$\boxed{\text{Final result: } m[1,n]}$

Next task: Use iterative method and a table to compute $m[i,j]$ for $i = 1, \cdots n$, $j = 1, \cdots n$

Use table entry $m[i][j]$ for $m[i,j]$

table entry $\quad$ recursive function value.

8



Key idea: We need to increase the chain length ℓ from 2, 3, 4 ---

e.g. $\ell = 2$ A_1A_2, A_2A_3, A_3A_4 --- $A_1A_2A_3A_4$ ---

$\ell = 3$. $A_1A_2A_3$ $\ell = 2$

$m[i,j] = \min_{i \le k \le j-1} \{ m[i,k] + m[k+1,j] + P_{i-1}P_kP_j \}$

$m[i,j]$: We want $i = 1, \dots, n$, $j = 1, \dots, n$

But this is NOT a feasible order!!

$m[i,i] \leftarrow 0 \ \forall i = 1, 2, \dots, n$.

For $\ell = 2$ to n do

  For $i = 1$ to $\boxed{x}$ do ($n - \ell + 1$)

    $j = i + \ell - 1$; $m[i,j] \leftarrow \infty$

    For $k = i$ to $j-1$ do

      temp $\leftarrow m[i,k] + m[k+1,j] + P_{i-1}P_kP_j$;

      if (temp < $m[i,j]$) { $m[i,j] \leftarrow$ temp; $s[i,j] \leftarrow k$; }

$\overbrace{A_i \cdots A_j}^{\ell}$

$j - i + 1 = \ell$

$\Rightarrow j = i + \ell - 1$

$\underbrace{A_1 \cdots A_x}_{\ell} \overbrace{\cdots A_n}^{\ell}$

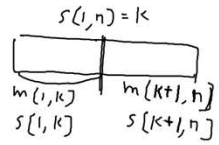
$\overbrace{A_x \cdots A_n}^{\ell}$

last i

$n - x + 1 = \ell$

$\Rightarrow \boxed{x = n - \ell + 1}$

$O(n^3)$ time



Final solution: $m[1,n]$, $s[1,n]$.

9

Ex: Longest Common Subsequence.

Given 2 strings e.g. $X = ABCBDAB$, $Y = BDCABA$

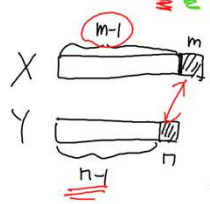
common subsequence: BCAB, CAB, ABA.

Find the longest common subsequence between X and Y.

Suppose X has length m $X[1..m]$ common subsequence in $Z[1..k]$

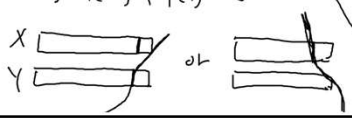
Y " n. $Y[1..n]$

Define: $c(m,n)$: length of the longest common subsequence between $X[1..m]$, $Y[1..n]$



If $X(m) = Y(n)$ then $c(m,n) = 1 + c(m-1, n-1)$

If $X(m) \neq Y(n)$ then



$c(m,n) = \max\{ c(m, n-1), c(m-1, n) \}$

10

General Subproblem:

X: 1 … i

Y: 1 … j

$$c[i,j] = \begin{cases} 1 + c[i-1, j-1] & \text{if } X[i] = Y[j] \\ \max\{c[i,j-1], c[i-1,j]\} & \text{if } X[i] \neq Y[j] \end{cases}$$

boundary conditions: $c[0,j] = 0 \quad \forall j = 1, \dots n.$

$c[i,0] = 0 \quad \forall i = 1, \dots m.$

```
For i = 1 to m
  For j = 1 to n
    if (X[i] == Y[j]) → { c[i,j] = 1 + c[i-1, j-1];
                          s[i,j] = 'match' }
    else if (c[i,j-1] > c[i-1,j]) { c[i,j] ← c[i,j-1]
                                    s[i,j] ← 'Y goes left' }
         else { c[i,j] ← c[i-1,j]
                s[i,j] ← 'X goes left' }
```

time: $O(mn)$

11

Optimal Binary Search Trees:

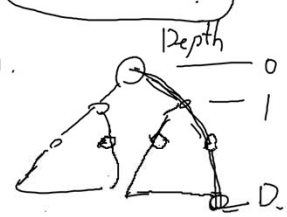
Given sorted keys $k_1 < k_2 < k_3 < \dots < k_n$ (with probabilities $p_1, p_2, p_3, \dots, p_n$)

p_i: probability that key k_i will be searched.

dummy integral values $d_0 \quad d_1 \quad d_2 \quad d_3 \quad \dots \quad d_n$

unsuccessful search $q_0 \quad q_1 \quad q_2 \quad q_3 \quad q_n$.

$$\sum_{i=1}^{n} p_i + \sum_{i=0}^{n} q_i = 1$$



Expected search cost $= \sum_{i=1}^{n} (\text{depth}(k_i)+1) \cdot p_i + \sum_{i=0}^{n} (\text{depth}(d_i)+1) \cdot q_i$

nodes visited during a search.

$$= 1 + \sum_{i=1}^{n} \text{depth}(k_i) \cdot p_i + \sum_{i=0}^{n} \text{depth}(d_i) \cdot q_i$$

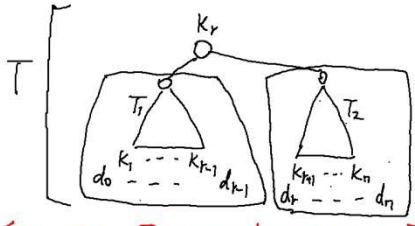
Goal: Put the keys $k_1 \dots k_n$ to a binary search tree (each node stores a key) s.t. the expected search cost is minimized

12

$$\begin{array}{ccccccccc} & p_1 & & p_2 & & p_3 & & & p_n \\ & k_1 & < & k_2 & < & k_3 & < \cdots & < & k_n \\ d_0 & & d_1 & & d_2 & & d_3 & \cdots & d_n \\ q_0 & & q_1 & & q_2 & & q_3 & & q_n \end{array}$$

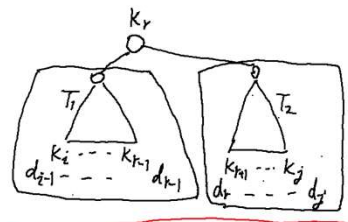
p_i: probability that key k_i will be searched.

Expected search cost $= 1 + \sum_{i=1}^{n} \text{depth}(k_i) \cdot p_i + \sum_{i=0}^{n} \text{depth}(d_i) \cdot q_i$



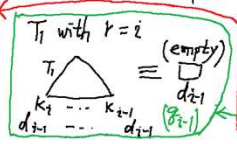
We want: $e[1, n]$ optimal in range $k_1 \cdots k_n$, $d_0 \cdots d_n$

When splitting T_1/T_2 $\Rightarrow$ We need general $e[i, j]$



For $\sum_k p_k$, $\sum_k q_k$: compute prefix sum first

(See textbook for additional details)



$$e[i,j] = \min_{i \le r \le j} \left\{ e[i, r-1] + \left(\sum_{k=i}^{r-1} p_k + \sum_{k=i-1}^{r-1} q_k \right) + p_r + e[r+1, j] + \left(\sum_{k=r+1}^{j} p_k + \sum_{k=r}^{j} q_k \right) \right\} \text{ if } i \le j$$

$e[i, i-1] = q_{i-1}$

13

DP: other variations (1) cost functions need NOT be max, or min

Tips: They can be boolean functions.

$$B[\ ] = \begin{cases} \text{true} & 1 \\ \text{false} & 0. \end{cases}$$

op: and, or, --(boolean ops)

(2) Ex: knapsack problem: items $1, 2, 3, \cdots, n$.

weights: $w_1, w_2, \cdots, w_n$.

benefits: $b_1, b_2, \cdots, b_n$.

knapsack capacity W (W is an integer > 0).

We want to put items into knapsack: $\sum_{i \in \text{knapsack}} w_i \le W$.

s.t. $\sum_{i \in \text{knapsack}} b_i$ is max.

Encode the info exactly: $\sum_{i \in \text{knapsack}} w_i = w$, consider $w = 1, \cdots, W$.

14