I214 System Optimization
Chapter 6: Combinatorial Optimization (Part B)

Brian Kurkoski

Japan Advanced Institute of Science and Technology

2023 January

Outline

6.3 Dynamic Programming

Dynamic programming solves an optimization problem by dividing the main problem into smaller subproblems.

If it is possible to find an optimal solution to the subproblems, then this can be used to find the optimal solution of the main problem.

A problem exhibits optimal substructure if the optimal solution contains optimal solutions to subproblems.

To find a dynamic programming method for solving a specific problem:

1. Characterize the structure of the optimal solution.
2. Recursively define the value of the optimal solution.
3. Compute the optimal value.
4. Construct the optimal solution.

6.3.1 Rod Cutting

Problem

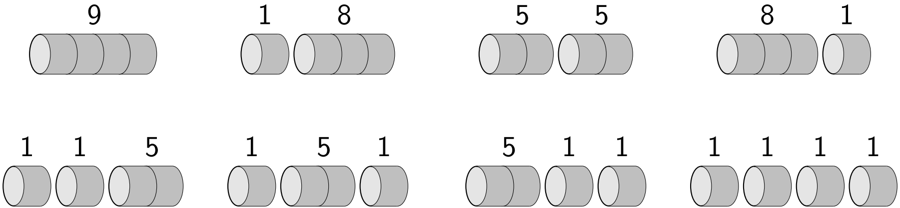
The Stone River Supply Company buys long steel rods and cuts them; each cut has no cost. For a rod of length i meters, the income is p_i dollars, given by the following table:

length i	1	2	3	4	5	6	7	8	9	10
income p_i	1	5	8	9	10	17	17	20	24	30

Given a long rod of length n, the company wants to cut it to maximize the total income r_n.

Rod Cutting with $n = 4$

All ways to cut a rod of length $n = 4$; the number above the rod gives the income. The most efficient way is denoted $2 + 2 = 4$. This gives income $r_4 = p_2 + p_2 = 5 + 5 = 10$.



length i	1	2	3	4	5	6	7	8	9	10
income p_i	1	5	8	9	10	17	17	20	24	30

6.3.2 Chain of Matrix Multiplication

When we compute matrix multiplication of n matrices $A_1 A_2 \cdots A_n$, we repeatedly perform multiplication of two matrices.

Now we want to find the order of matrix multiplications for minimizing the total number of scalar multiplications.

Corresponds to the assignment of parentheses

Use the dynamic programming method:

1. Characterize the structure of the optimal solution.
2. Recursively define the value of the optimal solution.
3. Compute the optimal value.
4. Construct the optimal solution.

Example

We consider $A_1 A_2 \cdots A_6$, where the size of each matrix is given in the table.

matrix	size
A_1	30×35
A_2	35×15
A_3	15×5
A_4	5×10
A_5	10×20
A_6	20×25

Matrix Chain Table

	j=1	*j=2*	*j=3*	*j=4*	*j=5*	*j=6*
i=1	0 	15750 1	7875 1	9375 3	11875 3	15125 3
i=2		0 	2625 2	4375 3	7125 3	10500 3
i=3			0 	750 3	2500 3	5375 3
i=4				0 	1000 4	3500 5
i=5					0 	5000 5
i=6						0

m[i,j]
s[i,j]

Matrix Chain Table

	j=1	j=2	j=3	j=4	j=5	j=6
i=1	0	15750 1	7875 1	9375 3	11875 3	15125 3
i=2		0	2625 2	4375 3	7125 3	10500 3
i=3			0	750 3	2500 3	5375 3
i=4				0	1000 4	3500 5
i=5					0	5000 5
i=6						0

m[i,j] *s[i,j]*

	j=1	j=2	j=3	j=4	j=5	j=6
i=1	**0**	15750 1	**7875 1**	9375 3	11875 3	**15125 3**
i=2		0	**2625 2**	4375 3	7125 3	10500 3
i=3			0	750 3	2500 3	5375 3
i=4				0	**1000 4**	**3500 5**
i=5					0	5000 5
i=6						**0**

m[i,j] *s[i,j]*

Figure 1: Table of $m[i,j]$, $s[i,j]$.

Class Info

- Homework 6. Extended Deadline: Monday, January 30 at 18:00 (today).
- Next lecture: Friday, February 3. Finish Combinatorial Optimization
- Homework 7 (KKT conditions). Deadline: February 3 at 18:00
- Homework 8 (dynamic programming). Do not submit. Solutions will be published before final exam.
- Final lecture: Monday, February 6. Review course.
- Final exam: Wednesday, February 8 at 13:30–15:10.
  - Covers Chapters 5 and 6 only
  - Differs from midterm: not all numerical problems, some analytical problems