Matrix Multiplication Chains

- Determine the best way to compute the matrix product $M_1 x M_2 x M_3 x \ldots x M_q$.
- Let the dimensions of M_i be $r_i x r_{i+1}$.
- $q-1$ matrix multiplications are to be done.
- Decide the matrices involved in each of these multiplications.

Decision Sequence

- $M_1 x M_2 x M_3 x \ldots x M_q$
- Determine the $q-1$ matrix products in reverse order.
  - What is the last multiplication?
  - What is the next to last multiplication?
  - And so on.

Problem State

- $M_1 x M_2 x M_3 x \ldots x M_q$
- The matrices involved in each multiplication are a contiguous subset of the given q matrices.
- The problem state is given by a set of pairs of the form (i, j), $i <= j$.
  - The pair (i,j) denotes a problem in which the matrix product $M_i x M_{i+1} x \ldots x M_j$ is to be computed.
  - The initial state is $(1,q)$.
  - If the last matrix product is $(M_1 x M_2 x \ldots x M_k) x (M_{k+1} x M_{k+2} x \ldots x M_q)$, the state becomes $\{(1,k), (k+1,q)\}$.

Verify Principle Of Optimality

- Let $M_{ij} = M_i x M_{i+1} x \ldots x M_j$, $i <= j$.
- Suppose that the last multiplication in the best way to compute M_{ij} is $M_{ik} x M_{k+1,j}$ for some k, $i <= k < j$.
- Irrespective of what k is, a best computation of M_{ij} in which the last product is $M_{ik} x M_{k+1,j}$ has the property that M_{ik} and $M_{k+1,j}$ are computed in the best possible way.
- So the principle of optimality holds and dynamic programming may be applied.

Recurrence Equations

- Let $c(i,j)$ be the cost of an optimal (best) way to compute M_{ij}, $i <= j$.
- $c(1,q)$ is the cost of the best way to multiply the given q matrices.
- Let $kay(i,j) = k$ be such that the last product in the optimal computation of M_{ij} is $M_{ik} x M_{k+1,j}$.
- $c(i,i) = 0, 1 <= i <= q$. $(M_{ii} = M_i)$
- $c(i,i+1) = r_i r_{i+1} r_{i+2}, 1 <= i < q$. $(M_{ii+1} = M_i x M_{i+1})$
- $kay(i,i+1) = i$.

c(i, i+s), 1 < s < q

- The last multiplication in the best way to compute $M_{i,i+s}$ is $M_{ik} x M_{k+1,i+s}$ for some k, $i <= k < i+s$.
- If we knew k, we could claim:

  $$c(i,i+s) = c(i,k) + c(k+1,i+s) + r_i r_{k+1} r_{i+s+1}$$

- Since $i <= k < i+s$, we can claim

  $c(i,i+s) = \min\{c(i,k) + c(k+1,i+s) + r_i r_{k+1} r_{i+s+1}\}$, where the $\min$ is taken over $i <= k < i+s$.
- $kay(i,i+s)$ is the k that yields above $\min$.

Recurrence Equations

- $c(i,i+s) =$

  $$\min_{i <= k < i+s}\{c(i,k) + c(k+1,i+s) + r_i r_{k+1} r_{i+s+1}\}$$

- $c(*,*)$ terms on right side involve fewer matrices than does the $c(*,*)$ term on the left side.
- So compute in the order $s = 2, 3, \ldots, q-1$.

Recursive Implementation

- See text for recursive codes.
- Code that does not avoid recomputation of already computed $c(i,j)$s runs in $Omega(2^q)$ time.
- Code that does not recompute already computed $c(i,j)$s runs in $O(q^3)$ time.
- Implement nonrecursively for best worst-case efficiency.

Example

- q = 4, (10 x 1) * (1 x 10) * (10 x 1) * (1 x 10)
- r = [r_1,r_2 ,r_3,r_4,r_5] = [10, 1, 10, 1, 10]

c(i,j), i <= j

kay(i,j), i <= j

s = 0

c(i,i) and kay(i,i) , 1 <= i <= 4 are to be computed.

c(i,j), i <= j

kay(i,j), i <= j

s = 1

c(i,i+1) and kay(i,i+1) , 1 <= i <= 3 are to be computed.

c(i,j), i <= j

kay(i,j), i <= j

s = 1

- c(i,i+1) = $r_i r_{i+1} r_{i+2}$, 1 <= i < q. (M_{ii+1}= M_ix M_{i+1})
- kay(i,i+1) = i.
- r = [r_1,r_2 ,r_3,r_4,r_5] = [10, 1, 10, 1, 10]

c(i,j), i <= j

kay(i,j), i <= j

s = 2

- $c(i,i+2) = \min\{c(i,i) + c(i+1,i+2) + r_i r_{i+1} r_{i+3},$
  $c(i,i+1) + c(i+2,i+2) + r_i r_{i+2} r_{i+3}\}$
- $r = [r_1, r_2, r_3, r_4, r_5] = [10, 1, 10, 1, 10]$

0	100		
	0	10	
		0	100
			0

c(i,j), i <= j

0	1		
	0	2	
		0	3
			0

kay(i,j), i <= j

s = 2

- $c(1,3) = \min\{c(1,1) + c(2,3) + r_1 r_2 r_4,$
  $c(1,2) + c(3,3) + r_1 r_3 r_4\}$
- $r = [r_1, r_2, r_3, r_4, r_5] = [10, 1, 10, 1, 10]$
- $c(1,3) = \min\{0 + 10 + 10, 100 + 0 + 100\}$

0	100	20	
	0	10	
		0	100
			0

c(i,j), i <= j

0	1	1	
	0	2	
		0	3
			0

kay(i,j), i <= j

s = 2

- $c(2,4) = \min\{c(2,2) + c(3,4) + r_2 r_3 r_5,$
  $c(2,3) + c(4,4) + r_2 r_4 r_5\}$
- $r = [r_1, r_2, r_3, r_4, r_5] = [10, 1, 10, 1, 10]$
- $c(2,4) = \min\{0 + 100 + 100, 10 + 0 + 10\}$

0	100	20	
	0	10	20
		0	100
			0

c(i,j), i <= j

0	1	1	
	0	2	3
		0	3
			0

kay(i,j), i <= j

s = 3

- $c(1,4) = \min\{c(1,1) + c(2,4) + r_1 r_2 r_5,$
  $c(1,2) + c(3,4) + r_1 r_3 r_5, c(1,3) + c(4,4) + r_1 r_4 r_5\}$
- $r = [r_1, r_2, r_3, r_4, r_5] = [10, 1, 10, 1, 10]$
- $c(1,4) = \min\{0+20+100, 100+100+1000, 20+0+100\}$

0	100	20	120
	0	10	20
		0	100
			0

c(i,j), i <= j

0	1	1	1
	0	2	3
		0	3
			0

kay(i,j), i <= j

Determine The Best Way To Compute M_{14}

- kay(1,4) = 1.
- So the last multiplication is $M_{14} = M_{11} \times M_{24}$.
- M_{11} involves a single matrix and no multiply.
- Find best way to compute M_{24}.

0	100	20	120
	0	10	20
		0	100
			0

c(i,j), i <= j

0	1	1	1
	0	2	3
		0	3
			0

kay(i,j), i <= j

Determine The Best Way To Compute M_{24}

- kay(2,4) = 3 .
- So the last multiplication is $M_{24} = M_{23} \times M_{44}$.
- $M_{23} = M_{22} \times M_{33}$.
- M_{44} involves a single matrix and no multiply.

0	100	20	120
	0	10	20
		0	100
			0

c(i,j), i <= j

0	1	1	1
	0	2	3
		0	3
			0

kay(i,j), i <= j

The Best Way To Compute M_{14}

- The multiplications (in reverse order) are:
  - $M_{14} = M_{11} \times M_{24}$
  - $M_{24} = M_{23} \times M_{44}$
  - $M_{23} = M_{22} \times M_{33}$

Time Complexity

	1	2	3	4
1	0	100	20	120
2		0	10	20
3			0	100
4				0

c(i,j), i <= j

- $O(q^2)$ c(i,j) values are to be computed, where q is the number of matrices.
- $c(i,i+s) = \min_{i <= k < i+s}\{c(i,k) + c(k+1,i+s) + r_i r_{k+1} r_{i+s+1}\}$.
- Each c(i,j) is the min of O(q) terms.
- Each of these terms is computed in O(1) time.
- So all c(i,j) are computed in $O(q^3)$ time.

Time Complexity

	1	2	3	4
1	0	1	1	1
2		0	2	3
3			0	3
4				0

kay(i,j), i <= j

- The traceback takes O(1) time to determine each matrix product that is to be done.
- q-1 products are to be done.
- Traceback time is O(q).