The Knuth-Yao Quadrangle Inequality Speedup is a Consequence of Total Monotonicity

Wolfgang W. Bein	(University of Nevada)
Mordecai J. Golin	(Hong Kong UST)
Lawrence L. Larmore	(University of Nevada)
Yan Zhang	(Hong Kong UST)

Motivation

- Nothing new: material here goes back 20-30 years.
- There are two classic cookbook
  Dynamic Programming Speedups in the literature:
  Knuth-Yao technique & SMAWK algorithm.
- They "feel" similar. Are they related?
- Knuth-Yao predates online algorithms.
  Can the KY speedup be maintained online?
- Answers to the two questions turned out to be related.
- Note: major confusion arises in the analysis because certain essential terms, e.g., quadrangle-inequality, monotone and online-algorithm have been used in very different ways in the two techniques' literature.

Outline

- Background
  - Kunth-Yao (KY) Quadrangle Inequality (QI) Speedup
  - SMAWK Algorithm for finding
    Row Minima of Totally Monotone (TM) Matrices
- The D^d Decomposition
  A transformation from QI to TM such that
  SMAWK solves KY problem as quickly as KY.
- The L^m and R^m Decompositions
  Another transformation from QI to TM that
  (1) implies KY speedup and (2) enables online solution.
- Extensions
  Applying the technique to known generalizations of KY.

Outline

- Background
  - Kunth-Yao (KY) Quadrangle Inequality (QI) Speedup
  - SMAWK Algorithm for finding
    Row Minima of Totally Monotone (TM) Matrices
- The D^d Decomposition
  A transformation from QI to TM such that
  SMAWK solves KY problem as quickly as KY.
- The L^m and R^m Decompositions
  Another transformation from QI to TM that
  (1) implies KY speedup and (2) enables online solution.
- Extensions
  Applying the technique to known generalizations of KY.

Background

Background

- Kunth-Yao Quadrangle Inequality Speedup

Background

- Kunth-Yao Quadrangle Inequality Speedup
  - D. E. Knuth (1971) and F. F. Yao (1980,1982)

Background

- Kunth-Yao Quadrangle Inequality Speedup
  - D. E. Knuth (1971) and F. F. Yao (1980,1982)
  - $\Theta(n)$ speedup: $O(n^3)$ down to $O(n^2)$

Background

- Kunth-Yao Quadrangle Inequality Speedup
  - D. E. Knuth (1971) and F. F. Yao (1980,1982)
  - $\Theta(n)$ speedup: $O(n^3)$ down to $O(n^2)$
- SMAWK Algorithm for finding
  Row Minima of Totally Monotone Matrices

Background

- Kunth-Yao Quadrangle Inequality Speedup
  - D. E. Knuth (1971) and F. F. Yao (1980,1982)
  - $\Theta(n)$ speedup: $O(n^3)$ down to $O(n^2)$
- SMAWK Algorithm for finding
  Row Minima of Totally Monotone Matrices
  - A. Aggarwal, M. M. Klawe, S. Moran, P. Shor, R. Wilber (1986)

Background

- Kunth-Yao Quadrangle Inequality Speedup
  - D. E. Knuth (1971) and F. F. Yao (1980,1982)
  - $\Theta(n)$ speedup: $O(n^3)$ down to $O(n^2)$
- SMAWK Algorithm for finding
  Row Minima of Totally Monotone Matrices
  - A. Aggarwal, M. M. Klawe, S. Moran, P. Shor, R. Wilber (1986)
  - $\Theta(n)$ speedup: $O(n^2)$ down to $O(n)$

Background

- Kunth-Yao Quadrangle Inequality Speedup
  - D. E. Knuth (1971) and F. F. Yao (1980,1982)
  - $\Theta(n)$ speedup: $O(n^3)$ down to $O(n^2)$
- SMAWK Algorithm for finding
  Row Minima of Totally Monotone Matrices
  - A. Aggarwal, M. M. Klawe, S. Moran, P. Shor, R. Wilber (1986)
  - $\Theta(n)$ speedup: $O(n^2)$ down to $O(n)$
- Both techniques are often used to speed up DPs.

Background

- Kunth-Yao Quadrangle Inequality Speedup
  - D. E. Knuth (1971) and F. F. Yao (1980,1982)
  - $\Theta(n)$ speedup: $O(n^3)$ down to $O(n^2)$
- SMAWK Algorithm for finding
  Row Minima of Totally Monotone Matrices
  - A. Aggarwal, M. M. Klawe, S. Moran, P. Shor, R. Wilber (1986)
  - $\Theta(n)$ speedup: $O(n^2)$ down to $O(n)$
- Both techniques are often used to speed up DPs.
- $\boxed{\text{How are the two techniques related?}}$

Quadrangle Inequality

Quadrangle Inequality

- Original Motivation

  Computing Optimal Binary Search Trees (Optimal BST)

  [Gilbert and Moore (1959)]

Quadrangle Inequality

- Original Motivation
  Computing Optimal Binary Search Trees (Optimal BST)
  [Gilbert and Moore (1959)]
- Optimal BST
  - Construct a search tree for n keys

Quadrangle Inequality

- Original Motivation

  Computing Optimal Binary Search Trees (Optimal BST)

  [Gilbert and Moore (1959)]

- Optimal BST
  - Construct a search tree for n keys
  - n internal nodes corresponds to successful search

    p_l, $(l = 1 \ldots n)$ is the weight that search-key $=$ Key$_l$

Quadrangle Inequality

- Original Motivation
  Computing Optimal Binary Search Trees (Optimal BST)
  [Gilbert and Moore (1959)]
- Optimal BST
  - Construct a search tree for n keys
  - n internal nodes corresponds to successful search
    p_l, $(l = 1 \ldots n)$ is the weight that search-key $=$ Key$_l$
  - $n + 1$ external nodes corresponds to unsuccessful search
    q_l, $(l = 0 \ldots n)$ is the weight that Key$_l$ $<$ search-key $<$ Key$_{l+1}$

Quadrangle Inequality

- Original Motivation
  Computing Optimal Binary Search Trees (Optimal BST)
  [Gilbert and Moore (1959)]
- Optimal BST
  - Construct a search tree for n keys
  - n internal nodes corresponds to successful search
    p_l, $(l = 1 \dots n)$ is the weight that search-key $=$ Key$_l$
  - $n + 1$ external nodes corresponds to unsuccessful search
    q_l, $(l = 0 \dots n)$ is the weight that Key$_l$ $<$ search-key $<$ Key$_{l+1}$
  - Minimize the number of comparisons

$$\sum_{1 \le l \le n} p_l \cdot (1 + \underbrace{d(p_l)}_{\text{depth}}) + \sum_{0 \le l \le n} q_l \cdot \underbrace{d(q_l)}_{\text{depth}}$$

Optimal BST

- Minimize $\sum_{1 \le l \le n} p_l \cdot (1 + d(p_l)) + \sum_{0 \le l \le n} q_l \cdot d(q_l)$

Optimal BST

- Minimize $\sum_{1 \le l \le n} p_l \cdot (1 + d(p_l)) + \sum_{0 \le l \le n} q_l \cdot d(q_l)$
- An example

Optimal BST

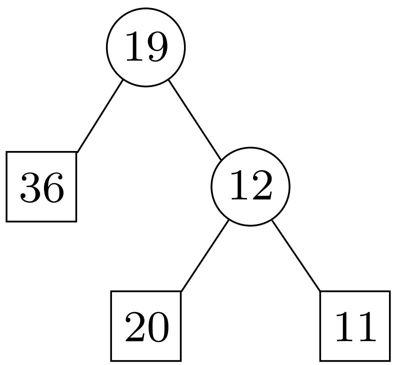
- Minimize $\sum_{1 \le l \le n} p_l \cdot (1 + d(p_l)) + \sum_{0 \le l \le n} q_l \cdot d(q_l)$
- An example

  $n = 2 \qquad p = (19, 12), \quad q = (36, 20, 11)$

Optimal BST

- Minimize $\sum_{1 \le l \le n} p_l \cdot (1 + d(p_l)) + \sum_{0 \le l \le n} q_l \cdot d(q_l)$
- An example

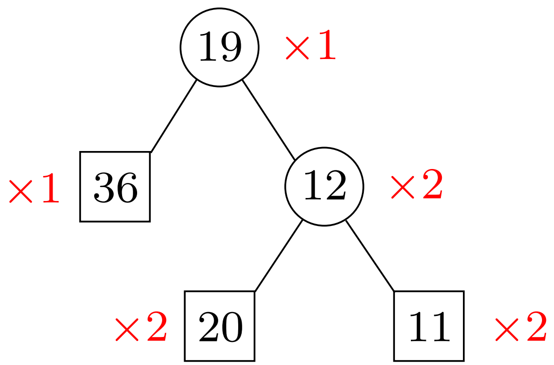
  $n = 2 \qquad p = (19, 12), \quad q = (36, 20, 11)$



Optimal BST

- Minimize $\sum_{1 \le l \le n} p_l \cdot (1 + d(p_l)) + \sum_{0 \le l \le n} q_l \cdot d(q_l)$
- An example

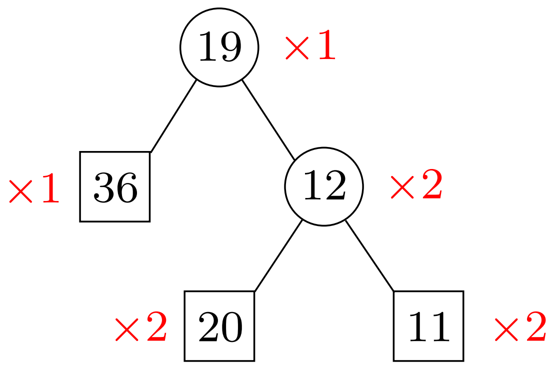
  $n = 2 \qquad p = (19, 12), \quad q = (36, 20, 11)$



Optimal BST

- Minimize $\sum_{1 \le l \le n} p_l \cdot (1 + d(p_l)) + \sum_{0 \le l \le n} q_l \cdot d(q_l)$
- An example

  $n = 2 \qquad p = (19, 12), \quad q = (36, 20, 11)$

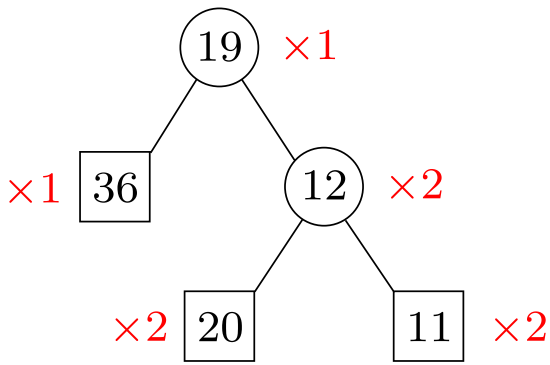


$\text{Cost} = 141$

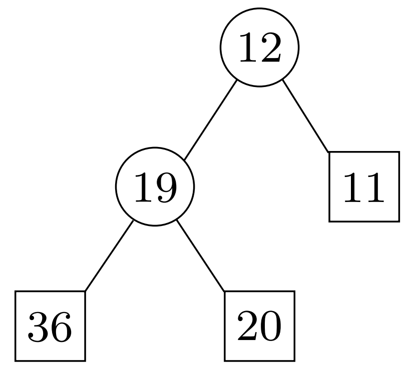
Optimal BST

- Minimize $\sum_{1 \le l \le n} p_l \cdot (1 + d(p_l)) + \sum_{0 \le l \le n} q_l \cdot d(q_l)$
- An example

  $n = 2 \qquad p = (19, 12), \quad q = (36, 20, 11)$



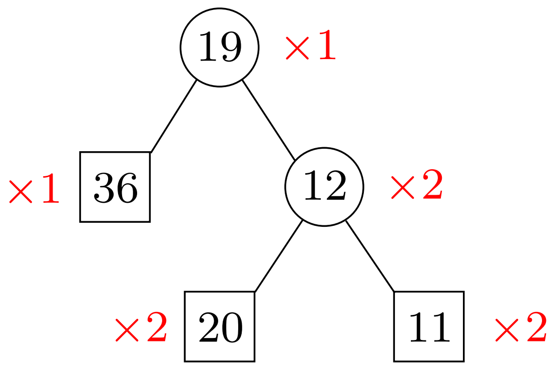
$\text{Cost} = 141$



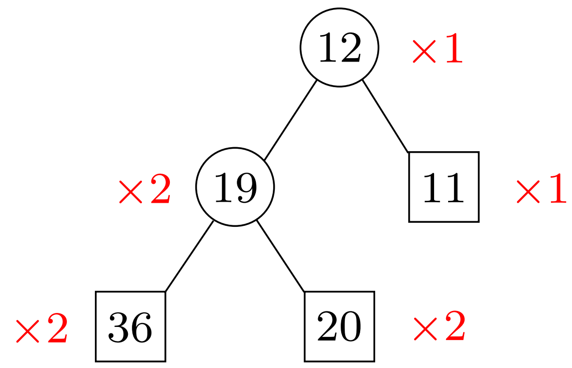
Optimal BST

- Minimize $\sum_{1 \le l \le n} p_l \cdot (1 + d(p_l)) + \sum_{0 \le l \le n} q_l \cdot d(q_l)$
- An example

  $n = 2 \qquad p = (19, 12), \quad q = (36, 20, 11)$



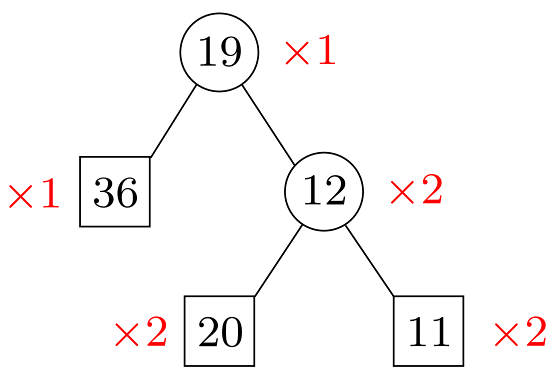
Cost $= 141$



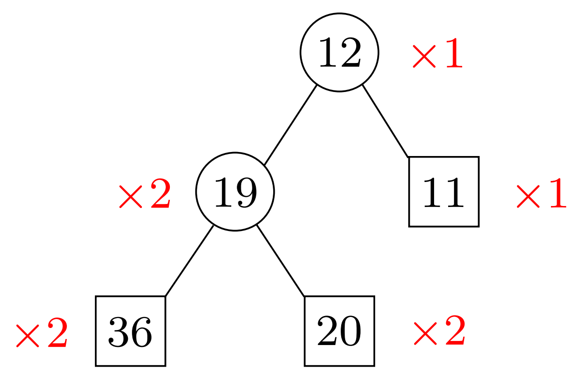
Optimal BST

- Minimize $\displaystyle\sum_{1\le l\le n} p_l \cdot (1 + d(p_l)) + \sum_{0\le l\le n} q_l \cdot d(q_l)$
- An example

  $n = 2 \qquad p = (19, 12), \quad q = (36, 20, 11)$



$\text{Cost} = 141$



$\text{Cost} = 173$

Optimal BST

- Solution: Dynamic Programming (DP)

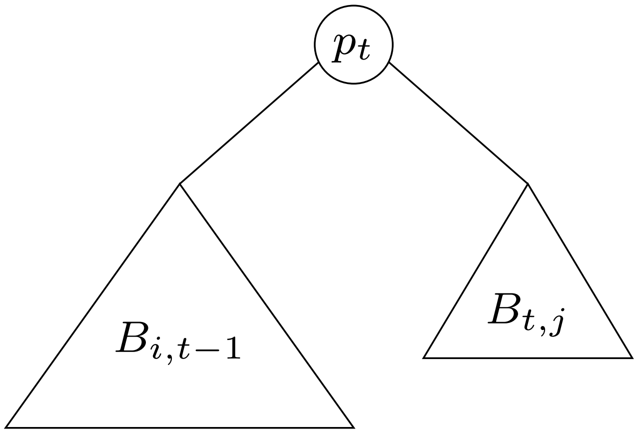
Optimal BST

- Solution: Dynamic Programming (DP)
  - $B_{i,j}$ the optimal BST for the subproblem $\text{Key}_{i+1}, \ldots, \text{Key}_j$

Optimal BST

- Solution: Dynamic Programming (DP)
  - $B_{i,j}$ the optimal BST for the subproblem $\text{Key}_{i+1}, \ldots, \text{Key}_j$
  - DP recurrence

$$B_{i,j} = \sum_{l=i+1}^{j} p_l + \sum_{l=i}^{j} q_l + \min_{i<t\leq j}\{B_{i,t-1} + B_{t,j}\}$$



Optimal BST

- DP: Straightforward Calculation

$$B_{i,j} = \sum_{l=i+1}^{j} p_l + \sum_{l=i}^{j} q_l + \min_{i<t\leq j}\{B_{i,t-1} + B_{t,j}\}$$

Optimal BST

- DP: Straightforward Calculation

$$B_{i,j} = \sum_{l=i+1}^{j} p_l + \sum_{l=i}^{j} q_l + \min_{i<t\le j}\{B_{i,t-1} + B_{t,j}\}$$

	0	1	2	3	4	5	6
0	0						
1		0					
2			0				
3				0			
4					0		
5						0	
6							0

$B_{i,j}$ depends on the entries to the left and below.

Optimal BST

- DP: Straightforward Calculation

$$B_{i,j} = \sum_{l=i+1}^{j} p_l + \sum_{l=i}^{j} q_l + \min_{i<t\leq j}\{B_{i,t-1} + B_{t,j}\}$$

- An example

$n = 6 \qquad p = (88, 21, 19, 12, 14, 18) \quad q = (53, 89, 36, 20, 11, 19, 15)$

Optimal BST

- DP: Straightforward Calculation

$$B_{i,j} = \sum_{l=i+1}^{j} p_l + \sum_{l=i}^{j} q_l + \min_{i<t\leq j}\{B_{i,t-1} + B_{t,j}\}$$

- An example

$n = 6$ $\quad p = (88, 21, 19, 12, 14, 18)$ $\quad q = (53, 89, 36, 20, 11, 19, 15)$

	0	1	2	3	4	5	6
0	0						
1		0					
2			0				
3				0			
4					0		
5						0	
6							0

Optimal BST

- DP: Straightforward Calculation

$$B_{i,j} = \sum_{l=i+1}^{j} p_l + \sum_{l=i}^{j} q_l + \min_{i<t\leq j}\{B_{i,t-1} + B_{t,j}\}$$

- An example

$n = 6$ $\quad p = (88, 21, 19, 12, 14, 18)$ $\quad q = (53, 89, 36, 20, 11, 19, 15)$

	0	1	2	3	4	5	6
0	0	230					
1		0	146				
2			0	75			
3				0	43		
4					0	44	
5						0	52
6							0

Optimal BST

- DP: Straightforward Calculation

$$B_{i,j} = \sum_{l=i+1}^{j} p_l + \sum_{l=i}^{j} q_l + \min_{i<t\leq j}\{B_{i,t-1} + B_{t,j}\}$$

- An example

$n = 6 \qquad p = (88, 21, 19, 12, 14, 18) \quad q = (53, 89, 36, 20, 11, 19, 15)$

	0	1	2	3	4	5	6
0	0	230	433				
1		0	146	260			
2			0	75	141		
3				0	43	119	
4					0	44	121
5						0	52
6							0

Optimal BST

- DP: Straightforward Calculation

$$B_{i,j} = \sum_{l=i+1}^{j} p_l + \sum_{l=i}^{j} q_l + \min_{i<t\le j}\{B_{i,t-1} + B_{t,j}\}$$

- An example

$n = 6 \qquad p = (88, 21, 19, 12, 14, 18) \quad q = (53, 89, 36, 20, 11, 19, 15)$

	0	1	2	3	4	5	6
0	0	230	433	586			
1		0	146	260	349		
2			0	75	141	250	
3				0	43	119	204
4					0	44	121
5						0	52
6							0

Optimal BST

- DP: Straightforward Calculation

$$B_{i,j} = \sum_{l=i+1}^{j} p_l + \sum_{l=i}^{j} q_l + \min_{i<t\leq j}\{B_{i,t-1} + B_{t,j}\}$$

- An example

$n = 6$ $\quad p = (88, 21, 19, 12, 14, 18)$ $\quad q = (53, 89, 36, 20, 11, 19, 15)$

	0	1	2	3	4	5	6
0	0	230	433	586	698		
1		0	146	260	349	491	
2			0	75	141	250	357
3				0	43	119	204
4					0	44	121
5						0	52
6							0

Optimal BST

- DP: Straightforward Calculation

$$B_{i,j} = \sum_{l=i+1}^{j} p_l + \sum_{l=i}^{j} q_l + \min_{i<t\leq j}\{B_{i,t-1} + B_{t,j}\}$$

- An example

$n = 6$ $\quad p = (88, 21, 19, 12, 14, 18)$ $\quad q = (53, 89, 36, 20, 11, 19, 15)$

	0	1	2	3	4	5	6
0	0	230	433	586	698	862	
1		0	146	260	349	491	624
2			0	75	141	250	357
3				0	43	119	204
4					0	44	121
5						0	52
6							0

Optimal BST

- DP: Straightforward Calculation

$$B_{i,j} = \sum_{l=i+1}^{j} p_l + \sum_{l=i}^{j} q_l + \min_{i<t\leq j}\{B_{i,t-1} + B_{t,j}\}$$

- An example

$n = 6 \qquad p = (88, 21, 19, 12, 14, 18) \quad q = (53, 89, 36, 20, 11, 19, 15)$

	0	1	2	3	4	5	6
0	0	230	433	586	698	862	1002
1		0	146	260	349	491	624
2			0	75	141	250	357
3				0	43	119	204
4					0	44	121
5						0	52
6							0

Optimal BST

- DP: Straightforward Calculation

$$B_{i,j} = \sum_{l=i+1}^{j} p_l + \sum_{l=i}^{j} q_l + \min_{i<t\le j}\{B_{i,t-1} + B_{t,j}\}$$

- An example

$n = 6 \qquad p = (88, 21, 19, 12, 14, 18) \quad q = (53, 89, 36, 20, 11, 19, 15)$

	0	1	2	3	4	5	6
0	0	230	433	586	698	862	1002
1		0	146	260	349	491	624
2			0	75	141	250	357
3				0	43	119	204
4					0	44	121
5						0	52
6							0

Optimal BST

- Naive: $O(n^3) = \sum_{i=1}^{n} \sum_{j=i}^{n} \Theta(j-i)$

$$B_{i,j} = \sum_{l=i+1}^{j} p_l + \sum_{l=i}^{j} q_l + \min_{i<t\leq j}\{B_{i,t-1} + B_{t,j}\}$$

Optimal BST

- Naive: $O(n^3) = \sum_{i=1}^{n} \sum_{j=i}^{n} \Theta(j-i)$

$$B_{i,j} = \sum_{l=i+1}^{j} p_l + \sum_{l=i}^{j} q_l + \min_{i<t\leq j}\{B_{i,t-1} + B_{t,j}\}$$

- Speedup: $O(n^2)$ [Knuth (1971)]

Optimal BST

- Naive: $O(n^3) = \sum_{i=1}^{n} \sum_{j=i}^{n} \Theta(j-i)$

$$B_{i,j} = \sum_{l=i+1}^{j} p_l + \sum_{l=i}^{j} q_l + \min_{i<t\leq j}\{B_{i,t-1} + B_{t,j}\}$$

- Speedup: $O(n^2)$ [Knuth (1971)]
  - $K_B(i,j)$ the largest index t that achieves the minimum.

Optimal BST

- Naive: $O(n^3) = \sum_{i=1}^{n} \sum_{j=i}^{n} \Theta(j-i)$

$$B_{i,j} = \sum_{l=i+1}^{j} p_l + \sum_{l=i}^{j} q_l + \min_{i<t\leq j}\{B_{i,t-1} + B_{t,j}\}$$

- Speedup: $O(n^2)$ [Knuth (1971)]
  - $K_B(i,j)$ the largest index t that achieves the minimum.
  - Theorem in [Knuth (1971)]

$$K_B(i,j) \leq K_B(i,j+1) \leq K_B(i+1,j+1)$$

Optimal BST

- Naive: $O(n^3) = \sum_{i=1}^{n} \sum_{j=i}^{n} \Theta(j-i)$

$$B_{i,j} = \sum_{l=i+1}^{j} p_l + \sum_{l=i}^{j} q_l + \min_{i<t\leq j}\{B_{i,t-1} + B_{t,j}\}$$

- Speedup: $O(n^2)$ [Knuth (1971)]
  - $K_B(i,j)$ the largest index t that achieves the minimum.
  - Theorem in [Knuth (1971)]

$$K_B(i,j) \leq K_B(i,j+1) \leq K_B(i+1,j+1)$$

	i	$i+1$
j	$K_B(i,j)$	$K_B(i,j+1)$
$j+1$		$K_B(i+1,j+1)$

Optimal BST

- Speedup: $B_{i,j} = \sum_{l=i+1}^{j} p_l + \sum_{l=i}^{j} q_l + \min_{i<t\leq j}\{B_{i,t-1} + B_{t,j}\}$

  $$K_B(i,j) \leq K_B(i,j+1) \leq K_B(i+1,j+1)$$

Optimal BST

- Speedup: $B_{i,j} = \sum_{l=i+1}^{j} p_l + \sum_{l=i}^{j} q_l + \min_{i<t\leq j}\{B_{i,t-1} + B_{t,j}\}$

$$K_B(i,j) \leq K_B(i,j+1) \leq K_B(i+1,j+1)$$

- The index table

Optimal BST

- Speedup: $B_{i,j} = \sum_{l=i+1}^{j} p_l + \sum_{l=i}^{j} q_l + \min_{i<t\leq j}\{B_{i,t-1} + B_{t,j}\}$

$$K_B(i,j) \leq K_B(i,j+1) \leq K_B(i+1,j+1)$$

- The index table

	0	1	2	3	4	5	6
0		0					
1			1				
2				2			
3					3		
4						4	
5							5
6							

Optimal BST

- Speedup: $B_{i,j} = \sum_{l=i+1}^{j} p_l + \sum_{l=i}^{j} q_l + \min_{i<t\leq j}\{B_{i,t-1} + B_{t,j}\}$

$$K_B(i,j) \leq K_B(i,j+1) \leq K_B(i+1,j+1)$$

- The index table

	0	1	2	3	4	5	6
0		0	0				
1			1				
2				2			
3					3		
4						4	
5							5
6							

Optimal BST

- Speedup: $B_{i,j} = \sum_{l=i+1}^{j} p_l + \sum_{l=i}^{j} q_l + \min_{i<t\leq j}\{B_{i,t-1} + B_{t,j}\}$

$$K_B(i,j) \leq K_B(i,j+1) \leq K_B(i+1,j+1)$$

- The index table

	0	1	2	3	4	5	6
0		0	0				
1			1	1			
2				2			
3					3		
4						4	
5							5
6							

Optimal BST

- Speedup: $B_{i,j} = \sum_{l=i+1}^{j} p_l + \sum_{l=i}^{j} q_l + \min_{i<t\leq j}\{B_{i,t-1} + B_{t,j}\}$

$$K_B(i,j) \leq K_B(i,j+1) \leq K_B(i+1,j+1)$$

- The index table

	0	1	2	3	4	5	6
0		0	0				
1			1	1			
2				2	2		
3					3		
4						4	
5							5
6							

Optimal BST

- Speedup: $B_{i,j} = \sum_{l=i+1}^{j} p_l + \sum_{l=i}^{j} q_l + \min_{i<t\le j}\{B_{i,t-1} + B_{t,j}\}$

$$K_B(i,j) \le K_B(i,j+1) \le K_B(i+1,j+1)$$

- The index table

	0	1	2	3	4	5	6
0		0	0				
1			1	1			
2				2	2		
3					3	4	
4						4	
5							5
6							

Optimal BST

- Speedup: $B_{i,j} = \sum_{l=i+1}^{j} p_l + \sum_{l=i}^{j} q_l + \min_{i<t\leq j}\{B_{i,t-1} + B_{t,j}\}$

$$K_B(i,j) \leq K_B(i,j+1) \leq K_B(i+1,j+1)$$

- The index table

	0	1	2	3	4	5	6
0		0	0				
1			1	1			
2				2	2		
3					3	4	
4						4	5
5							5
6							

Optimal BST

- Speedup: $B_{i,j} = \sum_{l=i+1}^{j} p_l + \sum_{l=i}^{j} q_l + \min_{i<t\leq j}\{B_{i,t-1} + B_{t,j}\}$

$$K_B(i,j) \leq K_B(i,j+1) \leq K_B(i+1,j+1)$$

- The index table

	0	1	2	3	4	5	6
0		0	0				
1			1	1			
2				2	2		
3					3	4	
4						4	5
5							5
6							

Optimal BST

- Speedup: $B_{i,j} = \sum_{l=i+1}^{j} p_l + \sum_{l=i}^{j} q_l + \min_{i<t\leq j}\{B_{i,t-1} + B_{t,j}\}$

$$K_B(i,j) \leq K_B(i,j+1) \leq K_B(i+1,j+1)$$

- The index table

	0	1	2	3	4	5	6
0		0	0	0			
1			1	1	1		
2				2	2	2	
3					3	4	4
4						4	5
5							5
6							

Optimal BST

- Speedup: $B_{i,j} = \sum_{l=i+1}^{j} p_l + \sum_{l=i}^{j} q_l + \min_{i<t\leq j}\{B_{i,t-1} + B_{t,j}\}$

$$K_B(i,j) \leq K_B(i,j+1) \leq K_B(i+1,j+1)$$

- The index table

	0	1	2	3	4	5	6
0		0	0	0	0		
1			1	1	1	1	
2				2	2	2	4
3					3	4	4
4						4	5
5							5
6							

Optimal BST

- Speedup: $B_{i,j} = \sum_{l=i+1}^{j} p_l + \sum_{l=i}^{j} q_l + \min_{i<t\leq j}\{B_{i,t-1} + B_{t,j}\}$

$$K_B(i,j) \leq K_B(i,j+1) \leq K_B(i+1,j+1)$$

- The index table

	0	1	2	3	4	5	6
0		0	0	0	0	1	
1			1	1	1	1	2
2				2	2	2	4
3					3	4	4
4						4	5
5							5
6							

Optimal BST

- Speedup: $B_{i,j} = \sum_{l=i+1}^{j} p_l + \sum_{l=i}^{j} q_l + \min_{i<t\leq j}\{B_{i,t-1} + B_{t,j}\}$

$$K_B(i,j) \leq K_B(i,j+1) \leq K_B(i+1,j+1)$$

- The index table

	0	1	2	3	4	5	6
0		0	0	0	0	1	1
1			1	1	1	1	2
2				2	2	2	4
3					3	4	4
4						4	5
5							5
6							

Optimal BST

- Speedup:
  - $K_B(i,j) \leq K_B(i,j+1) \leq K_B(i+1,j+1)$

Optimal BST

- Speedup:
  - $K_B(i,j) \leq K_B(i,j+1) \leq K_B(i+1,j+1)$
  - Each diagonal $j - i = d$

$$\begin{aligned} O(n) &= \sum_{i=1}^{n-d} \left(K_B(i+1,i+d) - K_B(i,i+d-1)\right) \\ &= K_B(n-d+1,n) - K_B(1,d) \end{aligned}$$

Optimal BST

- Speedup:
  - $K_B(i,j) \leq K_B(i,j+1) \leq K_B(i+1,j+1)$
  - Each diagonal $j - i = d$

$$\begin{aligned} O(n) &= \sum_{i=1}^{n-d} (K_B(i+1,i+d) - K_B(i,i+d-1)) \\ &= K_B(n-d+1,n) - K_B(1,d) \end{aligned}$$

  - $O(n^2)$ total work over all n diagonals.

Quadrangle Inequality

Quadrangle Inequality

- Definition [Yao (1980, 1982)]

Quadrangle Inequality

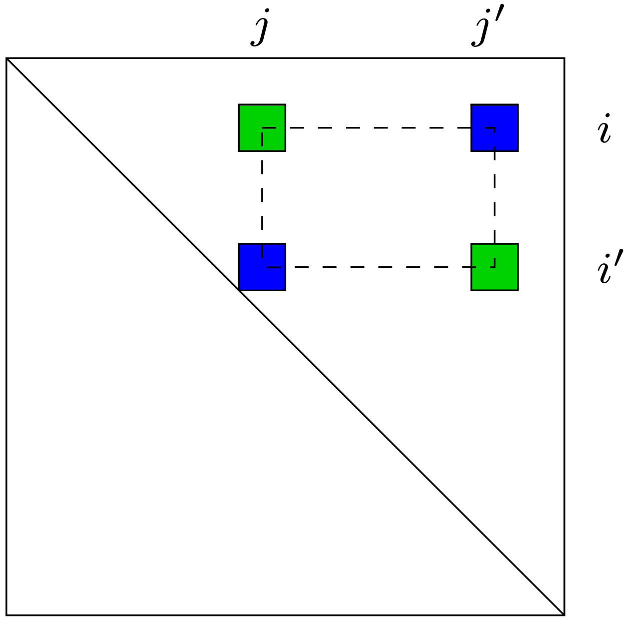
- Definition [Yao (1980, 1982)]
  - Function $f(i,j), (0 \le i \le j \le n)$
    satisfies a Quadrangle Inequality (QI), if $\forall i \le i' \le j \le j'$

$$f(i,j) + f(i',j') \le f(i',j) + f(i,j')$$

Quadrangle Inequality

- Definition [Yao (1980, 1982)]
  - Function $f(i,j), (0 \le i \le j \le n)$
    satisfies a Quadrangle Inequality (QI), if $\forall i \le i' \le j \le j'$

$$f(i,j) + f(i',j') \le f(i',j) + f(i,j')$$



Quadrangle Inequality

- Definition [Yao (1980, 1982)]
  - Function $f(i,j), (0 \le i \le j \le n)$
    satisfies a Quadrangle Inequality (QI), if $\forall i \le i' \le j \le j'$
    $$f(i,j) + f(i',j') \le f(i',j) + f(i,j')$$
  - Function $f(i,j), (0 \le i \le j \le n)$
    is Monotone over the integer lattice (MIL), if $\forall [i,j] \subseteq [i',j']$
    $$f(i,j) \le f(i',j')$$

Speedup using Quadrangle Inequality

$$B_{i,j} = w(i,j) + \min_{i<t\leq j}\{B_{i,t-1} + B_{t,j}\}$$

Speedup using Quadrangle Inequality

$$B_{i,j} = w(i,j) + \min_{i<t\leq j}\{B_{i,t-1} + B_{t,j}\}$$

- Lemmas from [Yao (1980)]

Speedup using Quadrangle Inequality

$$B_{i,j} = w(i,j) + \min_{i<t\leq j}\{B_{i,t-1} + B_{t,j}\}$$

- Lemmas from [Yao (1980)]
  - (A) If $w(i,j)$ satisfies QI and is MIL,
    $\Rightarrow B_{i,j}$ satisfies QI.

Speedup using Quadrangle Inequality

$$B_{i,j} = w(i,j) + \min_{i<t\leq j}\{B_{i,t-1} + B_{t,j}\}$$

- Lemmas from [Yao (1980)]
  - (A) If $w(i,j)$ satisfies QI and is MIL,
    $\Rightarrow B_{i,j}$ satisfies QI.
  - (B) If $B_{i,j}$ satisfies QI,
    $\Rightarrow K_B(i,j) \leq K_B(i,j+1) \leq K_B(i+1,j+1)$

Speedup using Quadrangle Inequality

$$B_{i,j} = w(i,j) + \min_{i<t\leq j}\{B_{i,t-1} + B_{t,j}\}$$

- Lemmas from [Yao (1980)]
  - (A) If $w(i,j)$ satisfies QI and is MIL,
    $\Rightarrow B_{i,j}$ satisfies QI.
  - (B) If $B_{i,j}$ satisfies QI,
    $\Rightarrow K_B(i,j) \leq K_B(i,j+1) \leq K_B(i+1,j+1)$

- In optimal BST problem,

$$B_{i,j} = \underbrace{\sum_{l=i+1}^{j} p_l + \sum_{l=i}^{j} q_l}_{w(i,j)} + \min_{i<t\leq j}\{B_{i,t-1} + B_{t,j}\}$$

Speedup using Quadrangle Inequality

$$B_{i,j} = w(i,j) + \min_{i<t\leq j}\{B_{i,t-1} + B_{t,j}\}$$

- Lemmas from [Yao (1980)]
  - (A) If $w(i,j)$ satisfies QI and is MIL,
    $\Rightarrow B_{i,j}$ satisfies QI.
  - (B) If $B_{i,j}$ satisfies QI,
    $\Rightarrow K_B(i,j) \leq K_B(i,j+1) \leq K_B(i+1,j+1)$

- In optimal BST problem,

$$B_{i,j} = \underbrace{\sum_{l=i+1}^{j} p_l + \sum_{l=i}^{j} q_l}_{w(i,j)} + \min_{i<t\leq j}\{B_{i,t-1} + B_{t,j}\}$$

  - Optimal BST $w(i,j)$ satisfies QI as equality and is MIL.

Speedup using Quadrangle Inequality

$$B_{i,j} = w(i,j) + \min_{i<t\leq j}\{B_{i,t-1} + B_{t,j}\}$$

- Lemmas from [Yao (1980)]
  - (A) If $w(i,j)$ satisfies QI and is MIL,
    $\Rightarrow B_{i,j}$ satisfies QI.
  - (B) If $B_{i,j}$ satisfies QI,
    $\Rightarrow K_B(i,j) \leq K_B(i,j+1) \leq K_B(i+1,j+1)$

- In optimal BST problem,

$$B_{i,j} = \underbrace{\sum_{l=i+1}^{j} p_l + \sum_{l=i}^{j} q_l}_{w(i,j)} + \min_{i<t\leq j}\{B_{i,t-1} + B_{t,j}\}$$

  - Optimal BST $w(i,j)$ satisfies QI as equality and is MIL.
  - $\Rightarrow$ exactly Knuth's result.

Online Problem

Online Problem

- Definition: Two-sided online problem

Online Problem

- Definition: Two-sided online problem
  - Current step: Optimal BST for $\text{Key}_{l+1}, \ldots, \text{Key}_r$

Online Problem

- Definition: Two-sided online problem
  - Current step: Optimal BST for $\text{Key}_{l+1}, \ldots, \text{Key}_r$
  - Next step: Add either Key_l or Key_{r+1}.

Online Problem

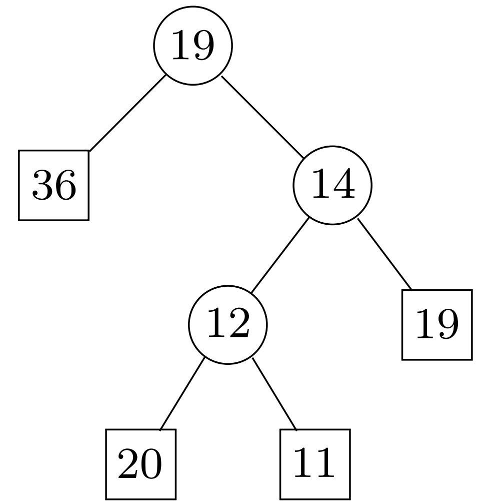
- Definition: Two-sided online problem
  - Current step: Optimal BST for $\text{Key}_{l+1}, \ldots, \text{Key}_r$
  - Next step: Add either Key_l or Key_{r+1}.
- An example

Online Problem

- Definition: Two-sided online problem
  - Current step: Optimal BST for $\text{Key}_{l+1}, \ldots, \text{Key}_r$
  - Next step: Add either Key_l or Key_{r+1}.
- An example

$p = (\quad 19, 12, 14 \quad) \quad q = (\quad 36, 20, 11, 19 \quad)$

	1	2	3	4	5	6
1						
2		0	75	141	250	
3			0	43	119	
4				0	44	
5					0	
6						

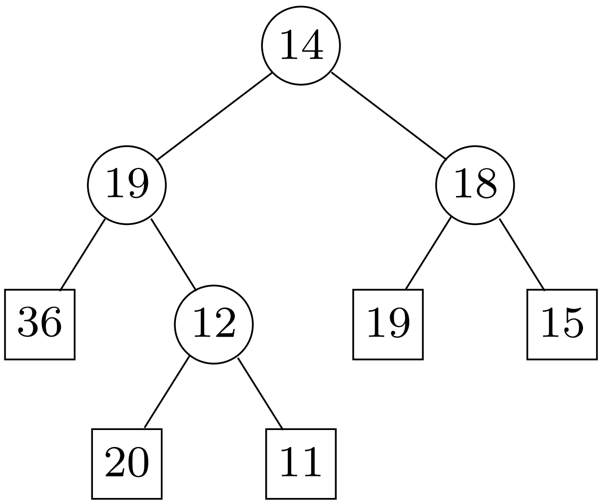


Online Problem

- Definition: Two-sided online problem
  - Current step: Optimal BST for $\text{Key}_{l+1}, \ldots, \text{Key}_r$
  - Next step: Add either Key_l or Key_{r+1}.
- An example

$p = (\quad 19, 12, 14, 18) \quad q = (\quad 36, 20, 11, 19, 15)$

	1	2	3	4	5	6
1						
2		0	75	141	250	357
3			0	43	119	204
4				0	44	121
5					0	52
6						0

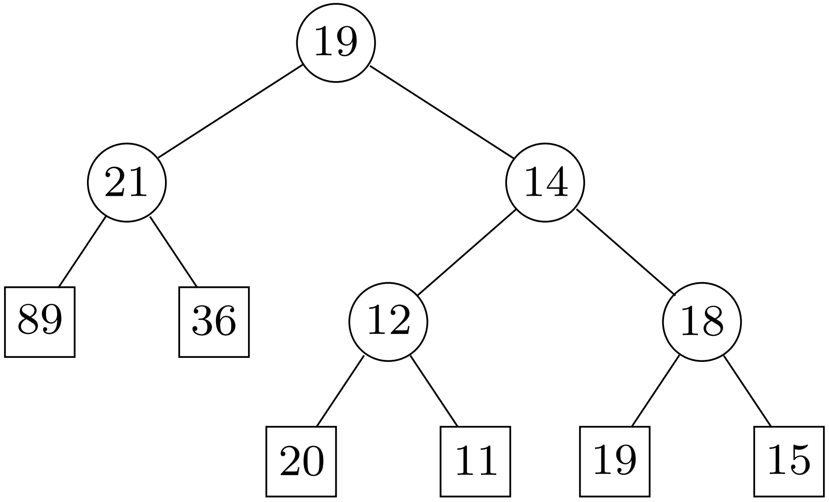


Online Problem

- Definition: Two-sided online problem
  - Current step: Optimal BST for $\text{Key}_{l+1}, \ldots, \text{Key}_r$
  - Next step: Add either Key_l or Key_{r+1}.
- An example

$p = (\quad 21, 19, 12, 14, 18) \quad q = (\quad 89, 36, 20, 11, 19, 15)$

	1	2	3	4	5	6
1	0	146	260	349	491	624
2		0	75	141	250	357
3			0	43	119	204
4				0	44	121
5					0	52
6						0



Outline

- Background
  - Kunth-Yao (KY) Quadrangle Inequality (QI) Speedup
  - SMAWK Algorithm for finding
    Row Minima of Totally Monotone (TM) Matrices
- The D^d Decomposition
  A transformation from QI to TM such that
  SMAWK solves KY problem as quickly as KY.
- The L^m and R^m Decompositions
  Another transformation from QI to TM that
  (1) implies KY speedup and (2) enables online solution.
- Extensions
  Applying the technique to known generalizations of KY.

Totally Monotone Matrices

- Definition

Totally Monotone Matrices

- Definition M is an $m \times n$ matrix

Totally Monotone Matrices

- Definition M is an $m \times n$ matrix
  - $\text{RM}_M(i)$ is index of rightmost minimum item of row i of M.

Totally Monotone Matrices

- Definition M is an $m \times n$ matrix
  - $\text{RM}_M(i)$ is index of rightmost minimum item of row i of M.
  - M is Monotone if $\forall i \leq i'$, $\text{RM}_M(i) \leq \text{RM}_M(i')$.

Totally Monotone Matrices

- Definition M is an $m \times n$ matrix
  - $\text{RM}_M(i)$ is index of rightmost minimum item of row i of M.
  - M is Monotone if $\forall i \leq i'$, $\text{RM}_M(i) \leq \text{RM}_M(i')$.

7	2	4	3	8	9
5	1	5	1	6	5
7	1	2	0	3	1
9	4	5	1	3	2
8	4	5	3	4	3
9	6	7	5	6	5

$\text{RM}_M(1) = 2$

$\text{RM}_M(2) = 4$

$\text{RM}_M(3) = 4$

$\text{RM}_M(4) = 4$

$\text{RM}_M(5) = 6$

$\text{RM}_M(6) = 6$

Totally Monotone Matrices

- Definition (Cond.)
  - A 2×2 Monotone matrix

2	4
4	5

2	3
5	3

7	1
2	2

Totally Monotone Matrices

- Definition (Cond.)
  - A 2×2 Monotone matrix

2	4
4	5

2	3
5	3

7	1
2	2

  - An $m \times n$ matrix M is Totally Monotone (TM)
    if every 2×2 submatrix is Monotone.

    (submatrix: not necessarily contiguous in the original matrix)

Totally Monotone Matrices

- Definition (Cond.)
  - A 2×2 Monotone matrix

2	4
4	5

2	3
5	3

7	1
2	2

  - An $m \times n$ matrix M is Totally Monotone (TM)
    if every 2×2 submatrix is Monotone.

    (submatrix: not necessarily contiguous in the original matrix)

  - Property

    M is Totally Monotone $\Rightarrow M$ is Monotone

Totally Monotone Matrices

- Definition (Cond.)
  - A 2×2 Monotone matrix

2	4
4	5

2	3
5	3

7	1
2	2

  - An $m \times n$ matrix M is Totally Monotone (TM)

    if every 2×2 submatrix is Monotone.

    (submatrix: not necessarily contiguous in the original matrix)

  - Property

    M is Totally Monotone $\Rightarrow$ M is Monotone

    M is Totally Monotone $\not\Leftarrow$ M is Monotone

SMAWK Algorithm

SMAWK Algorithm

- Motivation

  Find all m row minima of an implicitly given $m \times n$ matrix M

SMAWK Algorithm

- Motivation

  Find all m row minima of an implicitly given $m \times n$ matrix M

- Naive Algorithm: $O(mn)$

SMAWK Algorithm

- Motivation
  Find all m row minima of an implicitly given $m \times n$ matrix M
- Naive Algorithm: $O(mn)$
- SMAWK Algorithm
  [Aggarwal, Klawe, Moran, Shor, Wilber (1986)]

SMAWK Algorithm

- Motivation

  Find all m row minima of an implicitly given $m \times n$ matrix M

- Naive Algorithm: $O(mn)$

- SMAWK Algorithm

  [Aggarwal, Klawe, Moran, Shor, Wilber (1986)]

  - If M is Totally Monotone,

    all m row minima can be found in $O(m + n)$ time.

SMAWK Algorithm

- Motivation

  Find all m row minima of an implicitly given $m \times n$ matrix M

- Naive Algorithm: $O(mn)$
- SMAWK Algorithm

  [Aggarwal, Klawe, Moran, Shor, Wilber (1986)]

  - If M is Totally Monotone,

    all m row minima can be found in $O(m+n)$ time.

  - Usually $m = \Theta(n)$

    $\Theta(n)$ speedup: $O(n^2)$ down to $O(n)$.

SMAWK Algorithm

- Motivation

  Find all m row minima of an implicitly given $m \times n$ matrix M

- Naive Algorithm: $O(mn)$

- SMAWK Algorithm

  [Aggarwal, Klawe, Moran, Shor, Wilber (1986)]

  - If M is Totally Monotone,

    all m row minima can be found in $O(m+n)$ time.

  - Usually $m = \Theta(n)$

    $\Theta(n)$ speedup: $O(n^2)$ down to $O(n)$.

- SMAWK was culmination of decade(s) of work on similar problems; speedups using convexity and concavity.

SMAWK Algorithm

- Motivation

  Find all m row minima of an implicitly given $m \times n$ matrix M

- Naive Algorithm: $O(mn)$

- SMAWK Algorithm

  [Aggarwal, Klawe, Moran, Shor, Wilber (1986)]

  - If M is Totally Monotone,

    all m row minima can be found in $O(m+n)$ time.

  - Usually $m = \Theta(n)$

    $\Theta(n)$ speedup: $O(n^2)$ down to $O(n)$.

- SMAWK was culmination of decade(s) of work on similar problems; speedups using convexity and concavity.

- Has been used to speed up many DP problems, e.g., computational geometry, bioinformatics, k-center on a line, etc.

The Monge Property

The Monge Property

- Motivation

  TM property is often established via Monge property.

The Monge Property

- Motivation

  TM property is often established via Monge property.

- Definition

  An $m \times n$ matrix M is Monge if $\forall i \leq i'$ and $\forall j \leq j'$

$$M_{i,j} + M_{i',j'} \leq M_{i',j} + M_{i,j'}$$

The Monge Property

Quadrangle Inequality

Function $f(i,j)$

$\forall i \le i' \le j \le j'$

$f(i,j) + f(i',j') \le f(i',j) + f(i,j')$

Monge

Matrix M

$\forall i \le i'$ and $\forall j \le j'$

$M_{i,j} + M_{i',j'} \le M_{i',j} + M_{i,j'}$

The Monge Property

Quadrangle Inequality

Function $f(i,j)$

$\forall i \le i' \le j \le j'$

$f(i,j) + f(i',j') \le f(i',j) + f(i,j')$

Monge

Matrix M

$\forall i \le i'$ and $\forall j \le j'$

$M_{i,j} + M_{i',j'} \le M_{i',j} + M_{i,j'}$

- QI vs. Monge

The Monge Property

Quadrangle Inequality

Function $f(i,j)$

$\forall i \leq i' \leq j \leq j'$

$$f(i,j) + f(i',j') \leq f(i',j) + f(i,j')$$

Monge

Matrix M

$\forall i \leq i'$ and $\forall j \leq j'$

$$M_{i,j} + M_{i',j'} \leq M_{i',j} + M_{i,j'}$$

- QI vs. Monge
  - Different names for same type of inequality.

The Monge Property

Quadrangle Inequality

Function $f(i,j)$

$\forall i \leq i' \leq j \leq j'$

$$f(i,j) + f(i',j') \leq f(i',j) + f(i,j')$$

Monge

Matrix M

$\forall i \leq i'$ and $\forall j \leq j'$

$$M_{i,j} + M_{i',j'} \leq M_{i',j} + M_{i,j'}$$

- QI vs. Monge
  - Different names for same type of inequality.
  - Used differently in literature.

The Monge Property

Quadrangle Inequality

Function $f(i,j)$

$\forall i \leq i' \leq j \leq j'$

$f(i,j) + f(i',j') \leq f(i',j) + f(i,j')$

Monge

Matrix M

$\forall i \leq i'$ and $\forall j \leq j'$

$M_{i,j} + M_{i',j'} \leq M_{i',j} + M_{i,j'}$

- QI vs. Monge
  - Different names for same type of inequality.
  - Used differently in literature.
    - QI: $f(i,j)$ is function to be calculated.
    - Monge: $M_{i,j}$ implicitly given.

The Monge Property

Quadrangle Inequality

Function $f(i,j)$

$$\forall i \le i' \le j \le j'$$

$$f(i,j) + f(i',j') \le f(i',j) + f(i,j')$$

Monge

Matrix M

$\forall i \le i'$ and $\forall j \le j'$

$$M_{i,j} + M_{i',j'} \le M_{i',j} + M_{i,j'}$$

- QI vs. Monge
  - Different names for same type of inequality.
  - Used differently in literature.
    - QI: $f(i,j)$ is function to be calculated.
      Need all $f(i,j)$ entries.
    - Monge: $M_{i,j}$ implicitly given.
      Only need the row minima, but not other entries.

Monge Property

$$\forall i \le i' \quad \forall j \le j' \qquad M_{i,j} + M_{i',j'} \le M_{i',j} + M_{i,j'}$$

- Theorems

Monge Property

$$\forall i \leq i' \quad \forall j \leq j' \qquad M_{i,j} + M_{i',j'} \leq M_{i',j} + M_{i,j'}$$

- Theorems
  - M is Monge $\Rightarrow$ M is Totally Monotone
    M is Monge $\not\Leftarrow$ M is Totally Monotone

Monge Property

$$\forall i \leq i' \quad \forall j \leq j' \qquad M_{i,j} + M_{i',j'} \leq M_{i',j} + M_{i,j'}$$

- Theorems
  - M is Monge $\Rightarrow$ M is Totally Monotone
    M is Monge $\not\Leftarrow$ M is Totally Monotone
  - If $\forall i$ and $\forall j$, $M_{i,j} + M_{i+1,j+1} \leq M_{i+1,j} + M_{i,j+1}$,
    then M is Monge.

Monge Property

$$\forall i \le i' \quad \forall j \le j' \qquad M_{i,j} + M_{i',j'} \le M_{i',j} + M_{i,j'}$$

- Theorems
  - M is Monge $\Rightarrow$ M is Totally Monotone
    M is Monge $\not\Leftarrow$ M is Totally Monotone
  - If $\forall i$ and $\forall j$, $M_{i,j} + M_{i+1,j+1} \le M_{i+1,j} + M_{i,j+1}$,
    then M is Monge.
    - $\Rightarrow$ Only need to prove Monge property for adjacent rows and columns.

Monge Property

- General Scheme

Monge Property

- General Scheme
  1. Prove Monge Property for adjacent rows and columns

Monge Property

- General Scheme
  1. Prove Monge Property for adjacent rows and columns
  2. (Automatically implies) Monge Property

Monge Property

- General Scheme
  1. Prove Monge Property for adjacent rows and columns
  2. (Automatically implies) Monge Property
  3. (Automatically implies) Totally Monotone Property

Monge Property

- General Scheme
  1. Prove Monge Property for adjacent rows and columns
  2. (Automatically implies) Monge Property
  3. (Automatically implies) Totally Monotone Property
  4. Use SMAWK algorithm to find row minima

Monge Property

- General Scheme
  1. Prove Monge Property for adjacent rows and columns
  2. (Automatically implies) Monge Property
  3. (Automatically implies) Totally Monotone Property
  4. Use SMAWK algorithm to find row minima
  5. Usually $\Theta(n)$ speedup

Relationship?

Quadrangle Inequality Totally Monotone (Monge)

Relationship?

Quadrangle Inequality

A matrix to be calculated

Totally Monotone (Monge)

A matrix given implicitly

Relationship?

Quadrangle Inequality

A matrix to be calculated

Need all $O(n^2)$ entries

Totally Monotone (Monge)

A matrix given implicitly

Need only $O(n)$ row minima

Relationship?

Quadrangle Inequality

A matrix to be calculated

Need all $O(n^2)$ entries

$O(n^3)$ to $O(n^2)$ speedup

Totally Monotone (Monge)

A matrix given implicitly

Need only $O(n)$ row minima

$O(n^2)$ to $O(n)$ speedup

Relationship?

Quadrangle Inequality	Totally Monotone (Monge)
A matrix to be calculated	A matrix given implicitly
Need all $O(n^2)$ entries	Need only $O(n)$ row minima
$O(n^3)$ to $O(n^2)$ speedup	$O(n^2)$ to $O(n)$ speedup

- This talk

Relationship?

Quadrangle Inequality	Totally Monotone (Monge)
A matrix to be calculated	A matrix given implicitly
Need all $O(n^2)$ entries	Need only $O(n)$ row minima
$O(n^3)$ to $O(n^2)$ speedup	$O(n^2)$ to $O(n)$ speedup

- This talk
  - QI instance is decomposed into $\Theta(n)$ TM instances

Relationship?

Quadrangle Inequality	Totally Monotone (Monge)
A matrix to be calculated	A matrix given implicitly
Need all $O(n^2)$ entries	Need only $O(n)$ row minima
$O(n^3)$ to $O(n^2)$ speedup	$O(n^2)$ to $O(n)$ speedup

- This talk
  - QI instance is decomposed into $\Theta(n)$ TM instances
  - Each TM instance requires $O(n)$ time

Relationship?

Quadrangle Inequality	Totally Monotone (Monge)
A matrix to be calculated	A matrix given implicitly
Need all $O(n^2)$ entries	Need only $O(n)$ row minima
$O(n^3)$ to $O(n^2)$ speedup	$O(n^2)$ to $O(n)$ speedup

- This talk
  - QI instance is decomposed into $\Theta(n)$ TM instances
  - Each TM instance requires $O(n)$ time
  - $\Rightarrow$ QI instance requires $O(n^2)$ time in total

Decompositions

QI instance $\longrightarrow$ $\Theta(n)$ TM instances

Decompositions

QI instance $\longrightarrow \Theta(n)$ TM instances

- D^d decomposition

- L^m and R^m decompositions

Decompositions

QI instance $\longrightarrow$ $\Theta(n)$ TM instances

- D^d decomposition
  - Each diagonal $\longrightarrow$ TM instance

- L^m and R^m decompositions

Decompositions

QI instance $\longrightarrow$ $\Theta(n)$ TM instances

- D^d decomposition
  - Each diagonal $\longrightarrow$ TM instance
- L^m and R^m decompositions
  - L^m: Each row $\longrightarrow$ TM instance
  - R^m: Each column $\longrightarrow$ TM instance

Decompositions

QI instance $\longrightarrow$ $\Theta(n)$ TM instances

- D^d decomposition
  - Each diagonal $\longrightarrow$ TM instance
  - Permits solving QI problem directly using SMAWK.
    Same time bound as KY but different technique.
- L^m and R^m decompositions
  - L^m: Each row $\longrightarrow$ TM instance
  - R^m: Each column $\longrightarrow$ TM instance

Decompositions

QI instance $\longrightarrow$ $\Theta(n)$ TM instances

- D^d decomposition
  - Each diagonal $\longrightarrow$ TM instance
  - Permits solving QI problem directly using SMAWK.
    Same time bound as KY but different technique.
- L^m and R^m decompositions
  - L^m: Each row $\longrightarrow$ TM instance
  - R^m: Each column $\longrightarrow$ TM instance
  - Immediately implies the original KY speedup

Decompositions

QI instance $\longrightarrow$ $\Theta(n)$ TM instances

- D^d decomposition
  - Each diagonal $\longrightarrow$ TM instance
  - Permits solving QI problem directly using SMAWK.
    Same time bound as KY but different technique.

- L^m and R^m decompositions
  - L^m: Each row $\longrightarrow$ TM instance
  - R^m: Each column $\longrightarrow$ TM instance
  - Immediately implies the original KY speedup
  - Permits using algorithm of [Larmore, Schieber (1990)], to get "online" KY speedup.

D^d Decomposition

D^d Decomposition

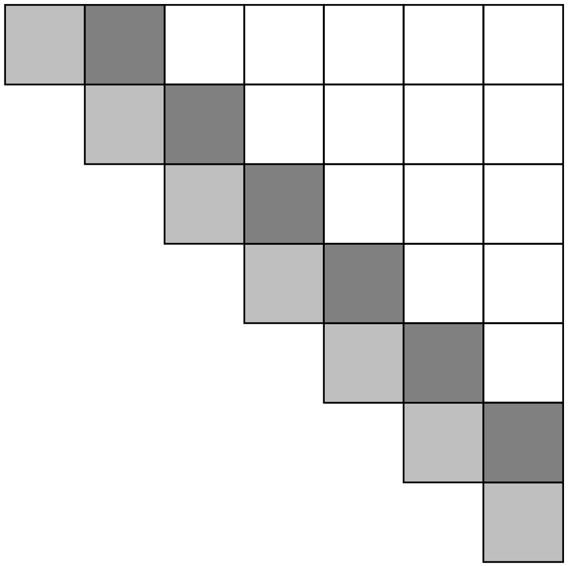
- Each diagonal d in original QI matrix
  corresponds to a new Monge Matrix D^d

D^d Decomposition

- Each diagonal d in original QI matrix
  corresponds to a new Monge Matrix D^d
- Entries on diagonal d are
  row minima of corresponding rows in D^d.

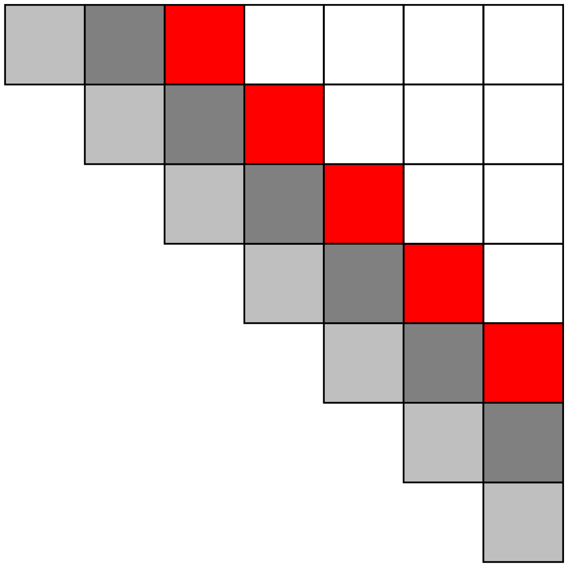
D^d Decomposition

- Each diagonal d in original QI matrix
  corresponds to a new Monge Matrix D^d
- Entries on diagonal d are
  row minima of corresponding rows in D^d.



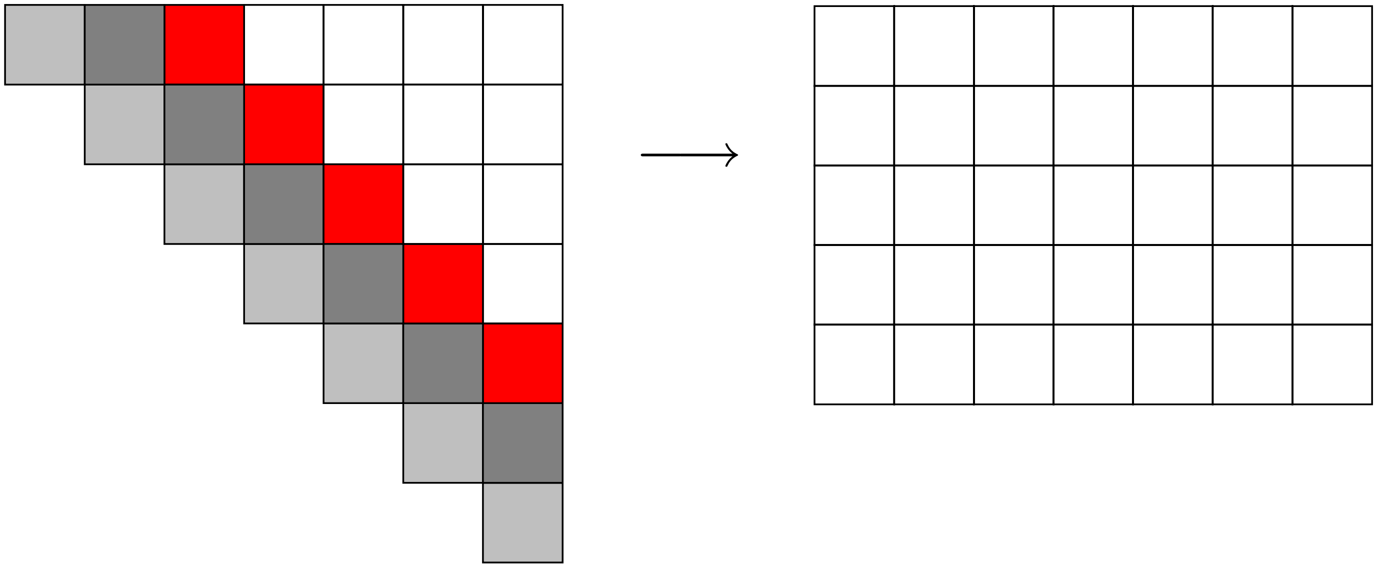
D^d Decomposition

- Each diagonal d in original QI matrix
  corresponds to a new Monge Matrix D^d
- Entries on diagonal d are
  row minima of corresponding rows in D^d.



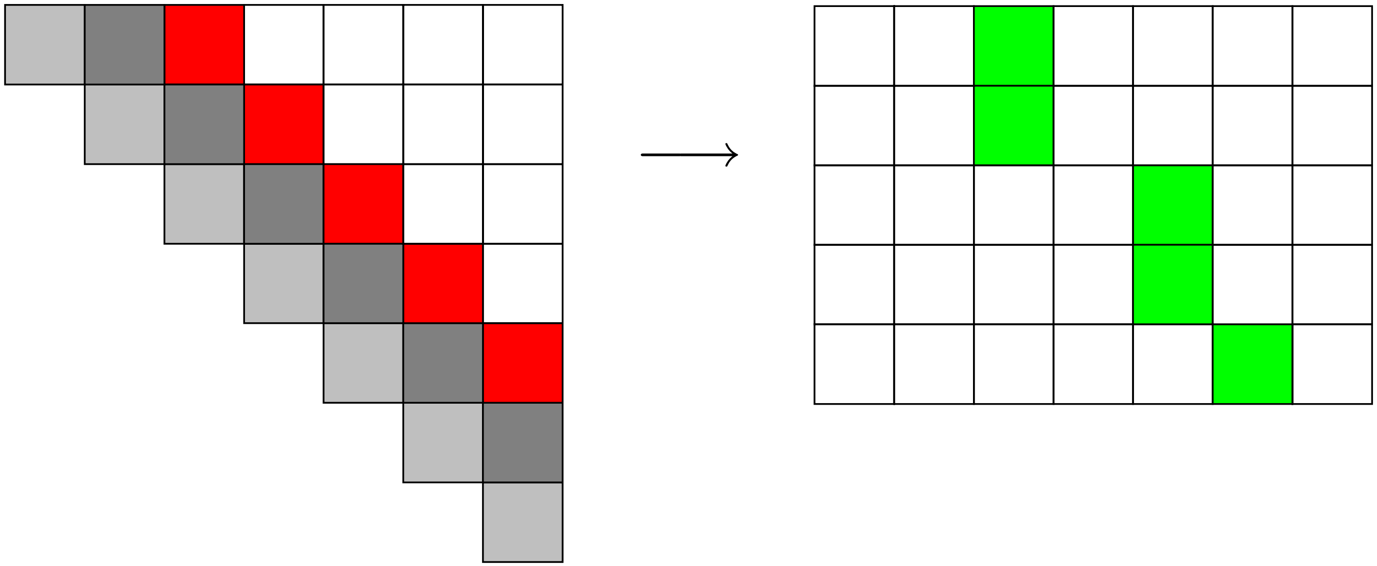
D^d Decomposition

- Each diagonal d in original QI matrix
  corresponds to a new Monge Matrix D^d
- Entries on diagonal d are
  row minima of corresponding rows in D^d.



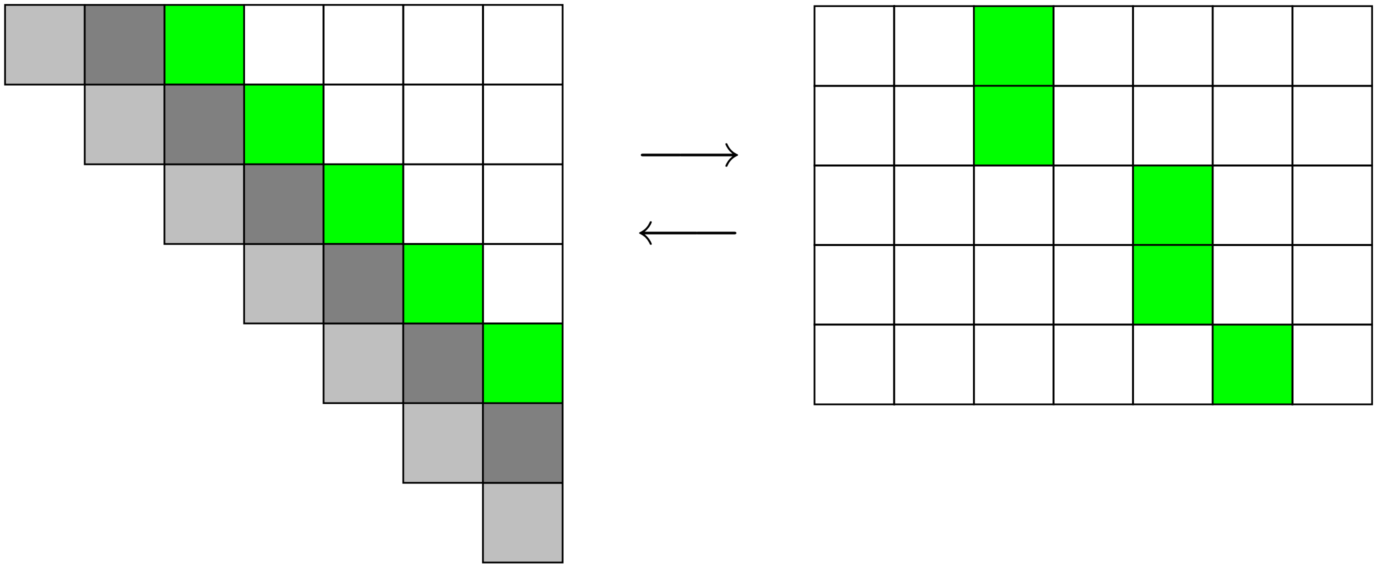
D^d Decomposition

- Each diagonal d in original QI matrix
  corresponds to a new Monge Matrix D^d
- Entries on diagonal d are
  row minima of corresponding rows in D^d.



D^d Decomposition

- Each diagonal d in original QI matrix
  corresponds to a new Monge Matrix D^d
- Entries on diagonal d are
  row minima of corresponding rows in D^d.



L^m and R^m Decompositions (R^m shown)

L^m and R^m Decompositions (R^m shown)

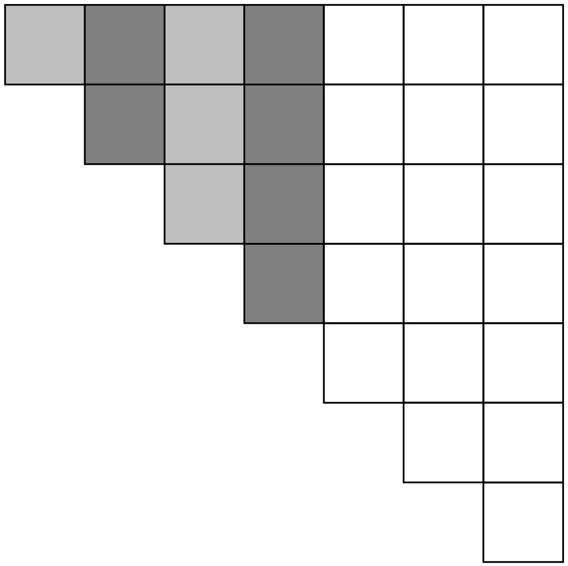
- Each column (row) m in original QI matrix
  corresponds to a new Monge Matrix R^m (L^m)

L^m and R^m Decompositions (R^m shown)

- Each column (row) m in original QI matrix
  corresponds to a new Monge Matrix R^m (L^m)
- Entries on column (row) m are
  row minima of corresponding rows in R^m (L^m).

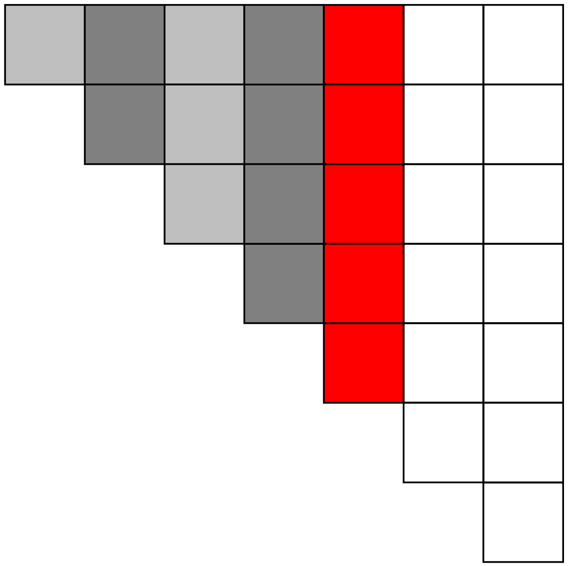
L^m and R^m Decompositions (R^m shown)

- Each column (row) m in original QI matrix
  corresponds to a new Monge Matrix R^m (L^m)
- Entries on column (row) m are
  row minima of corresponding rows in R^m (L^m).



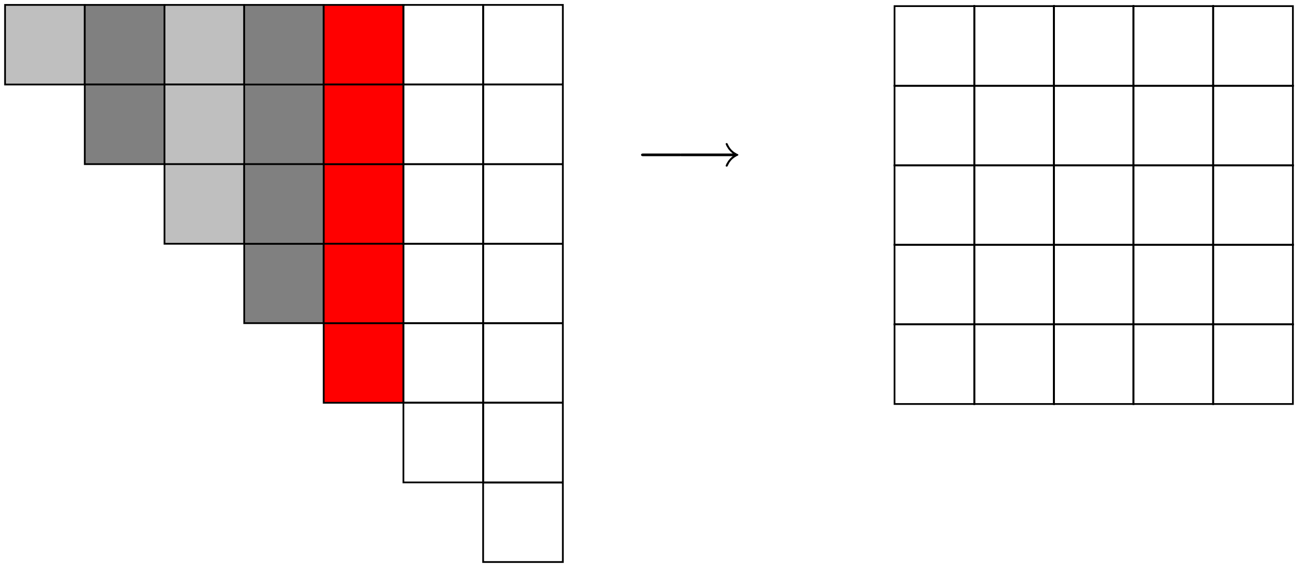
L^m and R^m Decompositions (R^m shown)

- Each column (row) m in original QI matrix
  corresponds to a new Monge Matrix R^m (L^m)
- Entries on column (row) m are
  row minima of corresponding rows in R^m (L^m).



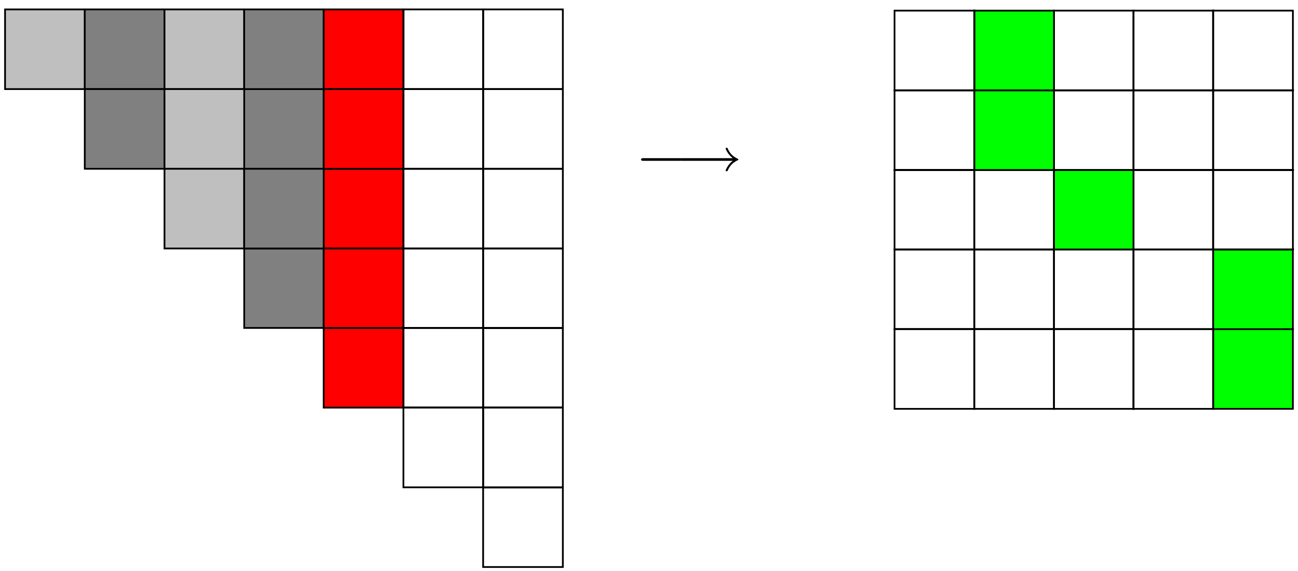
L^m and R^m Decompositions (R^m shown)

- Each column (row) m in original QI matrix
  corresponds to a new Monge Matrix R^m (L^m)
- Entries on column (row) m are
  row minima of corresponding rows in R^m (L^m).



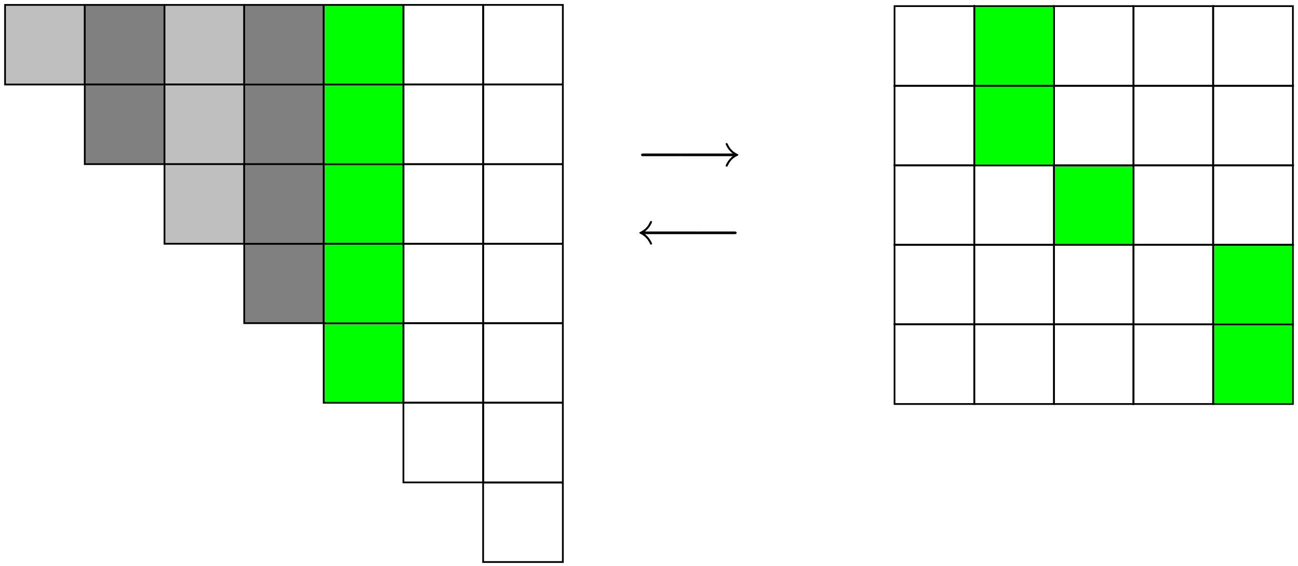
L^m and R^m Decompositions (R^m shown)

- Each column (row) m in original QI matrix
  corresponds to a new Monge Matrix R^m (L^m)
- Entries on column (row) m are
  row minima of corresponding rows in R^m (L^m).



L^m and R^m Decompositions (R^m shown)

- Each column (row) m in original QI matrix
  corresponds to a new Monge Matrix R^m (L^m)
- Entries on column (row) m are
  row minima of corresponding rows in R^m (L^m).



Outline

- Background
  - Kunth-Yao (KY) Quadrangle Inequality (QI) Speedup
  - SMAWK Algorithm for finding
    Row Minima of Totally Monotone (TM) Matrices
- The D^d Decomposition
  A transformation from QI to TM such that
  SMAWK solves KY problem as quickly as KY.
- The L^m and R^m Decompositions
  Another transformation from QI to TM that
  (1) implies KY speedup and (2) enables online solution.
- Extensions
  Applying the technique to known generalizations of KY.

D^d Decomposition

- Definition

D^d Decomposition

- Definition
  - General recurrence

    $B_{i,j} = w(i,j) + \min_{i<t\leq j}\{B_{i,t-1} + B_{t,j}\}$

D^d Decomposition

- Definition
  - General recurrence
    $B_{i,j} = w(i,j) + \min_{i<t\leq j}\{B_{i,t-1} + B_{t,j}\}$
  - For diagonal d, $(1 \leq d < n)$
    $B_{i,i+d} = w(i,i+d) + \min_{i<j\leq i+d}\{B_{i,j-1} + B_{j,i+d}\}$

D^d Decomposition

- Definition
  - General recurrence

    $$B_{i,j} = w(i,j) + \min_{i<t\leq j}\{B_{i,t-1} + B_{t,j}\}$$

  - For diagonal d, $(1 \leq d < n)$

    $$B_{i,i+d} = w(i,i+d) + \min_{i<j\leq i+d}\{B_{i,j-1} + B_{j,i+d}\}$$

  - Define $(n-d+1) \times (n+1)$ matrix D^d

    $$D^d_{i,j} = \begin{cases} w(i,i+d) + \{B_{i,j-1} + B_{j,i+d}\} & \text{if } 0 \leq i < j \leq i+d \leq n \\ \infty & \text{otherwise} \end{cases}$$

D^d Decomposition

- Definition
  - General recurrence

    $$B_{i,j} = w(i,j) + \min_{i<t\leq j}\{B_{i,t-1} + B_{t,j}\}$$

  - For diagonal d, $(1 \leq d < n)$

    $$B_{i,i+d} = w(i,i+d) + \min_{i<j\leq i+d}\{B_{i,j-1} + B_{j,i+d}\}$$

  - Define $(n-d+1) \times (n+1)$ matrix D^d

    $$D^d_{i,j} = \begin{cases} w(i,i+d) + \{B_{i,j-1} + B_{j,i+d}\} & \text{if } 0 \leq i < j \leq i+d \leq n \\ \infty & \text{otherwise} \end{cases}$$

  - Then,

    $$B_{i,i+d} = \min_{i<j\leq i+d} D^d_{i,j} = \min_{0\leq j\leq n} D^d_{i,j}$$

D^d Decomposition

- Definition
  - General recurrence
    $$B_{i,j} = w(i,j) + \min_{i<t\le j}\{B_{i,t-1} + B_{t,j}\}$$
  - For diagonal d, $(1 \le d < n)$
    $$B_{i,i+d} = w(i,i+d) + \min_{i<j\le i+d}\{B_{i,j-1} + B_{j,i+d}\}$$
  - Define $(n-d+1)\times(n+1)$ matrix D^d
    $$D^d_{i,j} = \begin{cases} w(i,i+d) + \{B_{i,j-1} + B_{j,i+d}\} & \text{if } 0 \le i < j \le i+d \le n \\ \infty & \text{otherwise} \end{cases}$$
  - Then,
    $$B_{i,i+d} = \min_{i<j\le i+d} D^d_{i,j} = \min_{0\le j\le n} D^d_{i,j}$$
- Lemma
  - D^d is Monge, for each $1 \le d < n$.

D^d Decomposition

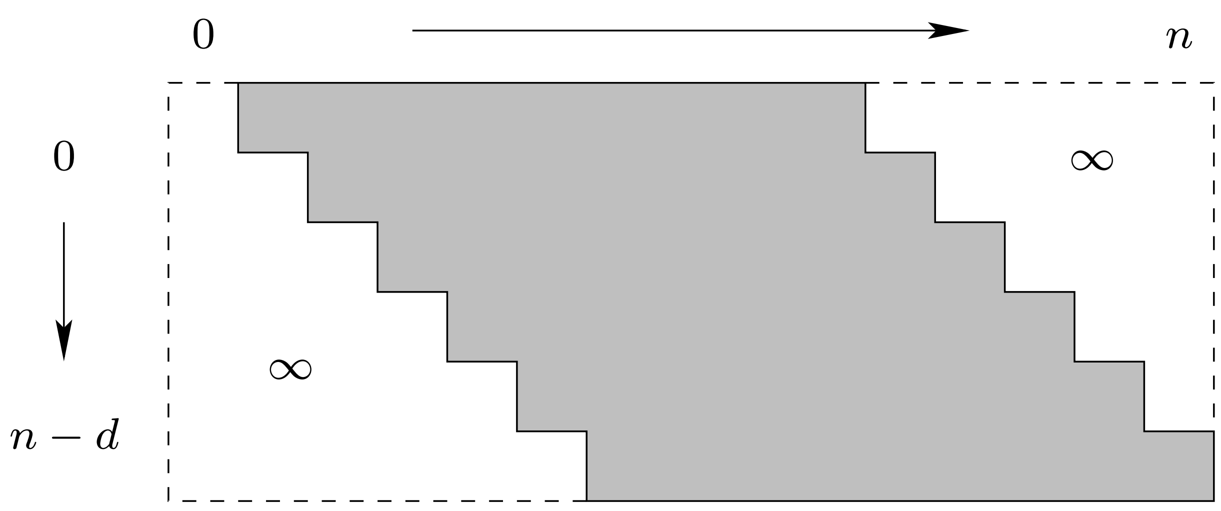
$$D^d_{i,j} = \begin{cases} w(i,i+d) + \{B_{i,j-1} + B_{j,i+d}\} & \text{if } 0 \le i < j \le i+d \le n \\ \infty & \text{otherwise} \end{cases}$$

- Shape of D^d

D^d Decomposition

$$D^d_{i,j} = \begin{cases} w(i, i+d) + \{B_{i,j-1} + B_{j,i+d}\} & \text{if } 0 \le i < j \le i+d \le n \\ \infty & \text{otherwise} \end{cases}$$

- Shape of D^d



D^d is Monge

D^d is Monge

Definition $\quad D^d_{i,j} = w(i, i+d) + \{B_{i,j-1} + B_{j,i+d}\}$

D^d is Monge

Definition $\quad D^d_{i,j} = w(i, i+d) + \{B_{i,j-1} + B_{j,i+d}\}$

Goal

$$D^d_{i,j} + D^d_{i+1,j+1} \;\leq\; D^d_{i+1,j} + D^d_{i,j+1}$$

D^d is Monge

Definition $D^d_{i,j} = w(i, i + d) + \{B_{i,j-1} + B_{j,i+d}\}$

By definition

$$D^d_{i,j} + D^d_{i+1,j+1} = \{w(i, i + d) + w(i + 1, i + d + 1)\}+$$
$$\{B_{i,j-1} + B_{i+1,j}\} + \{B_{j,i+d} + B_{j+1,i+d+1}\}$$

$$D^d_{i+1,j} + D^d_{i,j+1} = \{w(i + 1, i + d + 1) + w(i, i + d)\}+$$
$$\{B_{i+1,j-1} + B_{i,j}\} + \{B_{j,i+d+1} + B_{j+1,i+d}\}$$

Goal

$$D^d_{i,j} + D^d_{i+1,j+1} \le D^d_{i+1,j} + D^d_{i,j+1}$$

D^d is Monge

Definition $\quad D^d_{i,j} = w(i, i+d) + \{B_{i,j-1} + B_{j,i+d}\}$

By definition

$$D^d_{i,j} + D^d_{i+1,j+1} = \{w(i, i+d) + w(i+1, i+d+1)\} + \{B_{i,j-1} + B_{i+1,j}\} + \{B_{j,i+d} + B_{j+1,i+d+1}\}$$

$$D^d_{i+1,j} + D^d_{i,j+1} = \{w(i+1, i+d+1) + w(i, i+d)\} + \{B_{i+1,j-1} + B_{i,j}\} + \{B_{j,i+d+1} + B_{j+1,i+d}\}$$

Since B satisfies QI,

$$B_{i,j-1} + B_{i+1,j} \leq B_{i+1,j-1} + B_{i,j}$$

$$B_{j,i+d} + B_{j+1,i+d+1} \leq B_{j,i+d+1} + B_{j+1,i+d}$$

Goal

$$D^d_{i,j} + D^d_{i+1,j+1} \leq D^d_{i+1,j} + D^d_{i,j+1}$$

D^d is Monge

Definition $\quad D^d_{i,j} = w(i, i+d) + \{B_{i,j-1} + B_{j,i+d}\}$

By definition

$$D^d_{i,j} + D^d_{i+1,j+1} = \{w(i,i+d) + w(i+1,i+d+1)\} + \{B_{i,j-1} + B_{i+1,j}\} + \{B_{j,i+d} + B_{j+1,i+d+1}\}$$

$$D^d_{i+1,j} + D^d_{i,j+1} = \{w(i+1,i+d+1) + w(i,i+d)\} + \{B_{i+1,j-1} + B_{i,j}\} + \{B_{j,i+d+1} + B_{j+1,i+d}\}$$

Since B satisfies QI,

$$B_{i,j-1} + B_{i+1,j} \leq B_{i+1,j-1} + B_{i,j}$$

$$B_{j,i+d} + B_{j+1,i+d+1} \leq B_{j,i+d+1} + B_{j+1,i+d}$$

So

$$D^d_{i,j} + D^d_{i+1,j+1} \ \leq \ D^d_{i+1,j} + D^d_{i,j+1}$$

SMAWK replaces KY

SMAWK replaces KY

- We know
  - $D^d_{i,j} = \begin{cases} w(i, i+d) + \{B_{i,j-1} + B_{j,i+d}\} & \text{if } 0 \le i < j \le i+d \le n \\ \infty & \text{otherwise} \end{cases}$
  - $B_{i,i+d} = \min_{0 \le j \le n} D^d_{i,j}$ = minimum of row i of D^d
  - D^d is Monge, for each $1 \le d < n$.

SMAWK replaces KY

- We know
  - $D^d_{i,j} = \begin{cases} w(i,i+d) + \{B_{i,j-1} + B_{j,i+d}\} & \text{if } 0 \le i < j \le i+d \le n \\ \infty & \text{otherwise} \end{cases}$
  - $B_{i,i+d} = \min_{0 \le j \le n} D^d_{i,j}$ = minimum of row i of D^d
  - D^d is Monge, for each $1 \le d < n$.
- For fixed d, SMAWK can be used to find all the $B_{i,i+d}$ (row minima of D^d) in $O(n)$ time.

SMAWK replaces KY

- We know
  - $D^d_{i,j} = \begin{cases} w(i,i+d) + \{B_{i,j-1} + B_{j,i+d}\} & \text{if } 0 \le i < j \le i+d \le n \\ \infty & \text{otherwise} \end{cases}$
  - $B_{i,i+d} = \min_{0 \le j \le n} D^d_{i,j}$ = minimum of row i of D^d
  - D^d is Monge, for each $1 \le d < n$.

- For fixed d, SMAWK can be used to find all the $B_{i,i+d}$ (row minima of D^d) in $O(n)$ time.

- $\Rightarrow O(n^2)$ time for all D^d.

SMAWK replaces KY

- We know
  - $D^d_{i,j} = \begin{cases} w(i,i+d) + \{B_{i,j-1} + B_{j,i+d}\} & \text{if } 0 \le i < j \le i+d \le n \\ \infty & \text{otherwise} \end{cases}$
  - $B_{i,i+d} = \min_{0 \le j \le n} D^d_{i,j}$ = minimum of row i of D^d
  - D^d is Monge, for each $1 \le d < n$.
- For fixed d, SMAWK can be used to find all the $B_{i,i+d}$ (row minima of D^d) in $O(n)$ time.
- $\Rightarrow O(n^2)$ time for all D^d.
- Note: Must run SMAWK on D^d in the order $d = 1, 2, 3, \ldots$

  Entries in D^d depend upon row minima of $D^{d'}$ where $d' < d$.

Outline

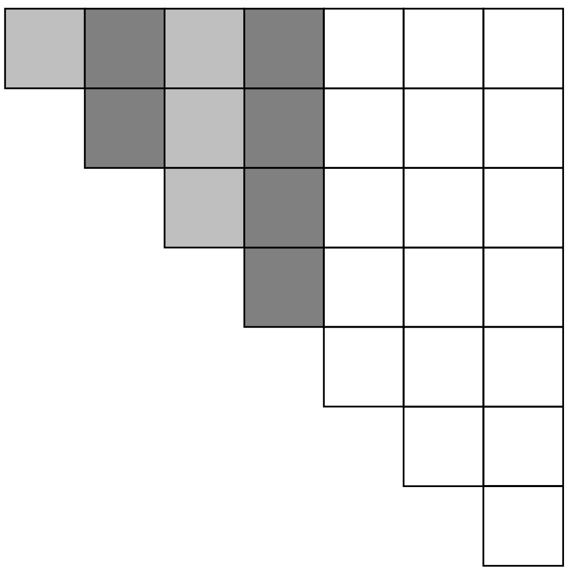
- Background
  - Kunth-Yao (KY) Quadrangle Inequality (QI) Speedup
  - SMAWK Algorithm for finding
    Row Minima of Totally Monotone (TM) Matrices
- The D^d Decomposition
  A transformation from QI to TM such that
  SMAWK solves KY problem as quickly as KY.
- The L^m and R^m Decompositions
  Another transformation from QI to TM that
  (1) implies KY speedup and (2) enables online solution.
- Extensions
  Applying the technique to known generalizations of KY.

R^m Decomposition

- R^m decomposition

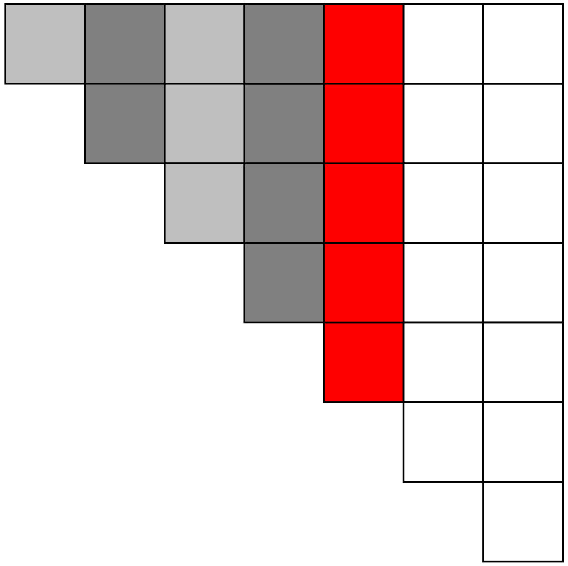
R^m Decomposition

- R^m decomposition



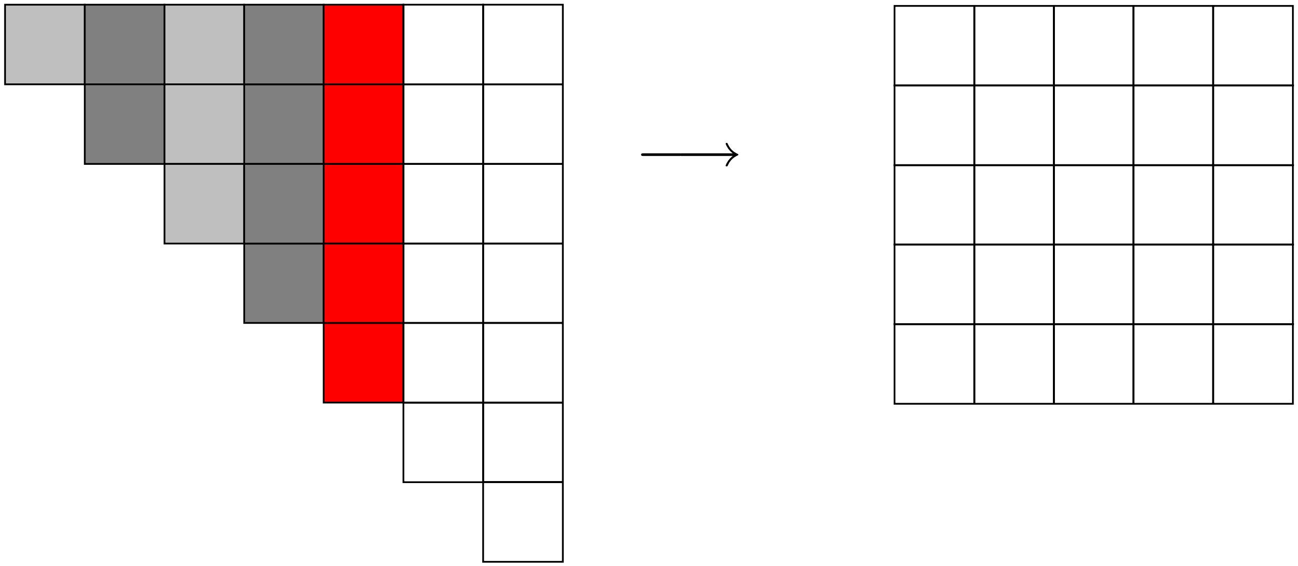
R^m Decomposition

- R^m decomposition



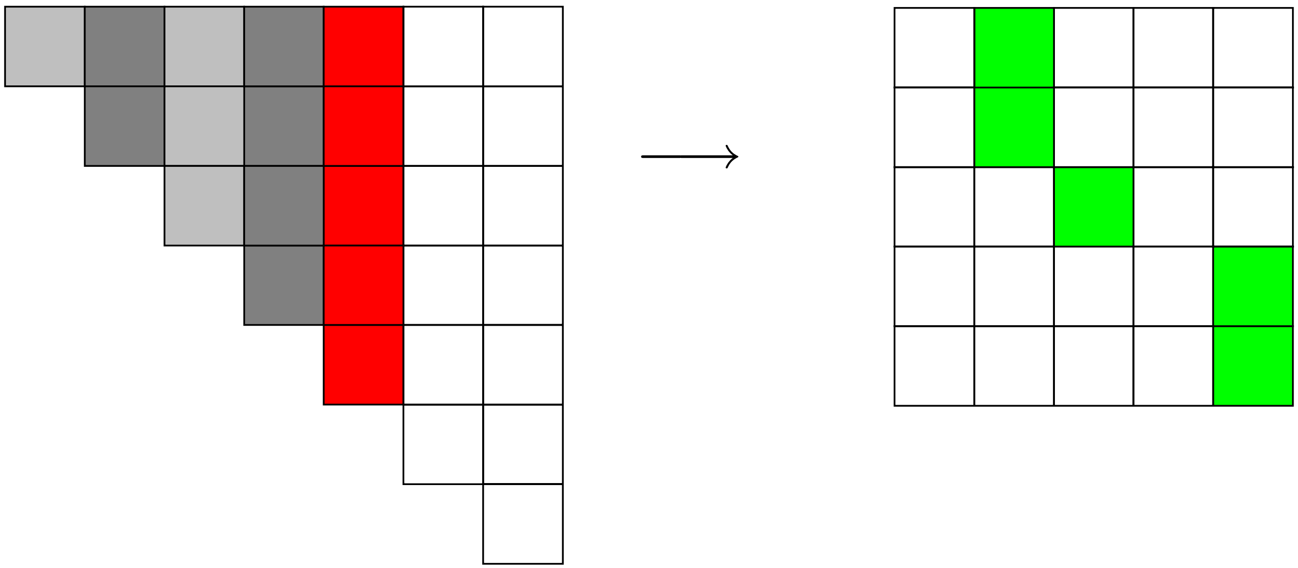
R^m Decomposition

- R^m decomposition



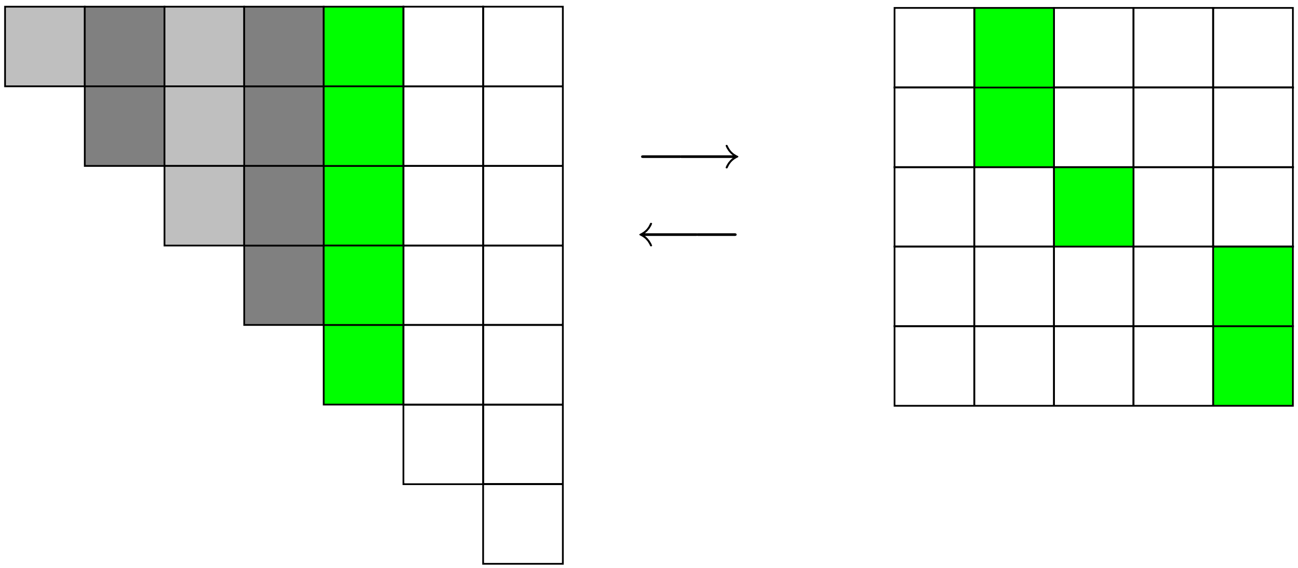
R^m Decomposition

- R^m decomposition



R^m Decomposition

- R^m decomposition



R^m Decomposition

- Definition

R^m Decomposition

- Definition
  - General recurrence

    $B_{i,j} = w(i,j) + \min_{i<t\leq j}\{B_{i,t-1} + B_{t,j}\}$

R^m Decomposition

- Definition
  - General recurrence

    $B_{i,j} = w(i,j) + \min_{i<t\leq j}\{B_{i,t-1} + B_{t,j}\}$
  - For column m, $(1 \leq m \leq n)$

    $B_{i,m} = w(i,m) + \min_{i<j\leq m}\{B_{i,j-1} + B_{j,m}\}$

R^m Decomposition

- Definition
  - General recurrence
    $$B_{i,j} = w(i,j) + \min_{i<t\leq j}\{B_{i,t-1} + B_{t,j}\}$$
  - For column m, $(1 \leq m \leq n)$
    $$B_{i,m} = w(i,m) + \min_{i<j\leq m}\{B_{i,j-1} + B_{j,m}\}$$
  - Define $(m+1) \times (m+1)$ matrix R^m
    $$R^m_{i,j} = \begin{cases} w(i,m) + \{B_{i,j-1} + B_{j,m}\} & \text{if } 0 \leq i < j \leq m \\ \infty & \text{otherwise} \end{cases}$$

R^m Decomposition

- Definition
  - General recurrence
    $$B_{i,j} = w(i,j) + \min_{i<t\leq j}\{B_{i,t-1} + B_{t,j}\}$$
  - For column m, $(1 \leq m \leq n)$
    $$B_{i,m} = w(i,m) + \min_{i<j\leq m}\{B_{i,j-1} + B_{j,m}\}$$
  - Define $(m+1)\times(m+1)$ matrix R^m
    $$R^m_{i,j} = \begin{cases} w(i,m) + \{B_{i,j-1} + B_{j,m}\} & \text{if } 0 \leq i < j \leq m \\ \infty & \text{otherwise} \end{cases}$$
  - Then
    $$B_{i,m} = \min_{i<j\leq m} R^m_{i,j} = \min_{0<j\leq m} R^m_{i,j}$$

R^m Decomposition

- Definition
  - General recurrence
    $$B_{i,j} = w(i,j) + \min_{i<t\le j}\{B_{i,t-1} + B_{t,j}\}$$
  - For column m, $(1 \le m \le n)$
    $$B_{i,m} = w(i,m) + \min_{i<j\le m}\{B_{i,j-1} + B_{j,m}\}$$
  - Define $(m+1)\times(m+1)$ matrix R^m
    $$R^m_{i,j} = \begin{cases} w(i,m) + \{B_{i,j-1} + B_{j,m}\} & \text{if } 0 \le i < j \le m \\ \infty & \text{otherwise} \end{cases}$$
  - Then
    $$B_{i,m} = \min_{i<j\le m} R^m_{i,j} = \min_{0<j\le m} R^m_{i,j}$$
- Lemma
  - R^m is Monge, for each $1 \le m \le n$.

R^m Decomposition

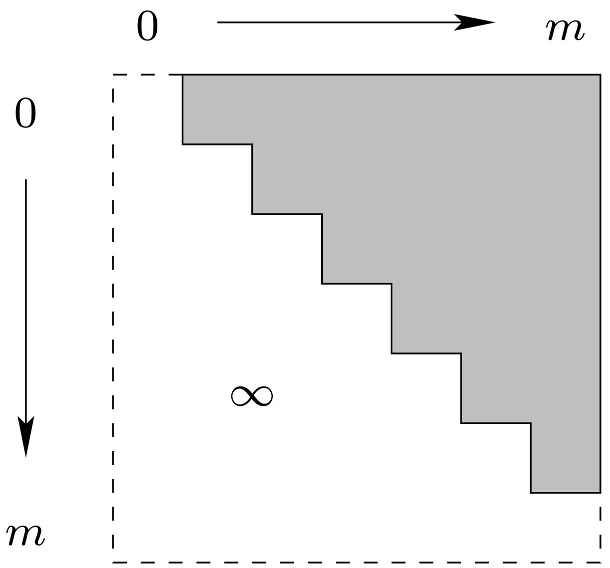
$$R^m_{i,j} = \begin{cases} w(i,m) + \{B_{i,j-1} + B_{j,m}\} & \text{if } 0 \le i < j \le m \\ \infty & \text{otherwise} \end{cases}$$

- Shape of R^m

R^m Decomposition

$$R^m_{i,j} = \begin{cases} w(i,m) + \{B_{i,j-1} + B_{j,m}\} & \text{if } 0 \leq i < j \leq m \\ \infty & \text{otherwise} \end{cases}$$

- Shape of R^m



R^m is Monge

R^m is Monge

Definition $\qquad R^m_{i,j} = w(i,m) + \{B_{i,j-1} + B_{j,m}\}$

R^m is Monge

Definition $\quad R^m_{i,j} = w(i,m) + \{B_{i,j-1} + B_{j,m}\}$

Goal

$$R^m_{i,j} + R^m_{i+1,j+1} \;\le\; R^m_{i+1,j} + R^m_{i,j+1}$$

R^m is Monge

Definition $\quad R^m_{i,j} = w(i,m) + \{B_{i,j-1} + B_{j,m}\}$

By definition

$$R^m_{i,j} + R^m_{i+1,j+1} = \{w(i,m) + w(i+1,m)\} + \{B_{i,j-1} + B_{i+1,j}\} + \{B_{j,m} + B_{j+1,m}\}$$

$$R^m_{i+1,j} + R^m_{i,j+1} = \{w(i+1,m) + w(i,m)\} + \{B_{i+1,j-1} + B_{i,j}\} + \{B_{j,m} + B_{j+1,m}\}$$

Goal

$$R^m_{i,j} + R^m_{i+1,j+1} \;\leq\; R^m_{i+1,j} + R^m_{i,j+1}$$

R^m is Monge

Definition $\quad R^m_{i,j} = w(i,m) + \{B_{i,j-1} + B_{j,m}\}$

By definition

$$R^m_{i,j} + R^m_{i+1,j+1} = \{w(i,m) + w(i+1,m)\} + \{B_{i,j-1} + B_{i+1,j}\} + \{B_{j,m} + B_{j+1,m}\}$$

$$R^m_{i+1,j} + R^m_{i,j+1} = \{w(i+1,m) + w(i,m)\} + \{B_{i+1,j-1} + B_{i,j}\} + \{B_{j,m} + B_{j+1,m}\}$$

Since B satisfies QI,

$$B_{i,j-1} + B_{i+1,j} \leq B_{i+1,j-1} + B_{i,j}$$

Goal

$$R^m_{i,j} + R^m_{i+1,j+1} \ \leq \ R^m_{i+1,j} + R^m_{i,j+1}$$

R^m is Monge

Definition $\quad R^m_{i,j} = w(i,m) + \{B_{i,j-1} + B_{j,m}\}$

By definition

$$R^m_{i,j} + R^m_{i+1,j+1} = \{w(i,m) + w(i+1,m)\} + \{B_{i,j-1} + B_{i+1,j}\} + \{B_{j,m} + B_{j+1,m}\}$$

$$R^m_{i+1,j} + R^m_{i,j+1} = \{w(i+1,m) + w(i,m)\} + \{B_{i+1,j-1} + B_{i,j}\} + \{B_{j,m} + B_{j+1,m}\}$$

Since B satisfies QI,

$$B_{i,j-1} + B_{i+1,j} \leq B_{i+1,j-1} + B_{i,j}$$

So

$$R^m_{i,j} + R^m_{i+1,j+1} \leq R^m_{i+1,j} + R^m_{i,j+1}$$

Outline

- Background
  - Kunth-Yao (KY) Quadrangle Inequality (QI) Speedup
  - SMAWK Algorithm for finding
    Row Minima of Totally Monotone (TM) Matrices
- The D^d Decomposition
  A transformation from QI to TM such that
  SMAWK solves KY problem as quickly as KY.
- The L^m and R^m Decompositions
  Another transformation from QI to TM that
  (1) implies KY speedup and (2) enables online solution.
- Extensions
  Applying the technique to known generalizations of KY.

L^m and R^m Imply Original KY Result

- KY Speedup
  - $K_B(i,j) \le K_B(i,j+1) \le K_B(i+1,j+1)$

L^m and R^m Imply Original KY Result

- KY Speedup
  - $K_B(i,j) \le K_B(i,j+1) \le K_B(i+1,j+1)$
- $R^m \longrightarrow K_B(i,j+1) \le K_B(i+1,j+1)$

L^m and R^m Imply Original KY Result

- KY Speedup
  - $K_B(i,j) \le K_B(i,j+1) \le K_B(i+1,j+1)$
- $R^m \longrightarrow K_B(i,j+1) \le K_B(i+1,j+1)$
  - Recall

    $\text{RM}_{R^m}(i)$ is index of rightmost minimum of row i of R^m.

1	1	2	2	2	2	$\text{RM}_M(1) = 2$
1	1	1	1	2	2	$\text{RM}_M(2) = 4$
1	1	1	1	2	2	$\text{RM}_M(3) = 4$
1	1	1	1	2	2	$\text{RM}_M(4) = 4$
1	1	1	1	1	1	$\text{RM}_M(5) = 6$
1	1	1	1	1	1	$\text{RM}_M(6) = 6$

L^m and R^m Imply Original KY Result

- KY Speedup
  - $K_B(i,j) \le K_B(i,j+1) \le K_B(i+1,j+1)$
- $R^m \longrightarrow K_B(i,j+1) \le K_B(i+1,j+1)$
  - Recall

    $\mathrm{RM}_{R^m}(i)$ is index of rightmost minimum of row i of R^m.
  - From the definition

    $$B_{i,m} = \min_{i<j\le m} R^m_{i,j} \quad \longrightarrow \quad K_B(i,m) = \mathrm{RM}_{R^m}(i)$$

L^m and R^m Imply Original KY Result

- KY Speedup
  - $K_B(i,j) \leq K_B(i,j+1) \leq K_B(i+1,j+1)$
- $R^m \longrightarrow K_B(i,j+1) \leq K_B(i+1,j+1)$
  - Recall

    $\text{RM}_{R^m}(i)$ is index of rightmost minimum of row i of R^m.
  - From the definition

    $$B_{i,m} = \min_{i<j\leq m} R^m_{i,j} \quad \longrightarrow \quad K_B(i,m) = \text{RM}_{R^m}(i)$$
  - So

    $$\begin{aligned} R^m \text{ is TM} \quad &\longrightarrow \quad \text{RM}_{R^m}(i) \leq \text{RM}_{R^m}(i+1) \\ &\longrightarrow \quad K_B(i,m) \leq K_B(i+1,m) \end{aligned}$$

L^m and R^m Imply Original KY Result

- KY Speedup
  - $K_B(i,j) \le K_B(i,j+1) \le K_B(i+1,j+1)$
- $R^m \longrightarrow K_B(i,j+1) \le K_B(i+1,j+1)$
  - Recall

    $\mathrm{RM}_{R^m}(i)$ is index of rightmost minimum of row i of R^m.
  - From the definition

    $$B_{i,m} = \min_{i<j\le m} R^m_{i,j} \quad \longrightarrow \quad K_B(i,m) = \mathrm{RM}_{R^m}(i)$$
  - So

    $$\begin{aligned} R^m \text{ is TM} \quad &\longrightarrow \quad \mathrm{RM}_{R^m}(i) \le \mathrm{RM}_{R^m}(i+1) \\ &\longrightarrow \quad K_B(i,m) \le K_B(i+1,m) \end{aligned}$$
- $L^m \longrightarrow K_B(i,j) \le K_B(i,j+1)$
  - Similar

Outline

- Background
  - Kunth-Yao (KY) Quadrangle Inequality (QI) Speedup
  - SMAWK Algorithm for finding
    Row Minima of Totally Monotone (TM) Matrices
- The D^d Decomposition
  A transformation from QI to TM such that
  SMAWK solves KY problem as quickly as KY.
- The L^m and R^m Decompositions
  Another transformation from QI to TM that
  (1) implies KY speedup and (2) enables online solution.
- Extensions
  Applying the technique to known generalizations of KY.

LARSCH Algorithm

LARSCH Algorithm

- D^d decomposition

LARSCH Algorithm

- D^d decomposition
  - $D^d_{i,j} = w(i, i+d) + \{B_{i,j-1} + B_{j,i+d}\} \qquad (0 \le i < j \le i+d \le n)$

LARSCH Algorithm

- D^d decomposition
  - $D^d_{i,j} = w(i, i+d) + \{B_{i,j-1} + B_{j,i+d}\} \qquad (0 \leq i < j \leq i+d \leq n)$
  - SMAWK algorithm

LARSCH Algorithm

- D^d decomposition
  - $D^d_{i,j} = w(i, i+d) + \{B_{i,j-1} + B_{j,i+d}\} \qquad (0 \leq i < j \leq i+d \leq n)$
  - SMAWK algorithm
- L^m and R^m decomposition

LARSCH Algorithm

- D^d decomposition
  - $D^d_{i,j} = w(i, i+d) + \{B_{i,j-1} + B_{j,i+d}\} \qquad (0 \leq i < j \leq i+d \leq n)$
  - SMAWK algorithm
- L^m and R^m decomposition
  - $R^m_{i,j} = w(i, m) + \{B_{i,j-1} + B_{j,m}\} \qquad (0 \leq i < j \leq m)$

LARSCH Algorithm

- D^d decomposition
  - $D^d_{i,j} = w(i, i+d) + \{B_{i,j-1} + B_{j,i+d}\} \qquad (0 \le i < j \le i+d \le n)$
  - SMAWK algorithm
- L^m and R^m decomposition
  - $R^m_{i,j} = w(i, m) + \{B_{i,j-1} + B_{j,m}\} \qquad (0 \le i < j \le m)$
  - Can not use SMAWK algorithm:

    $B_{j,m} = \min_t R^m_{j,t}$ is row-minima of row j of R^m
    and is therefore not known.

LARSCH Algorithm

- D^d decomposition
  - $D^d_{i,j} = w(i, i+d) + \{B_{i,j-1} + B_{j,i+d}\} \qquad (0 \le i < j \le i+d \le n)$
  - SMAWK algorithm
- L^m and R^m decomposition
  - $R^m_{i,j} = w(i, m) + \{B_{i,j-1} + B_{j,m}\} \qquad (0 \le i < j \le m)$
  - Can not use SMAWK algorithm:

    $B_{j,m} = \min_t R^m_{j,t}$ is row-minima of row j of R^m

    and is therefore not known.
  - LARSCH algorithm [Larmore, Schieber (1990)]

    permits calculating row minima of TM matrices in $O(n)$ time, even with this dependency.

LARSCH Algorithm

- D^d decomposition
  - $D^d_{i,j} = w(i, i+d) + \{B_{i,j-1} + B_{j,i+d}\} \qquad (0 \le i < j \le i+d \le n)$
  - SMAWK algorithm
- L^m and R^m decomposition
  - $R^m_{i,j} = w(i, m) + \{B_{i,j-1} + B_{j,m}\} \qquad (0 \le i < j \le m)$
  - Can not use SMAWK algorithm:

    $B_{j,m} = \min_t R^m_{j,t}$ is row-minima of row j of R^m
    and is therefore not known.
  - LARSCH algorithm [Larmore, Schieber (1990)]

    permits calculating row minima of TM matrices in $O(n)$ time, even with this dependency.
  - $O(n)$ time for each column $\Rightarrow O(n^2)$ in total.

LARSCH Algorithm

Finding row minima in totally monotone matrices with limited dependency.

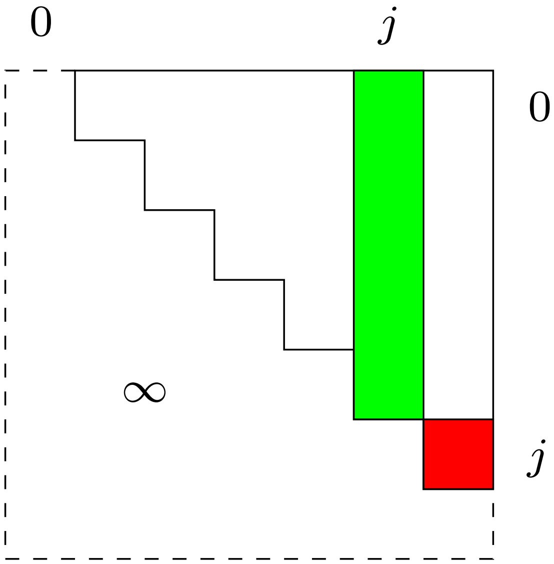
This is also known as online TM problem.

LARSCH Algorithm

Finding row minima in totally monotone matrices with limited dependency. This is also known as online TM problem.

Entries of column j can depend on the row minima of rows i where $M_{i,j} = \infty$.

Green: the column j.
Red: rows that column j can depend on.



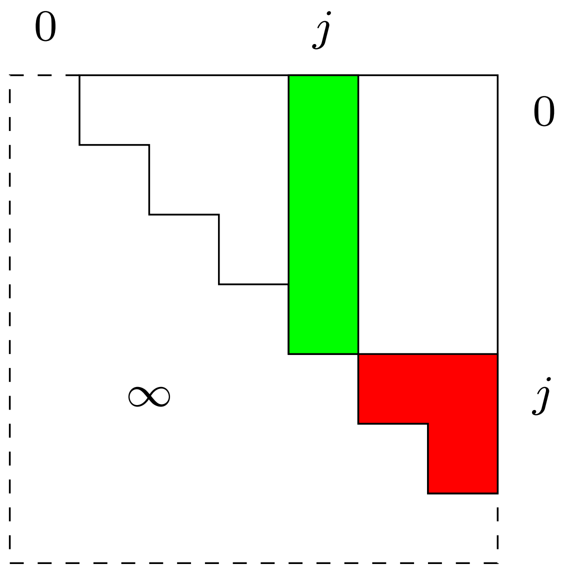
LARSCH Algorithm

Finding row minima in totally monotone matrices with limited dependency. This is also known as online TM problem.

Entries of column j can depend on the row minima of rows i where $M_{i,j} = \infty$.

Green: the column j.

Red: rows that column j can depend on.



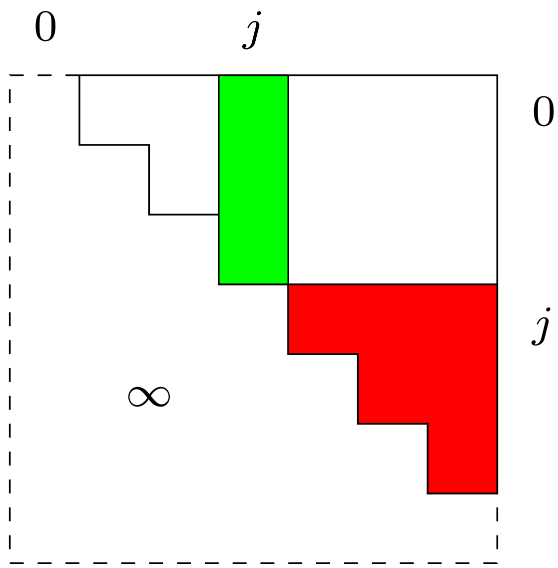
LARSCH Algorithm

Finding row minima in totally monotone matrices with limited dependency. This is also known as online TM problem.

Entries of column j can depend on the row minima of rows i where $M_{i,j} = \infty$.

Green: the column j.

Red: rows that column j can depend on.



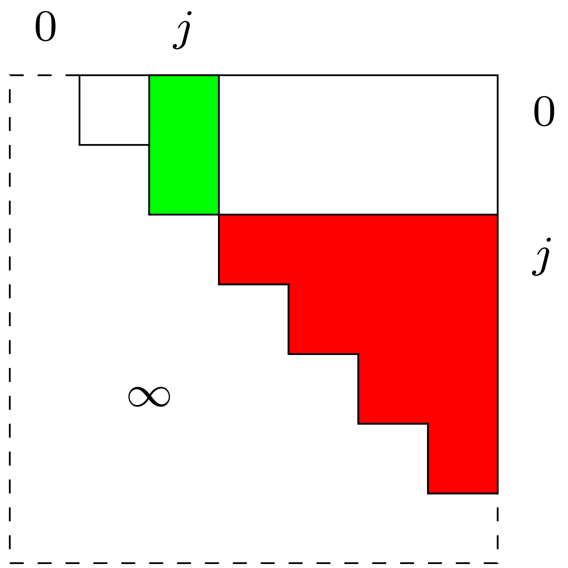
LARSCH Algorithm

Finding row minima in totally monotone matrices with limited dependency. This is also known as online TM problem.

Entries of column j can depend on the row minima of rows i where $M_{i,j} = \infty$.

Green: the column j.

Red: rows that column j can depend on.



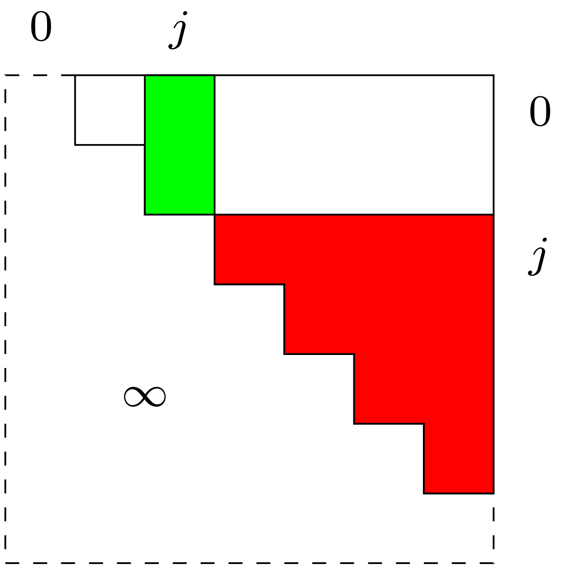
LARSCH Algorithm

Finding row minima in totally monotone matrices with limited dependency. This is also known as online TM problem.

Entries of column j can depend on the row minima of rows i where $M_{i,j} = \infty$.

Green: the column j.

Red: rows that column j can depend on.



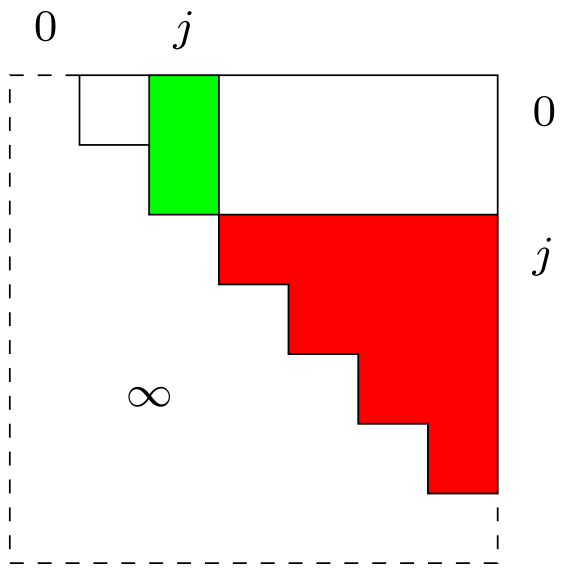
$$R^m_{i,j} = w(i,m) + \{B_{i,j-1} + B_{j,m}\} \qquad (0 \le i < j \le m)$$

LARSCH Algorithm

Finding row minima in totally monotone matrices with limited dependency.
This is also known as online TM problem.

Entries of column j can depend on the row minima of rows i where $M_{i,j} = \infty$.

Green: the column j.
Red: rows that column j can depend on.



$$R^m_{i,j} = w(i,m) + \{B_{i,j-1} + B_{j,m}\} \qquad (0 \le i < j \le m)$$

$\boxed{R^m \text{ satisfies the condition of LARSCH.}}$

Note

- Aggarwal and Park (FOCS '88) developed a 3-D monotone matrix representation of the KY problem and then showed how to use an algorithm due to Wilber (for online computation of maxima of certain concave sequences) to calculate "tube-maxima" of their matrices.
- Careful decomposition of their work yields a decomposition similar to L^m and an $O(n)$ algorithm for calculating its row-minima. This provides an alternative derivation of the previous result (with a symmetry argument extending it to R^m).

Online Algorithm

Online Algorithm

- Recall: Two-sided online

Online Algorithm

- Recall: Two-sided online
  - Current step: Optimal BST for $\text{Key}_{l+1}, \ldots, \text{Key}_r$

Online Algorithm

- Recall: Two-sided online
  - Current step: Optimal BST for $\text{Key}_{l+1}, \ldots, \text{Key}_r$
  - Next step: Add either Key_l or Key_{r+1}.

Online Algorithm

- Recall: Two-sided online
  - Current step: Optimal BST for $\text{Key}_{l+1}, \ldots, \text{Key}_r$
  - Next step: Add either Key_l or Key_{r+1}.

	1	2	3	4	5	6
1	0	146	260	349	491	624
2		0	75	141	250	357
3			0	43	119	204
4				0	44	121
5					0	52
6						0

Online Algorithm

- Recall: Two-sided online
  - Current step: Optimal BST for $\text{Key}_{l+1}, \ldots, \text{Key}_r$
  - Next step: Add either Key_l or Key_{r+1}.
- Online algorithm: using LARSCH

Online Algorithm

- Recall: Two-sided online
  - Current step: Optimal BST for $\text{Key}_{l+1}, \ldots, \text{Key}_r$
  - Next step: Add either Key_l or Key_{r+1}.
- Online algorithm: using LARSCH
  - Add Key_{r+1}

Online Algorithm

- Recall: Two-sided online
  - Current step: Optimal BST for $\text{Key}_{l+1}, \ldots, \text{Key}_r$
  - Next step: Add either Key_l or Key_{r+1}.
- Online algorithm: using LARSCH
  - Add Key_{r+1}
  - Construct R^{r+1}

Online Algorithm

- Recall: Two-sided online
  - Current step: Optimal BST for $\text{Key}_{l+1}, \ldots, \text{Key}_r$
  - Next step: Add either Key_l or Key_{r+1}.
- Online algorithm: using LARSCH
  - Add Key_{r+1}
  - Construct R^{r+1}
  - Solve by LARSCH

Online Algorithm

- Recall: Two-sided online
  - Current step: Optimal BST for $\text{Key}_{l+1}, \ldots, \text{Key}_r$
  - Next step: Add either Key_l or Key_{r+1}.
- Online algorithm: using LARSCH
  - Add Key_{r+1}
  - Construct R^{r+1}
  - Solve by LARSCH
  - $O(n)$ time worst case

Outline

- Background
  - Kunth-Yao (KY) Quadrangle Inequality (QI) Speedup
  - SMAWK Algorithm for finding
    Row Minima of Totally Monotone (TM) Matrices
- The D^d Decomposition
  A transformation from QI to TM such that
  SMAWK solves KY problem as quickly as KY.
- The L^m and R^m Decompositions
  Another transformation from QI to TM that
  (1) implies KY speedup and (2) enables online solution.
- Extensions
  Applying the technique to known generalizations of KY.

Extensions

- Some known extensions

Extensions

- Some known extensions
  - [Michelle L. Wachs (1989)]

Extensions

- Some known extensions
  - [Michelle L. Wachs (1989)]
  - [Al Borchers, Prosenjit Gupta (1994)]

Recurrence

Recurrence

- Original Knuth-Yao

  $B_{i,j} = w(i,j) + \min_{i<t\leq j}\{B_{i,t-1} + B_{t,j}\}$

Recurrence

- Original Knuth-Yao

  $B_{i,j} = w(i,j) + \min_{i<t\leq j}\{B_{i,t-1} + B_{t,j}\}$

- Borchers and Gupta

  $B_{i,j} = \min_{i<t\leq j}\{w(i,t,j) + aB_{i,t-1} + bB_{t,j}\}$

Generalization of QI

Generalization of QI

- Original Knuth-Yao
  - $B_{i,j} = w(i,j) + \min_{i<t\leq j}\{B_{i,t-1} + B_{t,j}\}$

Generalization of QI

- Original Knuth-Yao
  - $B_{i,j} = w(i,j) + \min_{i<t\leq j}\{B_{i,t-1} + B_{t,j}\}$
  - $w(i,j)$ satisfies QI, if $\forall i \leq i' \leq j \leq j'$

$$w(i,j) + w(i',j') \leq w(i',j) + w(i,j')$$

Generalization of QI

- Original Knuth-Yao
  - $B_{i,j} = w(i,j) + \min_{i<t\leq j}\{B_{i,t-1} + B_{t,j}\}$
  - $w(i,j)$ satisfies QI, if $\forall i \leq i' \leq j \leq j'$

$$w(i,j) + w(i',j') \leq w(i',j) + w(i,j')$$

- Borchers and Gupta
  - $B_{i,j} = \min_{i<t\leq j}\{w(i,t,j) + aB_{i,t-1} + bB_{t,j}\}$

Generalization of QI

- Original Knuth-Yao
  - $B_{i,j} = w(i,j) + \min_{i<t\le j}\{B_{i,t-1} + B_{t,j}\}$
  - $w(i,j)$ satisfies QI, if $\forall i \le i' \le j \le j'$

$$w(i,j) + w(i',j') \le w(i',j) + w(i,j')$$

- Borchers and Gupta
  - $B_{i,j} = \min_{i<t\le j}\{w(i,t,j) + aB_{i,t-1} + bB_{t,j}\}$
  - $w(i,t,j)$ satisfies QI, if $\forall i \le i' < t \le t' \le j'$ and $t \le j \le j'$

$$w(i,t,j) + w(i',t',j') \le w(i',t,j) + w(i,t',j')$$

and $\forall i < t \le t' \le j \le j'$ and $i \le i' < t'$

$$w(i',t',j') + w(i,t,j) \le w(i',t',j) + w(i,t,j')$$

Generalization of QI

- Original Knuth-Yao
  - $B_{i,j} = w(i,j) + \min_{i<t\leq j}\{B_{i,t-1} + B_{t,j}\}$
  - $w(i,j)$ satisfies QI, if $\forall i \leq i' \leq j \leq j'$

$$w(i,j) + w(i',j') \leq w(i',j) + w(i,j')$$

- Borchers and Gupta
  - $B_{i,j} = \min_{i<t\leq j}\{w(i,t,j) + aB_{i,t-1} + bB_{t,j}\}$
  - $w(i,t,j)$ satisfies QI, if $\forall i \leq i' < t \leq t' \leq j'$ and $t \leq j \leq j'$

$$w(i,t,j) + w(i',t',j') \leq w(i',t,j) + w(i,t',j')$$

and $\forall i < t \leq t' \leq j \leq j'$ and $i \leq i' < t'$

$$w(i',t',j') + w(i,t,j) \leq w(i',t',j) + w(i,t,j')$$

- If the value of $w(i,t,j)$ is independent of t, the Borchers and Gupta definition becomes the original Knuth-Yao definition.

Generalization of MIL

Generalization of MIL

- Original Knuth-Yao
  - $B_{i,j} = w(i,j) + \min_{i<t\leq j}\{B_{i,t-1} + B_{t,j}\}$

Generalization of MIL

- Original Knuth-Yao
  - $B_{i,j} = w(i,j) + \min_{i<t\leq j}\{B_{i,t-1} + B_{t,j}\}$
  - $w(i,j)$ is Monotone on the integer lattice (MIL), if $\forall [i,j] \subseteq [i',j']$, $w(i,j) \leq w(i',j')$.

Generalization of MIL

- Original Knuth-Yao
  - $B_{i,j} = w(i,j) + \min_{i<t\leq j}\{B_{i,t-1} + B_{t,j}\}$
  - $w(i,j)$ is Monotone on the integer lattice (MIL),
    if $\forall [i,j] \subseteq [i',j']$, $w(i,j) \leq w(i',j')$.
- Borchers and Gupta
  - $B_{i,j} = \min_{i<t\leq j}\{w(i,t,j) + aB_{i,t-1} + bB_{t,j}\}$

Generalization of MIL

- Original Knuth-Yao
  - $B_{i,j} = w(i,j) + \min_{i<t\leq j}\{B_{i,t-1} + B_{t,j}\}$
  - $w(i,j)$ is Monotone on the integer lattice (MIL),
    if $\forall [i,j] \subseteq [i',j']$, $w(i,j) \leq w(i',j')$.
- Borchers and Gupta
  - $B_{i,j} = \min_{i<t\leq j}\{w(i,t,j) + aB_{i,t-1} + bB_{t,j}\}$
  - $w(i,t,j)$ is Monotone on the integer lattice (MIL),
    if $\forall [i,j] \subseteq [i',j']$ and $i < t \leq j$, $w(i,t,j) \leq w(i',t,j')$.

Applications

Applications

- [Borchers, Gupta (1994)]
  Rectilinear Steiner Minimal Arborescence (RSMA) of a slide

Applications

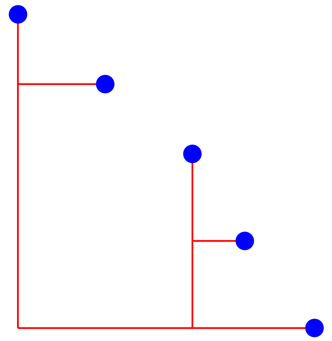
- [Borchers, Gupta (1994)]
  Rectilinear Steiner Minimal Arborescence (RSMA) of a slide
  - Slide: a set of points (x_i, y_i) such that,
    if $i < j$, then $x_i < x_j$ and $y_i > y_j$.

Applications

- [Borchers, Gupta (1994)]
  Rectilinear Steiner Minimal Arborescence (RSMA) of a slide
  - Slide: a set of points (x_i, y_i) such that,
    if $i < j$, then $x_i < x_j$ and $y_i > y_j$.
  - RSMA: a directed tree where each edge
    either goes up or to the right.

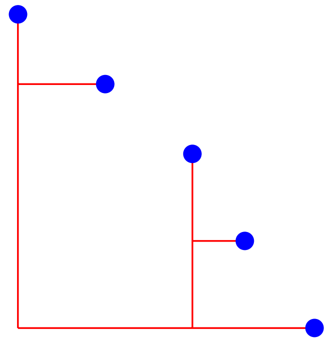
Applications

- [Borchers, Gupta (1994)]
  Rectilinear Steiner Minimal Arborescence (RSMA) of a slide
  - Slide: a set of points (x_i, y_i) such that, if $i < j$, then $x_i < x_j$ and $y_i > y_j$.
  - RSMA: a directed tree where each edge either goes up or to the right.



Applications

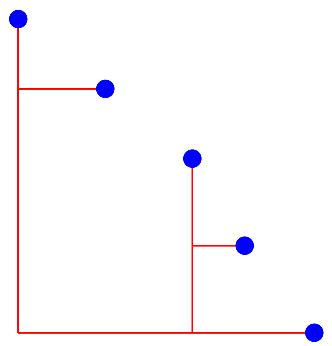
- [Borchers, Gupta (1994)]
  Rectilinear Steiner Minimal Arborescence (RSMA) of a slide
  - Slide: a set of points (x_i, y_i) such that, if $i < j$, then $x_i < x_j$ and $y_i > y_j$.
  - RSMA: a directed tree where each edge either goes up or to the right.



- $B_{i,j} = \min_{i<t\leq j}\{\underbrace{(x_t - x_i + y_{t-1} - y_j)}_{w(i,t,j)} + B_{i,t-1} + B_{t,j}\}$

Applications

- [Borchers, Gupta (1994)]
  Rectilinear Steiner Minimal Arborescence (RSMA) of a slide
  - Slide: a set of points (x_i, y_i) such that, if $i < j$, then $x_i < x_j$ and $y_i > y_j$.
  - RSMA: a directed tree where each edge either goes up or to the right.



  - $B_{i,j} = \min_{i<t\leq j}\{\underbrace{(x_t - x_i + y_{t-1} - y_j)}_{w(i,t,j)} + B_{i,t-1} + B_{t,j}\}$
  - $w(i,t,j)$ satisfies generalized QI and MIL.

Outline

- Background
  - Kunth-Yao (KY) Quadrangle Inequality (QI) Speedup
  - SMAWK Algorithm for finding

    Row Minima of Totally Monotone (TM) Matrices
- The D^d Decomposition

  A transformation from QI to TM such that

  SMAWK solves KY problem as quickly as KY.
- The L^m and R^m Decompositions

  Another transformation from QI to TM that

  (1) implies KY speedup and (2) enables online solution.
- Extensions

  Applying the technique to known generalizations of KY.

Questions?