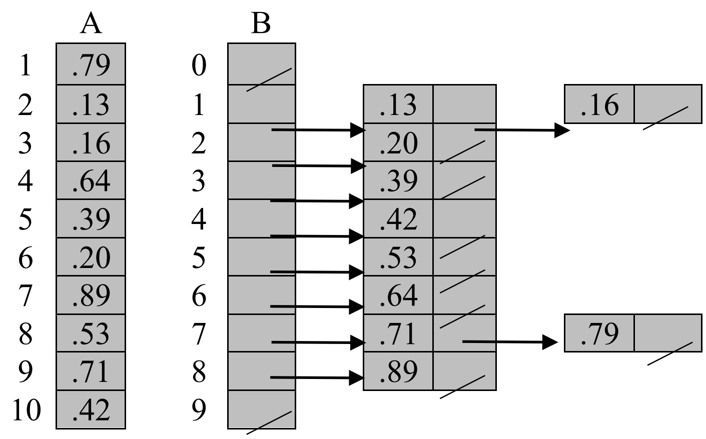
Vinh T. Nguyen

Dr. Rattikorn Hewett

CS 5381 - Analysis of Algorithms
November 22, 2017

1. **QUESTION 13**

Using the Figures below as a model, illustrate the operation of BUCKET-SORT on the array A = ⟨.79, .13, .16, .64, .39, .20, .89, .53, .71, .42⟩.



2. QUESTION 14

Question: *What is the worst-case running time for the bucket-sort algorithm? What simple change to the bucket-sort algorithm preserves its linear expected running time and makes its worst-case running time O(nlgn)?*
Answer:
The worst-case running time for the bucket-sort algorithm when one bucket contains all the inputs. For example, when the inputs contains all .1x (where x is any number from 0-9) as this is not expected. Because the bucket-sort use insert sort to sort the list in the bucket. This insertion sort take the worst-running time $O(n^2)$. So the worst-case running time for the bucket-sort algorithm is: $O(n^2)$.

Because bucket-sort uses insertion sort to sort the list, we just simply change the insertion sort by another algorithms that have the worst-case running time *O(nlogn)*. For example Heapsort or Merge sort.

3. QUESTION 15

Question: *Find an optimal parenthesization of a matrix-chain product whose sequence dimension is <5, 10, 3, 12, 5, 50, 6>*

Answer:

index	0	1	2	3	4	5	6
P	5	10	3	12	5	50	6

N = length (A) – 1 = 7-1=6

Construct a table

	1	0					
	2		0				
	3			0			
j	4				0		
	5					0	
	6						0
		1	2	3	4	5	6
	i						

First iteration.

$m[1,2] = m[1,1] + m[2,2] + P_0P_1P_2 = 0 + 0 + 5x10x3 = 150$

$m[2,3] = m[2,2]+ + m[3,3] + P_1P_2P_3 = 0 + 0 + 10x3x12 = 360$

$m\ [3,4] = m[3,3]+ + m[4,4] + P_2P_3P_4 = 0 + 0 + 3x12x5 = 180$

$m\ [4,5] = m[4,4]+ + m[5,5] + P_3P_4P_5 = 0 + 0 + 12x5x50 = 3000$

$m\ [5,6] = m[5,5]+ + m[6,6] + P_4P_5\ P_6 = 0 + 0 + 5x50x6 = 1500$

	1	0					
	2	150	0				
	3		360	0			
j	4			180	0		
	5				3000	0	
	6					1500	0
		1	2	3	4	5	6
	i						

2nd iteration.

$m[1,3] = m[1,1] + m[2,3] + P_0P_1P_3 = 0 + 360 + 5x10x12 = 960$ k=1

$m[1,3] = m[1,2] + m[3,3] + P_0P_2P_3 = 150 + 0 + 5x3x12 = 330$ k=2

min(m[1,3]) = 330 when k=2

$m[2,4] = m[2,2] + m[3,4] + P_1P_2P_4 = 0 + 180 + 10x3x5 = 330$ k=2

$m[2,4] = m[2,3] + m[4,4] + P_1P_3P_4 = 360 + 0 + 10x12x5 = 960$ k=3

min(m[2,4]) = 330 when k=2

m[3,5] = m[3,3] + m[4,5] + $P_2P_3P_5$ = 0 + 3000 + 3x12x50 = 4,800 k=3
m[3,5] = m[3,4] + m[5,5] + $P_2P_4P_5$ = 180 + 0 + 3x5x50 = 930 k=4
min(m[3,5]) = 930 when k=4

m[4,6] = m[4,4] + m[5,6] + $P_3P_4P_6$ = 0 + 1500 + 12x5x6 = 1,860 k=4
m[4,6] = m[4,5] + m[6,6] + $P_3P_5P_6$ = 3000 + 0 + 12x50x6 = 6,600 k=5
min(m[4,6]) = 1,860 when k=4

	1	0					
	2	150	0				
	3	330	360	0			
j	4		330	180	0		
	5			930	3000	0	
	6				1,860	1500	0
		1	2	3	4	5	6
	i						

3rd iteration.

m[1,4] = m[1,1] + m[2,4] + $P_0P_1P_4$ = 0 + 330 + 5x10x5 = 580 k=1
m[1,4] = m[1,2] + m[3,4] + $P_0P_2P_4$ = 150 + 180 + 5x3x5 = 405 k=2
m[1,4] = m[1,3] + m[4,4] + $P_0P_3P_4$ = 330 + 0 + 5x12x5 = 630 k=3
min(m[1,4]) = 405 when k=2

m[2,5] = m[2,2] + m[3,5] + $P_1P_2P_5$ = 0 + 930 + 10x3x50 = 2,430 k=2
m[2,5] = m[2,3] + m[4,5] + $P_1P_3P_5$ = 360 + 3000 + 10x12x50 = 9,360 k=3
m[2,5] = m[2,4] + m[5,5] + $P_1P_4P_5$ = 330 + 0 + 10x5x50 = 2,830 k=4
min(m[2,5]) = 2,430 when k=2

m[3,6] = m[3,3] + m[4,6] + $P_2P_3P_6$ = 0 + 1,860 + 3x12x6 = 2,076 k=3
m[3,6] = m[3,4] + m[5,6] + $P_2P_4P_6$ = 180 + 1,500 + 3x5x6 = 1,770 k=4
m[3,6] = m[3,5] + m[6,6] + $P_2P_5P_6$ = 930 + 0 + 3x50x6 = 1,830 k=5
min(m[3,6]) = 1,770 when k=4

	1	0					
	2	150	0				
	3	330	360	0			
j	4	405	330	180	0		
	5		2,430	930	3000	0	
	6			1,770	1,860	1500	0
		1	2	3	4	5	6
	i						

4th iteration.

m[1,5] = m[1,1] + m[2,5] + $P_0P_1P_5$ = 0 +2,430 + 5x10x50 = 4,930 k=1

m[1,5] = m[1,2] + m[3,5] + $P_0P_2P_5$ = 150 +930 + 5x3x50 = 1,830 k=2
m[1,5] = m[1,3] + m[4,5] + $P_0P_3P_5$ = 330 +3000 + 5x12x50 = 6,330 k=3
m[1,5] = m[1,4] + m[5,5] + $P_0P_4P_5$ = 405 +0 + 5x5x50 = 1,655 k=4
min(m[1,5]) = 1,655 when k=4

m[2,6] = m[2,2] + m[3,6] + $P_1P_2P_6$ = 0 +1,770+ 10x3x6 = 1,950 k=2
m[2,6] = m[2,3] + m[4,6] + $P_1P_3P_6$ = 360 +1,860+ 10x12x6 = 2,940 k=3
m[2,6] = m[2,4] + m[5,6] + $P_1P_4P_6$ = 330 +1,500+ 10x5x6 = 2,130 k=4
m[2,6] = m[2,5] + m[6,6] + $P_1P_5P_6$ = 2,430+0+ 10x50x6 = 5,430 k=5
min(m[2,6]) = 1,950 when k=2

	1	0					
	2	150	0				
	3	330	360	0			
j	4	405	330	180	0		
	5	1,655	2,430	930	3000	0	
	6		1,950	1,770	1,860	1500	0
		1	2	3	4	5	6
	i						

<u>5th iteration.</u>

m[1,6] = m[1,1] + m[2,6] + $P_0P_1P_6$ = 0+1950+ 5*10*6 = 2,250 k=1
m[1,6] = m[1,2] + m[3,6] + $P_0P_2P_6$ = 150+1770+ 5*3*6 = 2,010 k=2
m[1,6] = m[1,3] + m[4,6] + $P_0P_3P_6$ = 330+1860+ 5*12*6 = 2,550 k=3
m[1,6] = m[1,4] + m[5,6] + $P_0P_4P_6$ = 405+1500+ 5*5*6 = 2,055 k=4
m[1,6] = m[1,5] + m[6,6] + $P_0P_5P_6$ = 1655+0+ 5*50*6 = 3,155 k=5

min(m[2,6]) = 2,010 when k=2

	1	0					
	2	150	0				
	3	330	360	0			
j	4	405	330	180	0		
	5	1,655	2,430	930	3000	0	
	6	2010	1,950	1,770	1,860	1500	0
		1	2	3	4	5	6
	i						

S table

	1	0					
	2	0	0				
	3	2	0	0			
j	4	2	2	0	0		
	5	4	2	4	0	0	
	6	2	2	4	4	0	0
		1	2	3	4	5	6
				i			

A1 = 5x10 A4=12x5
A2 = 10x3 A5 = 5x50
A3 = 3x12 A6 = 50x6

From S table above we have.
m[1,6] = 2 => we have the first cut after A2: (A1xA2)x(A3xA4xA5xA6)
m[3,6] = 4 => we have the second cut after A4: (A1xA2)x((A3xA4)xA5xA6)

So the final result is: **(A1xA2)x((A3xA4)x(A5xA6))**

4. QUESTION 17

Question: *Explain why memorization is ineffective in speeding up a good divide-and-conquer algorithm such as merge-sort?*
Answer:
Divide-and-conquer algorithm breaks a big problem into NON-OVERLAPPING subproblems. Each recursive call is distinctive and can not be reusable. That's why memorizatioin is ineffective on this algorithm.