

2-4 Recurrences

魏恒峰

hfwei@nju.edu.cn

2018 年 04 月 18 日



Maximal Sum Subarray (Problem 4.1 – 5)

- ▶ Array $A[1 \cdots n]$, $a_i \geq 0$
- ▶ To find (the sum of) an MS in A

$$A[-2, 1, -3, \boxed{4, -1, 2, 1}, -5, 4]$$

$MSS[i]$: the sum of the MS ($MS[i]$) in $A[1 \cdots i]$

$MSS[i]$: the sum of the MS ($MS[i]$) in $A[1 \cdots i]$

$$mss = MSS[n]$$

$MSS[i]$: the sum of the MS ($MS[i]$) in $A[1 \cdots i]$

$$mss = MSS[n]$$

Q : Is $a_i \in MS[i]$?

$MSS[i]$: the sum of the MS ($MS[i]$) in $A[1 \cdots i]$

$$mss = MSS[n]$$

Q : Is $a_i \in MS[i]$?

$$MSS[i] = \max\{MSS[i-1], ???\}$$

$MSS[i]$: the sum of the MS *ending with* a_i or 0

$MSS[i]$: the sum of the MS *ending with* a_i or 0

$$mss = \max_{1 \leq i \leq n} MSS[i]$$

Q: where does the $MSS[i]$ start?

$MSS[i]$: the sum of the MS *ending with* a_i or 0

$$mss = \max_{1 \leq i \leq n} MSS[i]$$

Q : where does the $MS[i]$ start?

$$MSS[i] = \max \{MSS[i - 1] + a_i, 0\}$$

$MSS[i]$: the sum of the MS *ending with* a_i or 0

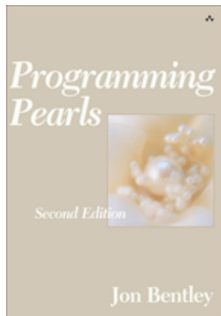
$$mss = \max_{1 \leq i \leq n} MSS[i]$$

Q : where does the MS $[i]$ start?

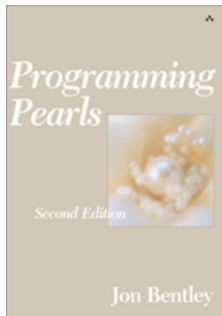
$$MSS[i] = \max \{MSS[i - 1] + a_i, 0\}$$

$$MSS[0] = 0$$

```
1: procedure MSS( $A[1 \cdots n]$ )
2:   MSS[0]  $\leftarrow$  0
3:   for  $i \leftarrow 1$  to  $n$  do
4:     MSS[ $i$ ]  $\leftarrow$   $\max \{ \text{MSS}[i - 1] + A[i], 0 \}$ 
5:   return  $\max_{1 \leq i \leq n} \text{MSS}[i]$ 
```

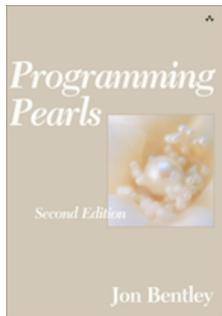


Ulf Grenander $O(n^3) \implies O(n^2)$



Ulf Grenander $O(n^3) \implies O(n^2)$

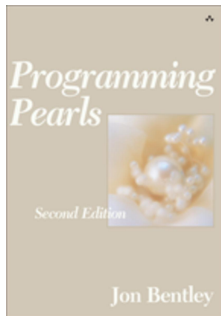
Michael Shamos $O(n \log n)$, onenight



Ulf Grenander $O(n^3) \implies O(n^2)$

Michael Shamos $O(n \log n)$, onenight

Jon Bentley Conjecture: $\Omega(n \log n)$

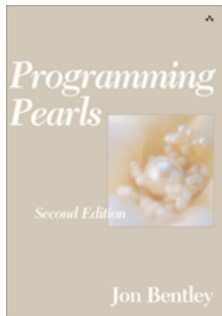


Ulf Grenander $O(n^3) \implies O(n^2)$

Michael Shamos $O(n \log n)$, onenight

Jon Bentley Conjecture: $\Omega(n \log n)$

Michael Shamos Carnegie Mellon seminar



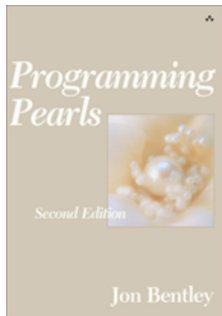
Ulf Grenander $O(n^3) \implies O(n^2)$

Michael Shamos $O(n \log n)$, onenight

Jon Bentley Conjecture: $\Omega(n \log n)$

Michael Shamos Carnegie Mellon seminar

Jay Kadane $O(n)$,



Ulf Grenander $O(n^3) \implies O(n^2)$

Michael Shamos $O(n \log n)$, onenight

Jon Bentley Conjecture: $\Omega(n \log n)$

Michael Shamos Carnegie Mellon seminar

Jay Kadane $O(n)$, ≤ 1 minute

Maximum-product subarray

Maximum-product subarray (Problem 7.4)

- ▶ Array $A[1 \dots n]$
- ▶ Find maximum-product subarray of A

Ending with i

		$\frac{1}{2}$	4	-2	5	$-\frac{1}{5}$	8

Ending with i

		$\frac{1}{2}$	4	-2	5	$-\frac{1}{5}$	8
MaxP[i]	1	$\frac{1}{2}$	4	-2	5	8	64

Ending with i

		$\frac{1}{2}$	4	-2	5	$-\frac{1}{5}$	8
MaxP[i]	1	$\frac{1}{2}$	4	-2	5	8	64
MinP[i]	1	$\frac{1}{2}$	2	-8	-40	-1	-8

Ending with i

		$\frac{1}{2}$	4	-2	5	$-\frac{1}{5}$	8
MaxP[i]	1	$\frac{1}{2}$	4	-2	5	8	64
MinP[i]	1	$\frac{1}{2}$	2	-8	-40	-1	-8

$$\text{MaxP}[i] = \max\{\text{MaxP}[i-1] \cdot a_i, \text{MinP}[i-1] \cdot a_i, a_i\}$$

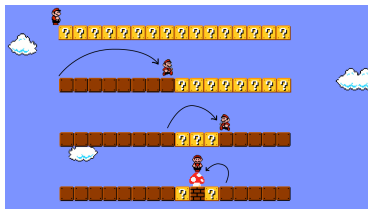
$$\text{MinP}[i] = \min\{\text{MaxP}[i-1] \cdot a_i, \text{MinP}[i-1] \cdot a_i, a_i\}$$

Binary Search (CLRS 4.5 – 3)

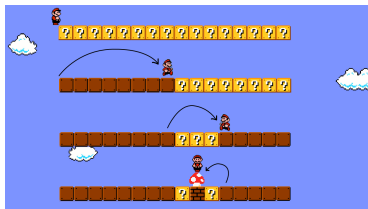
$$T(n) = T(n/2) + \Theta(1)$$

```
1: procedure BINARYSEARCH( $A, L, R, x$ )
2:   if  $R < L$  then
3:     return  $-1$ 
4:    $m \leftarrow L + (R - L)/2$ 
5:   if  $A[m] = x$  then
6:     return  $m$ 
7:   else if  $A[m] > x$  then
8:     return BINARYSEARCH( $A, L, m - 1, x$ )
9:   else
10:    return BINARYSEARCH( $A, m + 1, R, x$ )
```

$$T(n) = \Theta(\log n)$$



$$T(n) = \begin{cases} \max \left\{ T(\lfloor \frac{n-1}{2} \rfloor), T(\lceil \frac{n-1}{2} \rceil) \right\} + 1, & n > 2 \\ 1, & n = 1 \end{cases}$$



$$T(n) = \begin{cases} \max \left\{ T(\lfloor \frac{n-1}{2} \rfloor), T(\lceil \frac{n-1}{2} \rceil) \right\} + 1, & n > 2 \\ 1, & n = 1 \end{cases}$$

$$T(n) = \begin{cases} T(\lfloor \frac{n}{2} \rfloor) + 1, & n > 2 \\ 1, & n = 1 \end{cases}$$

$$T(n) = \begin{cases} T(\lfloor \frac{n}{2} \rfloor) + 1, & n > 2 \\ 1, & n = 1 \end{cases}$$

$$T(n) = \begin{cases} T(\lfloor \frac{n}{2} \rfloor) + 1, & n > 2 \\ 1, & n = 1 \end{cases}$$

$$2^k \leq n < 2^{k+1} \implies T(n) = k + 1$$

$$T(n) = \begin{cases} T(\lfloor \frac{n}{2} \rfloor) + 1, & n > 2 \\ 1, & n = 1 \end{cases}$$

$$2^k \leq n < 2^{k+1} \implies T(n) = k + 1$$

$$T(n) = \lfloor \lg n \rfloor + 1$$

$$T(n) = \begin{cases} T(\lfloor \frac{n}{2} \rfloor) + 1, & n > 2 \\ 1, & n = 1 \end{cases}$$

$$2^k \leq n < 2^{k+1} \implies T(n) = k + 1$$

$$T(n) = \lfloor \lg n \rfloor + 1$$

Theorem

The worst case time complexity of BINARYSEARCH on an input size of n
 =
of bits in the binary representation of n .



Analysis of Mergesort in CLRS (# of Comparisons; $a_i : \infty$ not Counted)

- (a) Analyze the **worst case** $W(n)$ and the **best case** $B(n)$ time complexity of mergesort *as accurately as possible*.

Explore the relation between them and the binary representations of numbers.

Plot $W(n)$ and $B(n)$ and explain what you observe.

- (b) Analyze the **average case** $A(n)$ time complexity of mergesort.

Plot $A(n)$ and explain what you observe.

- (c) **Prove that:** The minimum number of comparisons needed to merge two sorted arrays of equal size m is $2m - 1$.

Analysis of Mergesort in CLRS (# of Comparisons; $a_i : \infty$ not Counted)

- (a) Analyze the **worst case** $W(n)$ and the **best case** $B(n)$ time complexity of mergesort *as accurately as possible*.

Explore the relation between them and the binary representations of numbers.

Plot $W(n)$ and $B(n)$ and explain what you observe.

- (b) Analyze the **average case** $A(n)$ time complexity of mergesort.

Plot $A(n)$ and explain what you observe.

- (c) **Prove that:** The minimum number of comparisons needed to merge two sorted arrays of equal size m is $2m - 1$.



$W(n)$: Consider $W(n + 1)$

$$W(n) = W(\lfloor \frac{n}{2} \rfloor) + W(\lceil \frac{n}{2} \rceil) + (n - 1)$$

$$W(n) = W(\lfloor \frac{n}{2} \rfloor) + W(\lceil \frac{n}{2} \rceil) + (n - 1)$$

Theorem

The worst case time complexity of MERGESORT on an input size of n

=

*The total # of bits in the binary representations of **all the numbers** $< n$.*

$S(n)$: # of bits in the binary representations of all the numbers $< n$.

$S(n)$: # of bits in the binary representations of all the numbers $< n$.

$$S(n) = S(\lfloor \frac{n}{2} \rfloor) + S(\lceil \frac{n}{2} \rceil) + (n - 1)$$

$S(n)$: # of bits in the binary representations of all the numbers $< n$.

$$S(n) = S(\lfloor \frac{n}{2} \rfloor) + S(\lceil \frac{n}{2} \rceil) + (n - 1)$$

$$S(15) = S(7) + S(8) + 14$$

$S(n)$: # of bits in the binary representations of all the numbers $< n$.

$$S(n) = S(\lfloor \frac{n}{2} \rfloor) + S(\lceil \frac{n}{2} \rceil) + (n - 1)$$

$$S(15) = S(7) + S(8) + 14$$

1		1		1		1
10		10		10		10
11		11		11		11
100		100		100		100
101		101		101		101
110		110		110		110
111		111		111		111
1000	=	1000	+	1000	+	1000
1001		1001		1001		1001
1010		1010		1010		1010
1011		1011		1011		1011
1100		1100		1100		1100
1101		1101		1101		1101
1110		1110		1110		1110

Problem (Area-Efficient VLSI Layout)

Embed a **complete binary tree** of n nodes into a grid with minimum **area**.

- ▶ Complete binary tree circuit of

$$\# \text{layer} = 3, 5, 7, \dots$$

- ▶ Vertex on grid; no crossing edges
- ▶ Area:

$$\underbrace{A(n)}_{\text{area}} = \underbrace{H(n)}_{\text{height}} \times \underbrace{W(n)}_{\text{width}}$$

Problem (Area-Efficient VLSI Layout)

Embed a **complete binary tree** of n nodes into a grid with minimum **area**.

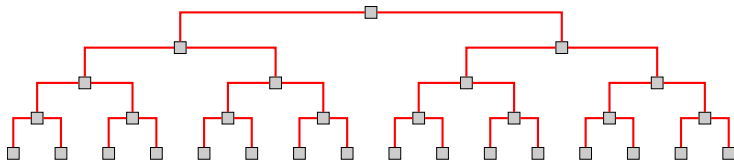
- ▶ Complete binary tree circuit of

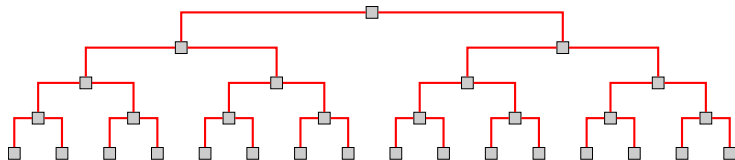
$$\# \text{layer} = 3, 5, 7, \dots$$

- ▶ Vertex on grid; no crossing edges
- ▶ Area:

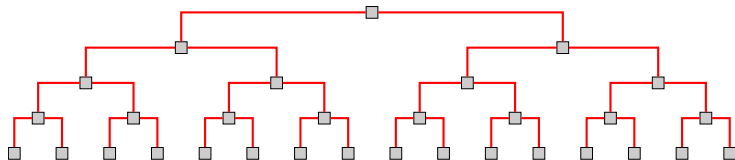
$$\underbrace{A(n)}_{\text{area}} = \underbrace{H(n)}_{\text{height}} \times \underbrace{W(n)}_{\text{width}}$$





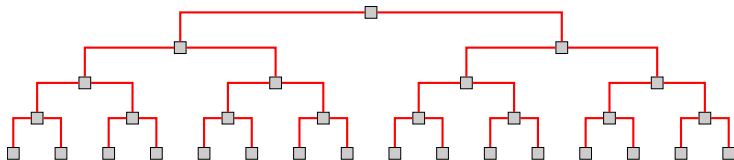


$$H(n) = H\left(\frac{n}{2}\right) + \Theta(1) = \Theta(\log n)$$



$$H(n) = H\left(\frac{n}{2}\right) + \Theta(1) = \Theta(\log n)$$

$$W(n) = 2W\left(\frac{n}{2}\right) + \Theta(1) = \Theta(n)$$



$$H(n) = H\left(\frac{n}{2}\right) + \Theta(1) = \Theta(\log n)$$

$$W(n) = 2W\left(\frac{n}{2}\right) + \Theta(1) = \Theta(n)$$

$$A(n) = \Theta(n \log n)$$

$$Q : \boxed{H(n)} \times \boxed{W(n)} = n$$

$$Q : \boxed{H(n)} \times \boxed{W(n)} = n$$

$$1 \times n$$

$$Q : \boxed{H(n)} \times \boxed{W(n)} = n$$

$$1 \times n$$

$$\frac{n}{\log n} \times \log n$$

$$Q : \boxed{H(n)} \times \boxed{W(n)} = n$$

$$1 \times n$$

$$\frac{n}{\log n} \times \log n$$

$$\boxed{\sqrt{n} \times \sqrt{n}}$$

$$Q : \boxed{H(n)} \times \boxed{W(n)} = n$$

$$1 \times n$$

$$\frac{n}{\log n} \times \log n$$

$$\boxed{\sqrt{n} \times \sqrt{n}}$$

$$H(n) = \Theta(\sqrt{n}), W(n) = \Theta(\sqrt{n}), A(n) = \Theta(n)$$

$$Q : \boxed{H(n)} \times \boxed{W(n)} = n$$

$$1 \times n$$

$$\frac{n}{\log n} \times \log n$$

$$\boxed{\sqrt{n} \times \sqrt{n}}$$

$$H(n) = \Theta(\sqrt{n}), \quad W(n) = \Theta(\sqrt{n}), \quad A(n) = \Theta(n)$$

$$H(n) = \square H\left(\frac{n}{\square}\right) + O(\square)$$

$$Q : \boxed{H(n)} \times \boxed{W(n)} = n$$

$$1 \times n$$

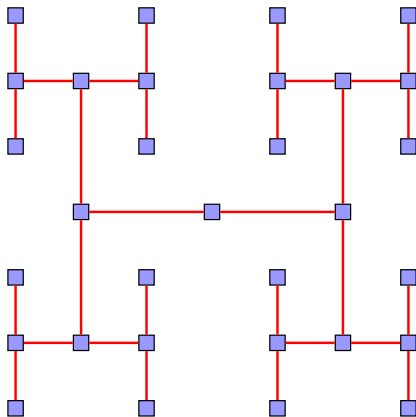
$$\frac{n}{\log n} \times \log n$$

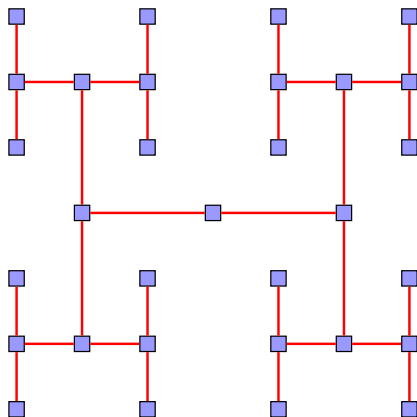
$$\boxed{\sqrt{n} \times \sqrt{n}}$$

$$H(n) = \Theta(\sqrt{n}), \quad W(n) = \Theta(\sqrt{n}), \quad A(n) = \Theta(n)$$

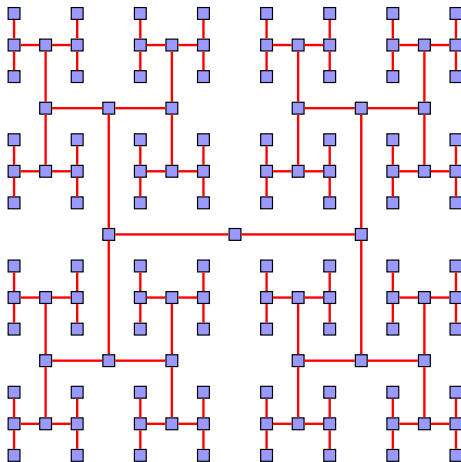
$$H(n) = \square H\left(\frac{n}{\square}\right) + O(\square)$$

$$\boxed{H(n) = 2H\left(\frac{n}{4}\right) + \Theta(1)}$$



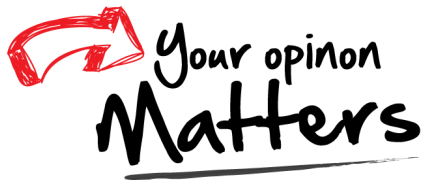


H -layout



“VLSI Theory and Parallel Supercomputing”, Charles E. Leiserson, 1989.

Thank
You!



Office 302

Mailbox: H016

hfwei@nju.edu.cn