2-4 Recurrences

魏恒峰

hfwei@nju.edu.cn

2018 年 04 月 18 日

Maximal Sum Subarray (Problem $4.1 - 5$)

- Array $A[1 \cdots n], a_i >=< 0$
- 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 $A[1 \cdots i]$

$$\text{mss} = \text{MSS}[n]$$

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

$$\mathsf{mss} = \mathsf{MSS}[n]$$

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

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

$$\text{mss} = \text{MSS}[n]$$

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

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

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

$$\boxed{\text{mss} = \max_{1 \le i \le n} \text{MSS}[i]}$$

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

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

$$\boxed{\text{mss} = \max_{1 \le i \le n} \text{MSS}[i]}$$

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

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

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

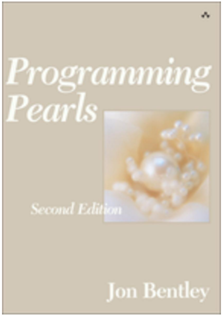
$$\boxed{\mathsf{mss} = \max_{1 \le i \le n} \mathsf{MSS}[i]}$$

Q : where does the $\mathsf{MS}[i]$ start?

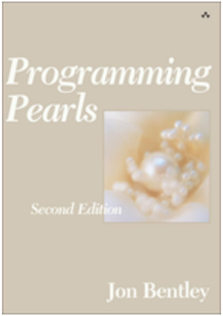
$$\boxed{\mathsf{MSS}[i] = \max \left\{ \mathsf{MSS}[i-1] + a_i, 0 \right\}}$$

$$\mathsf{MSS}[0] = 0$$

```
procedure MSS(A[1 ··· n])
    MSS[0] ← 0
    for i ← 1 to n do
        MSS[i] ← max {MSS[i − 1] + A[i], 0}
    return max_{1≤i≤n} MSS[i]
```

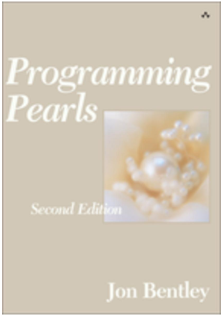


Ulf Grenander $O(n^3) \implies O(n^2)$



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

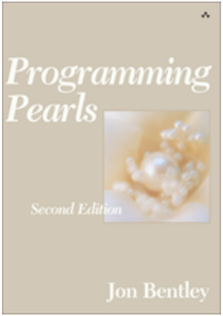
Michael Shamos $O(n \log n)$, onenight



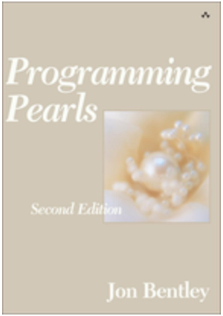
Ulf Grenander $O(n^3) \implies O(n^2)$

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

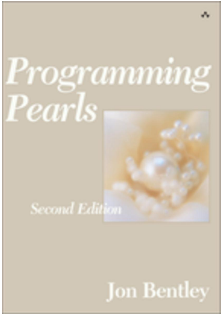
Jon Bentley Conjecture: $\Omega(n \log n)$



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 $A[1 \ldots n]$
- 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

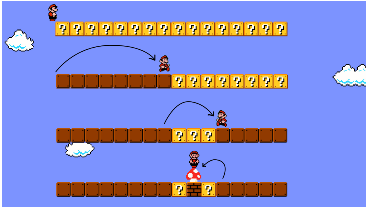
$$\mathsf{MaxP}[i] = \max\{\mathsf{MaxP}[i-1] \cdot a_i, \mathsf{MinP}[i-1] \cdot a_i, a_i\}$$
$$\mathsf{MinP}[i] = \min\{\mathsf{MaxP}[i-1] \cdot a_i, \mathsf{MinP}[i-1] \cdot a_i, a_i\}$$

Binary Search (CLRS $4.5-3$)

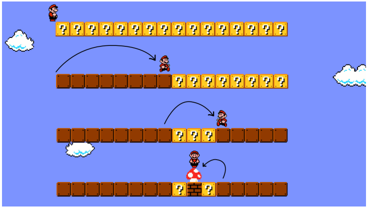
$$T(n) = T(n/2) + \Theta(1)$$

```
procedure BinarySearch(A, L, R, x)
    if R < L then
        return -1

    m ← L + (R - L)/2

    if A[m] = x then
        return m
    else if A[m] > x then
        return BinarySearch(A, L, m - 1, x)
    else
        return BinarySearch(A, m + 1, R, x)
```

$\boxed{T(n) = \Theta(\log n)}$



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



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

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

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

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

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

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

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

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

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

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

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

Theorem

The worst case time complexity of $\textsc{BinarySearch}$ *on an input size of* n

$$=$$

# of bits in the binary representation of n.


TREASURE
HUNT
on foot


Hints
&
Tips


Help
Wanted

Analysis of Mergesort in CLRS (# of Comparisions; $a_i : \infty$ 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 Comparisions; $a_i : \infty$ 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 $W(n+1)$

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

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

Theorem

The worst case time complexity of $\textsc{Mergesort}$ *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 $< n$.

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

1		1		1		**1**
10		10		**1**0		1**0**
11		**1**1		11		1**1**
100		100		**10**0		10**0**
101		**10**1		101		10**1**
110		110		**11**0		11**0**
111		**11**1		111		11**1**
1000	=	1000	+	**100**0	+	100**0**
1001		**100**1		1001		100**1**
1010		1010		**101**0		101**0**
1011		**101**1		1011		101**1**
1100		1100		**110**0		110**0**
1101		**110**1		1101		110**1**
1110		1110		**111**0		111**0**

Problem (Area-Efficient VLSI Layout)

Embedding a complete binary tree into a grid with minimum area.

- Complete binary tree circuit of

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

- Vertex on grid; no crossing edges
- Area:

$$\text{area} = \text{width} \times \text{height}$$

Problem (Area-Efficient VLSI Layout)

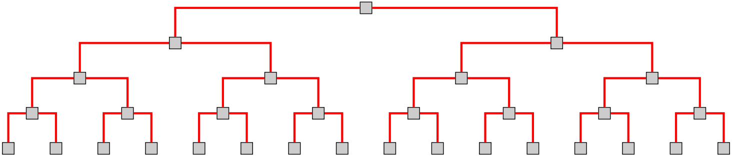
Embedding a complete binary tree into a grid with minimum area.

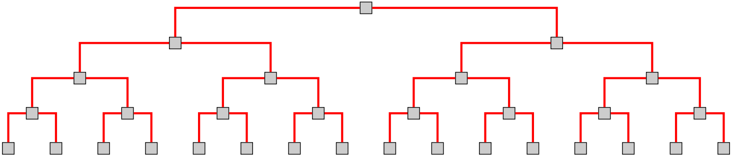
- Complete binary tree circuit of

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

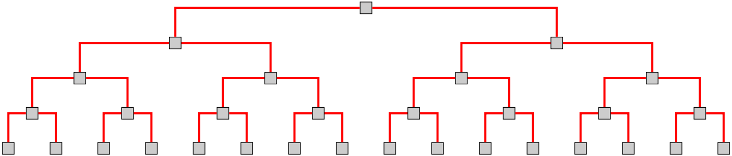
- Vertex on grid; no crossing edges
- Area:

$$\text{area} = \text{width} \times \text{height}$$



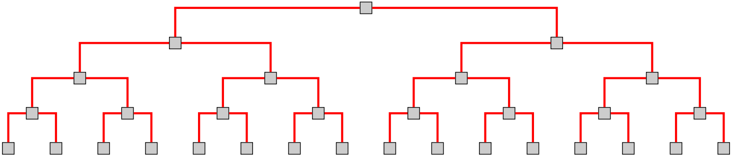


$$H(n) = H(\frac{n}{2}) + \Theta(1) = \Theta(\log n)$$



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

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



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

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

$$\boxed{A(n) = \Theta(n \log n)}$$

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

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

$$1 \times n$$

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

$$1 \times n$$

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

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

$$1 \times n$$

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

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

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

$$1 \times n$$

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

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

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

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

$$1 \times n$$

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

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

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

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

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

$$1 \times n$$

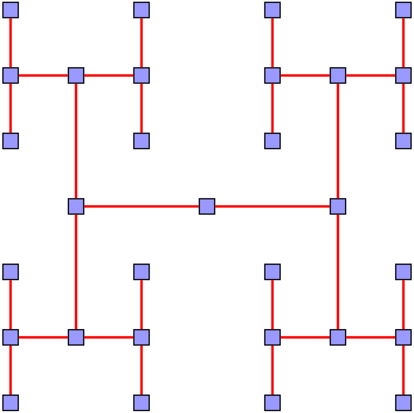
$$\frac{n}{\log n} \times \log n$$

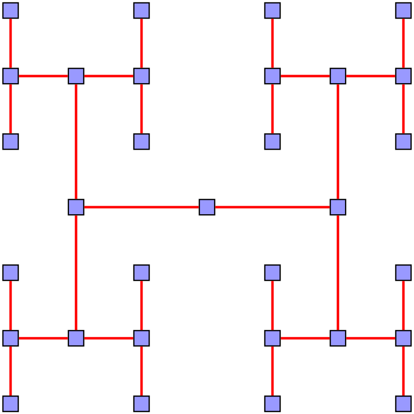
$$\boxed{\sqrt{n} \times \sqrt{n}}$$

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

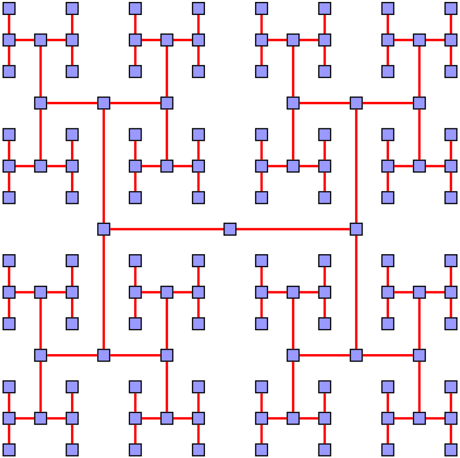
$$H(n) = \square H(\frac{n}{\square}) + O(\square)$$

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



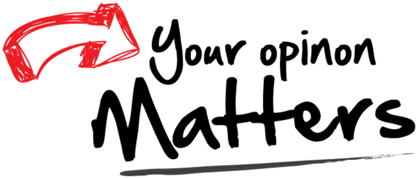


H-layout



"VLSI Theory and Parallel Supercomputing", Charles E. Leiserson, 1989.


Thank
You!



Office 302

Mailbox: H016

hfwei@nju.edu.cn