Tutorial for Mid-term Exam

November 12, 2013

Outline

Outline

Growth of Function

Problem (Growth of Function [**Test 1**])

$S(n) = 1^c + 2^c + 3^c + \ldots + n^c, c \in \mathbb{Z}^+$.

- $S(n) \in O(n^{c+1})$;
- $S(n) \in \Omega(n^{c+1})$.

To prove: $\exists$ constants $a > 0, n_0 > 0$ such that $S(n) \geq an^{c+1}, \forall n \geq n_0$.

$$S(n) \geq \underbrace{(\frac{n}{2})^c + (\frac{n}{2})^c + \ldots + (\frac{n}{2})^c}_{\# = \frac{n}{2}} = (\frac{n}{2})^c \cdot \frac{n}{2} = \underbrace{\frac{1}{2^{c+1}}}_{a} n^{c+1}.$$

Remark:

- inductive on *constant c*

Outline

List of Recurrences

- $T(n) = 2T(n-1) + O(1)$ $T(n) = O(2^n)$ Hanio tower
- $T(n) = 7T(\frac{n}{2}) + O(n^2)$
  $T(n) = O(n^{2.81})$ Strassen matrix multiplication
- $T(n) = 3T(\frac{n}{2}) + O(n)$
  $T(n) = O(n^{1.59})$ Gauss integer multiplication
- $T(n) = 2T(\frac{n}{2}) + O(n)$
  $T(n) = O(n \lg n)$ merge sort, median-quicksort
- $T(k) = 2T(\frac{k}{2}) + O(nk)$, $T(n,k) = 2T(\frac{n}{2}, \frac{k}{2}) + O(n)$ in exam
- $T(n) = T(\frac{n}{5}) + T(\frac{7n}{10}) + O(n)$ linear-time selection
- $T(n) = T(\frac{n}{2}) + O(1)$ $T(n) = O(\lg n)$ binary search
- $T(n) = 2T(\frac{n}{4}) + O(1)$ $T(n) = O(\sqrt{n})$ VLSI layout

Solving Recurrences

Problem (Solving Recurrences [**Test 2**])

- $T(n) = 5T(\frac{n}{2}) + \Theta(n)$ $T(n) = \Theta(n^{\lg 5})$
- $T(n) = 9T(\frac{n}{3}) + \Theta(n^2)$ $T(n) = \Theta(n^2 \lg n)$
- $T(n) = 2T(n-1) + O(1)$ $T(n) = O(2^n)$

Remark:

- $T(n) = 2T(n-1) + O(1)$: unfolding; recursion tree;
  $(1 + 2 + \ldots + 2^{n-1})O(1) = (2^n - 1)O(1)$;
  The Tower of Hanio.
- which to choose?

Merge Sort

Problem (Merge Sort [**Test 3**])

k sorted arrays, each with n elements; merge-sort them.

- one by one: $2n + 3n + \ldots + kn = \frac{k^2+k-2}{2} n$
- divide and conquer: $T(k) = 2T(\frac{k}{2}) + O(nk) = O(n \cdot k \lg k)$;
  unfolding the recursion; dfs (post-order);
  layer by layer; recursion tree.

k-Sort

Problem (k-Sort [Test 6])

$A[1 \ldots n]$, k blocks, each of size $\frac{n}{k}$:

- *sorted within each block*
- *fuzzy sorted among blocks*

E.g., 1 2 4 3; 7 6 8 5; 10 11 9 12; 15 13 16 14

It is a *unfinished* median-quicksort.

- recursion of "median + partition"
- $T(n, k) = 2T(\frac{n}{2}, \frac{k}{2}) + O(n)$
- recursion tree; $\lg k$ layers

VLSI Layout (1)

Problem (VLSI Layout)

Embed a complete binary tree with n leaves into a grid with minimum area.

- *VLSI: Very Large Scale Integration*
- *complete binary tree circuit: #layer $= 3, 5, 7, \ldots$*
- *n leaves (why only leaves?)*
- *grid; vertex + edge (no crossing)*
- *area*
- *take 5-layer complete binary tree as an example*

VLSI Layout (2)

- Naïve embedding

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

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

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

- Smart (H-Layout) embedding

$$\square \times \square = n?\ 1 \times n;\ \frac{n}{\lg n} \times \lg n;\ \sqrt{n} \times \sqrt{n}$$

Goal: $H(n) = \Theta(\sqrt{n}); W(n) = \Theta(\sqrt{n}); A(n) = \Theta(n)$.

$$H(n) = \square H(\frac{n}{\square}) + O(\square); H(n) = 2H(\frac{n}{4}) + O(n^{\frac{1}{2}-\epsilon})$$

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

Here it is: H-Layout

Median of Two Sorted Arrays (1)

Problem (Median of Two Sorted Arrays [$\mathbf{P_{245}}$, 5.18])

$A[1 \ldots n], B[1 \ldots n]$; sorted, ascending order; distinct.
find the median of the combined set of A and B ($O(\lg n)$).

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

$$A = 2, 4, 6, 8, 10 = \boxed{A_1 : (n-1)/2}\ \boxed{M_A : 1}\ \boxed{A_2 : (n-1)/2}$$

$$B = 1, 3, 5, 7, 9 = \boxed{B_1 : (n-1)/2}\ \boxed{M_B : 1}\ \boxed{B_2 : (n-1)/2}$$

$$\boxed{n \text{ is odd}}\, C = \boxed{C_1 : (n-1)}\ \boxed{M_C : 1}\ \boxed{C_2 : (n-1)}$$

$$M_A > M_B \Rightarrow \cancel{M_A + A_2; B_1} \Rightarrow (A_1 + M_1, M_2 + B_2)$$

why not $(A_1, M_2 + B_2)$? and not finished yet ... (why?)

Median of Two Sorted Arrays (2)

To prove: The median M'_C of the subproblem $(A_1 + M_1, M_2 + B_2)$ is also the median M_C of the original problem (A, B).

Proof: why?

$$M_C \in [M_B, M_A)$$

$$M'_C \in [M_B, M_A) \; \boxed{\frac{n+1}{2} - 1} \; \boxed{M'_C : 1} \; \boxed{\frac{n+1}{2} - 1}$$

Extensions:

- $A[1 \dots n], B[1 \dots n]$; not sorted;
- $A[1 \dots n], B[1 \dots n], C[1 \dots n]$; sorted; median
- $A[1 \dots n], B[1 \dots n], \dots$; sorted; median
- $A[1 \dots n], B[1 \dots n], \dots$; sorted; k-th smallest

Two Stacks, One Queue (1)

Problem (Two Stacks, One Queue [**Test 4**])

two stacks $\Rightarrow$ *one queue; correctness proof; amortized analysis*

$\textsc{Enqueue}(x)$: $\textsc{Push}(S_1, x)$;

$\textsc{Dequeue}()$: *while* $S_2 = \emptyset$, $\textsc{Push}(S_2, \textsc{Pop}(S_1))$; $\textsc{Pop}(S_2)$;

Ex: $\textsc{Enqueue}(1, 2, 3)$, $\textsc{Dequeue}()$

Simple observation: $\boxed{S_1 \text{ to push; } S_2 \text{ to pop.}}$

$FIFO \Leftrightarrow$

$$\forall \textsc{De}_1 \prec \textsc{De}_2 : a = \textsc{De}_1, b = \textsc{De}_2 \Rightarrow \exists \textsc{En}_1(a) \prec \textsc{En}_2(b).$$

- Summation method: the sequence is (NOT) known
- Accounting method: $\hat{c}_i = c_i + a_i$; $\sum_i^n \hat{c}_i \geq \sum_i^n c_i$;
  $\boxed{\sum_i^n a_i \geq 0, \forall n}$

Two Stacks, One Queue (2)

item:	PUSH into S_1	POP from S_1	PUSH into S_2	POP from S_2
	1	1	1	1

	$\hat{c}_i$	c_i	a_i
ENQUEUE	3	2	1
DEQUEUE	2	2	0

Ex: ENQUEUE$(1, 2, 3)$, DEQUEUE$()$

$$\sum a_i = \#S_1 \times 2 \geq 0.$$

Two Queues, One Stack (1)

Problem (Two Queues, One Stack)

implement/simulate a stack using two queues

Let's try [left: example; right: code]:

- push (1,2,3,4)
- pop (4) $\Rightarrow Q_1 \xrightarrow{(1,2,3)} Q_2$ [transfer]
- pop (3) $\Rightarrow Q_2 \xrightarrow{(1,2)} Q_1$
- push (5,6,7)
- pop 7 $\Rightarrow Q_1 \xrightarrow{(1,2,5,6)} Q_2$

Q_1 for push; Q_2 for pop:

Push(x): Enqueue(Q_1, x)

Pop(): $Q_1 \xrightarrow{transfer} Q_2$; swap $Q_1 \leftrightarrow Q_2$

Two Queues, One Stack (2)

Analysis:

$$Push^n(Push^1 Pop^1)^{n/2}$$

$$\sum c_i = n + (1 + (n+1)) \times \frac{n}{2} = n + (n+2) \times \frac{n}{2} = (n^2 + 4n)/2$$

$$(\sum c_i)/n = (n+4)/2 = \Theta(n)$$

Remark: Why so bad?

- only use *one* queue: push (1,2,3,4); pop (4); [circulate]
- one queue + circulate

  Push(x): Enqueue(Q_1, x); circulate(Q_1)

  Pop(): Dequeue(Q_1)
- review of "array doubling": expensive, cheap, cheap, ..., expensive
- Push: expensive, cheap, cheap, ..., expensive?

Two Queues, One Stack (3)

Hey, Q_2:

- split the elements into Q_2 (cache vs. memory);
  do not change Pop $\Rightarrow$ "stack order"
- I_1 [stack order]: Q_1 for top of stack; Q_2 for bottom of stack
  Push(x): Enqueue(Q_1, x); circulate(Q_1)
  Pop(): if $Q_1 \neq \emptyset$ Dequeue(Q_1); else Dequeue(Q_2)
- I_2 [$\#Q_1 < \sqrt{\#Q_2}$]: keep $\#Q_1$ small

Ex:

- Push$(1, 2, 3, 4, 5, 6)$:$Q_1 : 6$; $Q_2 : 5, 4, 3, 2, 1$ (why?)
- Push$(7, 8)$ how to [transfer] now?
- $Q_2 \xrightarrow{transfer} Q_1$; swap $Q_1 \leftrightarrow Q_2$
- Push$(9, 10, 11)$

Two Queues, One Stack (4)

- ultimate code:

$$\begin{aligned}
\textsc{Push}(x)\!:\ & \textsc{Enqueue}(Q_1, x);\ \text{circulate}(Q_1);\\
& \text{if } \#Q_1 \geq \sqrt{\#Q_2}\\
& Q_2 \xrightarrow{transfer} Q_1;\ \text{swap } Q_1 \leftrightarrow Q_2;\\
\textsc{Pop}()\!:\ & \text{if } Q_1 \neq \emptyset\ \textsc{Dequeue}(Q_1);\ \text{else } \textsc{Dequeue}(Q_2)
\end{aligned}$$

- analysis: Pop ($O(1)$), Push:

$$\begin{aligned}
\text{cheap:}\ & \#Q_1 < \sqrt{\#Q_2} \Rightarrow O(\sqrt{n})\\
\text{expensive:}\ & \#Q_1 \neq \sqrt{\#Q_2} \Rightarrow O(n)\\
\text{amortized:}\ & E, \underbrace{C, C, \ldots, C}_{\#=O(\sqrt{n})}, E \Rightarrow O(\sqrt{n})
\end{aligned}$$