Searching and Selection

Hengfeng Wei

hfwei@nju.edu.cn

April 8 $\sim$ April 12, 2017

Searching and Selection

The 3rd largest element (Problem 3.1)

$V_k(n)$: min #comparisons to find the k-th largest element of n elements.

$$V_1(n) = n - 1$$
$$V_2(n) = (n - 1) + (\lceil \log n \rceil - 1)$$

$$V_3(n) = ?$$

$$V_3(n) \leq (n - 1) + (\lceil \log n \rceil - 1) + (n - 3)$$

$$V_3(n) \leq (n - 1) + (\lceil \log n \rceil - 1) + (\lceil \log n \rceil - 1)$$

The 3rd largest element (Problem 3.1)

"Q_1: What is the exact value of $V_3(n)$?"

Theorem ($V_3(n)$)

$n \geq 6, n = 2^k + r (0 \leq r < 2^k)$:

$$V_3(n) = \begin{cases} (n-3) + 2k & r = 0, 1 \\ (n-3) + 2k + 1 & 2 \leq r \leq 2^{k-2} + 1 \\ (n-3) + 2k + 2 & \text{o.w.} \end{cases}$$

References

"Selecting the Top Three Elements" by Aigner, 1982.

Reference

"The Art of Computer Programming, Vol 3: Sorting and Searching (Section 5.3.3)" by Donald E. Knuth.

The 3rd largest element (Problem 3.1)

"Q_2: Does your algorithm need to find the 1st and the 2nd elements?"

"YES!"

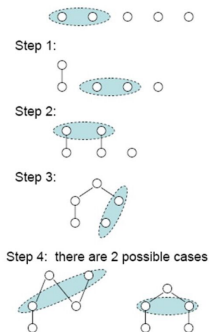"Q_3: Do all algorithms have to find the 1st and the 2nd elements?"
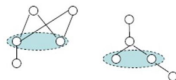
"NO!"

References

"Selecting the Top Three Elements" by Aigner, 1982.

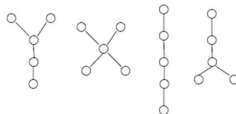# Selection with minimum #comparisons (Problem 3.2)

Selecting the median of 5 elements using 6 comparisons.

Sorting with minimum #comparisons (Problem 2.4)

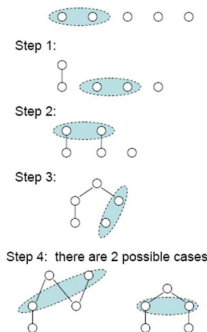Sorting 5 elements using 7 comparisons.

$$S(5) = 7$$

Reference

"The Art of Computer Programming, Vol 3: Sorting and Searching (Section 5.3.1)" by Donald E. Knuth.
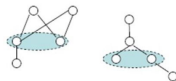
$$S(21) = 66$$

Sorting with minimum #comparisons (Problem 2.4)

Sorting 5 elements using 7 comparisons.
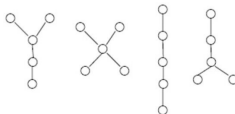
Medians of sorted arrays (Problem 3.7)

http://cs.stackexchange.com/a/33129/4911

Searching and Selection

max / min differences (Problem 4.5)

(a) unsorted; $\max |x - y|$; $O(n)$

(b) sorted; $\max |x - y|$; $O(1)$

(c) unsorted; $\min |x - y|$; $O(n \log n)$

(d) sorted; $\min |x - y|$; $O(n)$

Searching in matrix (Problem 4.6)

- M: matrix $m \times n$
- row: increasing from left to right
- col: increasing from top to down
- Is $x \in M$?

Divide and conquer.

$$T(m, n) = 3T(\frac{m}{2}, \frac{n}{2}) + 1$$

Always checking the lower left corner.

$$T(m, n) = m + n - 1$$

Searching in matrix (Problem 4.6)

Assume $M : n \times n$

$$W(n) \leq 2n - 1$$

$W(n) \geq 2n - 1$ by adversary argument!

$$i + j \leq n - 1 \implies x > M_{ij}$$
$$i + j > n - 1 \implies x < M_{ij}$$

Local minimum (Problem 4.11)

- ▶ Array $A[0 \ldots n-1]$
- ▶ Boundary conditions:

$$A[0] \geq A[1]$$
$$A[n-2] \leq A[n-1]$$

- ▶ Local minimum $A[i]$:

$$A[i-1] \geq A[i] \leq A[i+1]$$

- ▶ Goal: Find *any* local minimum.

Local minimum (Problem 4.11)

1. Checking each element:

$$T(n) = O(n)$$

2. $\min A$:

$$T(n) = O(n)$$

3. Required:

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

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

Local minimum (Problem 4.11)

2D local minimum:

- Matrix $M : n \times n$
- Boundary conditions:

$$\infty$$

- Local minimum $A[i, j]$:

$$A[i, j-1] \geq A[i, j] \leq A[i, j+1]$$
$$A[i-1, j] \geq A[i, j] \leq A[i+1, j]$$

- Goal: Find *any* local minimum.

$$O(n^2) \implies O(n \log n) \implies O(n) \implies O(\log n)$$

$a_i = i$ (Problem 4.2)

▶ Sorted integer sequence $\{a_1, a_2, \ldots, a_n\}$:

$$\forall i \neq j : a_i \neq a_j$$

▶ Goal:

$$\exists ?i : a_i = i$$

$$T(n) = O(n)$$

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

Smallest missing positive integer (Problem 4.3)

- Sorted array $A[1 \ldots n]$:

$$a_i \in \mathbb{Z}^+$$
$$\forall i \neq j : a_i \neq a_j$$

- Goal: Find the smallest missing positive integer.

$$T(n) = O(n)$$

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

Computing $\lceil\sqrt{N}\rceil$ (Problem 4.12)

- Given an n-bit natural number N ($0 \leq N < 2^n - 1$)
- Goal: Compute $\lceil\sqrt{N}\rceil$ using $O(n)$ additions and shifts.

Elementrary operations:

- n-bit + n-bit: $O(1)$
- n-bit shifted by 1-bit: $O(1)$
- $\Rightarrow x^2 : O(n)$

1. Naïve search: $O(2^n \cdot n)$
2. Binary search: $O(n \cdot n)$
3. Binary search in range:

$$2^{\lfloor\frac{n-1}{2}\rfloor} \leq \lceil\sqrt{N}\rceil \leq 2^{\lceil\frac{n}{2}\rceil}$$

$$\lg\left(2^{\lceil\frac{n}{2}\rceil} - 2^{\lfloor\frac{n-1}{2}\rfloor}\right) = O(n)$$

$$O(n \cdot n)$$

Computing $\lceil\sqrt{N}\rceil$ (Problem 4.12)

A Little History:

    2007: Mid-term problem

        $O(n)$ required; NO $O(n)$ solutions, however

$\sim$ 2013: $O(n^2)$

    2014: $O(n)$

Compute square root using (bit) additions and shifts as primitives

    Question: Given an n-bit natural number N, how to compute $\lceil\sqrt{N}\rceil$ using only $O(n)$ (bit) additions and shifts?

    5

    The tip is to use binary search. However, I could not achieve the required complexity (I got $O(n^2)$).

    asked 2 years ago

    viewed 1039 times

    active 2 years ago

$$x^2 : O(n) \to O(1)$$

Computing $\lceil\sqrt{N}\rceil$ (Problem 4.12)

Given

$$M = \lfloor N/4 \rfloor$$

$$x = \lceil\sqrt{M}\rceil \text{ and } (x, x^2),$$

what is

$$y = \lceil\sqrt{N}\rceil \text{ and } (y, y^2)?$$

An Example:

$$
\begin{array}{llll}
N = 280 & y = \lceil\sqrt{280}\rceil = 17 & y^2 = 289 \\
M = \lfloor 280/4 \rfloor = 70 & x = \lceil\sqrt{70}\rceil = 9 & x^2 = 81 \\
M = \lfloor 70/4 \rfloor = 17 & x = \lceil\sqrt{17}\rceil = 5 & x^2 = 25 \\
M = \lfloor 17/4 \rfloor = 4 & x = \lceil\sqrt{4}\rceil = 2 & x^2 = 4 \\
M = \lfloor 4/4 \rfloor = 1 & x = \lceil\sqrt{1}\rceil = 1 & x^2 = 1 \\
\end{array}
$$

Computing $\lceil \sqrt{N} \rceil$ (Problem 4.12)

Algorithm 1 Computing $\lceil \sqrt{N} \rceil$.

    procedure SQRT-ROOT(N)
        if $N < 3$ **then**
            return $1 \Rightarrow (1,1); 2 \Rightarrow (2,4); 3 \Rightarrow (2,4)$
        $M \leftarrow \lfloor N/4 \rfloor$
        $(x, x^2) \leftarrow$ SQRT-ROOT(M)
        return the (y, y^2) with $y^2 \sim N$:

$$(y, y^2) = \begin{cases} y = 2x & y^2 = 4x^2 \\ y = 2x + 1 & y^2 = 4x^2 + 4x + 1 \\ y = 2x - 1 & y^2 = 4x^2 - 4x + 1 \end{cases}$$

Computing $\lceil \sqrt{N} \rceil$

$$T(n) = T(n/4) + O(1) = \Theta(\lg n)$$

$$T(n) = T(n-2) + O(1) = \Theta(n)$$

Space for hashing (Problem 4.4)

$$\text{Key: } x$$
$$\text{Node: } y$$
$$\text{Load factor: } \alpha$$

- ► Closed-address hashing

$$h_c + \alpha y h_c$$

- ► Open-address hashing

$$\frac{\alpha h_c}{\frac{h_c + \alpha y h_c}{x}} = \frac{\alpha x}{1 + \alpha y}$$