2-4 Recurrences

魏恒峰

hfwei@nju.edu.cn

2018 年 04 月 09 日

Binary Search (CLRS $4.5-3$)

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

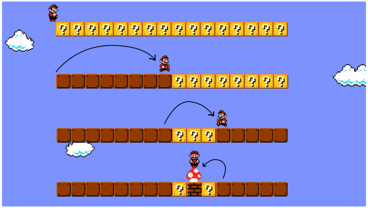
$$T(n) = \Theta(n \lg n)$$

People who analyze algorithms have double happiness.

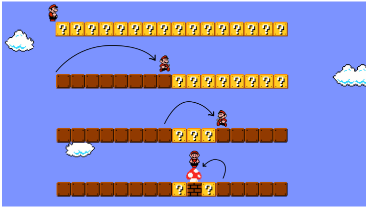
First of all they experience the sheer beauty of elegant mathematical patterns that surround elegant computational procedures.

Then they receive a practical payoff when their theories make it possible to get other jobs done more quickly and more economically.

— Donald E. Knuth (1995)



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



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

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

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

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

$$n = 2^k \implies T(n) = k + 1$$

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

$$n = 2^k \implies T(n) = k + 1$$

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

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

$$n = 2^k \implies T(n) = k + 1$$

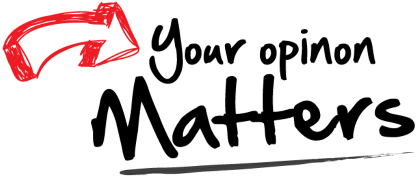
$$2^k \le n < 2^{k+1} \implies T(n) = k + 1$$

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

Theorem

The worst case time complexity (# of comparisons) of BinarySearch *on an input size of* n = *# of bits in the binary representation of* n.


Thank
You!



Office 302

Mailbox: H016

hfwei@nju.edu.cn