1 Recursion invariant: Exponentiator(k) returns 3^k in each recursion

Initialisation: when $k = 0$

$\text{Exponentiator}(k) = 3^0 = 1$

maintenance:

① k is even:

$\text{Exponentiator}(\frac{k}{2})$

recursion invariant is $3^{\frac{k}{2}}$

return $3^{\frac{k}{2}} \cdot 3^{\frac{k}{2}} = 3^k$

② k is odd:

Exponentiator $\left(\frac{k-1}{2}\right)$

recursion Invariant is $3^{\frac{k-1}{2}}$

return $3 * x * x \Rightarrow 3 \cdot 3^{\frac{k-1}{2}} \cdot 3^{\frac{k-1}{2}}$

$\Rightarrow 3^{k}$

Termination

Exponentiator (n) gives 3^{n}

which is at the top of recursion

$$T\left(\frac{n}{2}\right) \leq c \log_2 \frac{n}{2}$$

2 Base case:

when $n=1$ $T(1)=1$

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

$n=3$ $T(2)+1 = 2+1 = 3$

Assume for $n \geq 2$. $T(n) \leq c\log_2 n$ $(c \geq 2)$

step case:

$T(n) = T(\lceil \frac{n}{2} \rceil)+1 \leq c(\log_2(\lceil \frac{n}{2} \rceil))+1$

① Assume n is even:

$T(n) \leq c\log_2(\frac{n}{2})+1$

$= c(\log_2 \frac{n}{2})+1$

$= c(\log_2 \frac{1}{2} \times n)+1$

$= c(\log_2(\frac{1}{2}) + \log_2 n)+1$

$= c(-1+\log_2 n)+1$

$= c\log_2 n - c + 1$

$\leq c\log_2 n$

② Assume n is odd:

$T(n) \leq c(\log_2(\frac{n}{2}+1)+1)$

$= c(\log_2(\frac{n+2}{2})) + 1$

$= c(\log_2(n+2) - 1) + 1$

$= c\log_2(n+2) - c + 1$

$\leq c\log_2(n+2)$

Here we'd better use

situation 1 cés it gives us a more detailed bound.

3

$$T(n) = 2T(n^{0.25})+1$$

$$= 2T(n^{\frac{1}{4}})+1$$

$$m = \log n$$

$$T(2^m) = 2T(2^{\frac{m}{4}})+1$$

$$= S(m) = 2S(\frac{m}{4})+1$$

$a=2 \quad b=4 \quad f(m)=1$

$n^{\log_4 2}$ $\quad f(n) = 1 = n^0$

$\log_4 2 > 0$

so $\varepsilon = \frac{1}{2} - 0 = \frac{1}{2}$

case 1 applies:

Because $f(n) = O(n^{\log_b a - \varepsilon})$

for some constant $\varepsilon > 0$

then $T(n) = \Theta(\log_n^{\log_4 2})$

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

$$S(m) = \Theta(m^{\frac{1}{2}}) = (\log n)^{\frac{1}{2}}$$

$$T(n) = \theta(\log n)^{\frac{1}{2}}$$

$$= \theta(\sqrt{\log n})$$

4

$L := 0$
$R := n$

Function BinarySearch:

(A : Array[n) of Integer
K : Integer
L : Integer
R : Integer)

if R < L then:

Return False

while $L \le R$ do:

$M := L + (R-L)/3$

if A[M] < k, then

Return Binary Search (A, K,
m+1 , R)

else if A[M] > K
Return Binary Search (A, K
L, R-1)

else
Return M

The difference between new-binary search and traditional binary search is

New-binary search is devided by 3

one part is $\frac{n}{3}$ and another part is $\frac{2}{3}n$

Traditional binary-search is devided into 2 parts

New - binary search:

$$T(n) = \begin{cases} 1 & n=1 \\ T(\frac{2}{3}n)+1 & n>1 \end{cases}$$

$\Rightarrow T(n) = \log_2 n$

Traditional:

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

Now compute complexity

we apply substitution method

Base case:

$n=1 \rightarrow T(1) = 1 \quad c\log_{\frac{3}{2}} n = 0$

$n=2 \rightarrow T(1)+1 = 2$

$< \log_{\frac{3}{2}} 2$

$n=3 \rightarrow T(2)+1 = 3$

$< \log_{\frac{3}{2}} 3$

so we assume $T(n) = c\log_{\frac{2}{3}} n$

for $c \geqslant 2$ holds for all positive

Hypo:

$m < n \quad m = \frac{2}{3}n$

$T(\frac{2}{3}n) \leqslant c\log_2(\frac{2}{3}n)$

step case:

$$T(n) = T\left(\tfrac{2}{3}n\right) + 1$$

$$\leq c\left(\log_{\frac{3}{2}}\left(\tfrac{2}{3}n\right)\right) + 1$$

$$= c\left(\log_{\frac{3}{2}} n - \log_{\frac{3}{2}} \tfrac{3}{2}\right) + 1$$

$$= c\left(\log_{\frac{3}{2}} n - 1\right) + 1$$

$$= c \log_{\frac{3}{2}} n - c + 1$$

$$\leq \log_{\frac{3}{2}} n$$

$$T(n) = O\left(\log_{\frac{3}{2}} n\right)$$

5 (A)

$T(n) = 2T(\frac{n}{2}) + n^4$

$a = 2 \quad b = 2 \quad f(n) = n^4$

$n^{\log_2 2} = n$

$n < n^4 = f(n)$

$f(n) = \Omega(n^{\log_2 2+\varepsilon})$

where $\varepsilon = 3$

case 3 applies

if $f(n) = \Omega(n^{\log_b a+\varepsilon})$
$= n^4$

for some constant $\varepsilon > 0$

if $af(\frac{n}{b}) \le cf(n)$ for

some constant $c < 1$ and

all sufficiently large n,

then $T(n) = \Theta(f(n))$

$$af(\tfrac{n}{b}) = 2(\tfrac{n}{2})^4 = 2 \times \frac{n^4}{16} \le cf(n)$$

$$\frac{n^4}{8} \le cn^4$$

$$c \geq \frac{1}{8}$$

$$T(n) = \Theta(n^4)$$

(B)

$$T(n) = T\left(\frac{7}{10}n\right) + n$$

$a = 1 \quad b = \frac{10}{7} \quad f(n) = n$

$$n^{\log_{\frac{10}{7}} 1} = n^0 < n = f(n)$$

$$f(n) = \Omega\left(n^{\log_{\frac{10}{7}} + \varepsilon}\right)$$

where $\varepsilon = 1$

case 3 applies:

for some constant $\varepsilon > 0$

if $af\left(\frac{n}{b}\right) \leq cf(n)$ for .

some constant $c < 1$ and all sufficiently large n,

ten $T(n = \Theta(f(n))$

$$a f\left(\frac{n}{b}\right) = \frac{7n}{10} \le cn$$

$$\frac{7}{10} \le c$$

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

(c) $a=2 \quad b=4 \quad f(n)=\sqrt{n}$

$$n^{\log_4 2} = n^{\frac{1}{2}} = f(n)$$

$$f(n) = \Theta(n^{\log_4 2})$$

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

case 2 applies

if $f(n) = \Theta(n^{\log_b a})$

then $T(n) = \Theta(n^{\log_b a} \log n)$

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

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

(2) $T(n) = n^4$

$T(n) = n$

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

(3) $n^{\frac{1}{2}} \log n < n < n^4$

In general, c is

faster