Assignment 2

06. 09. 24

Problem 1

a) $f(n) = n^2 - 10n + 20$

for $n \to \infty$ $\quad n^2 \gg (-10n + 20)$

$\Rightarrow$ neglect LOT

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

b) $f(n) = 3n + k\log_2(n)$, $k > 0$

for $n \to \infty$ $\quad 3n \gg k\log_2(n)$

$\Rightarrow$ neglect LOT

$\Rightarrow$ $\underline{\underline{f(n) = \Theta(n)}}$

c) $f(n) = (n+k)^2 2^{n+k}$, $k > 0$

$f(n) = 2^{n+k}(n^2 + 2nk + k^2)$

$\Rightarrow (n^2+2nk+k^2) \Rightarrow n^2$ will dominate

$\Rightarrow (n^2+2nk+k^2) \approx n^2$

$\Rightarrow f(n) = 2^{n+k} \cdot n^2 = n^2 \cdot 2^n \cdot 2^k \overset{n \gg 1}{\approx} n^2 \cdot 2^n$

$n \to \infty \Rightarrow f(n) = 2^n$

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

d) $f(n) = n(\log_2(n) + \log_3(n) + \log_4(n))$

as $n \to \infty$ $\quad \log_2(n) \gg \log_3(n) \gg \log_4(n)$

$\Rightarrow$ $\underline{\underline{f(n) = \Theta(n\log_2(n))}}$

e) $f(n) = \sqrt[3]{kn} + 4$, $k > 0$

$\Rightarrow$ $\underline{\underline{f(n) = \Theta(n^{\frac{3}{2}})}}$

f) $f(n) = 6 \cdot 2^n + 2 \cdot 6^n$

for larger n we get

$\Rightarrow$ $6^n \gg 2^n$

$\Rightarrow$ $\underline{\underline{f(n) = \Theta(6^n)}}$

g) $f(n) = n^2 + n^k \log(n)$, $k > 0$

for $n \to$ large values

$n^2 \gg n^k \log(n)$

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

Problem 2

Base case low == high

⇒ function returns array[low]

⇒ $\underline{\Theta(1)}$

practical cases:

⇒ The algorithm does the following

- Divides the array into two halves
- Recursively solves the subproblem for left and right halves
- After the recursive calls, it computes the cross-subarray sum in $O(n)$

⇒ Runtime for this algorithm is

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

⇒ overall time complexity is:

$$\underline{\underline{O(n \log(n))}}$$

Problem 3

stack ← empty stack
function enqueve(element):
    Push element onto stack

function dequeve (element):
    if stack is empty:
        return "Queve is empty"

    if stack is not empty:
        element ← pop from stack

    return element

Time complexity:
enqueve operation is O(1)
dequeve operation is O(1)
if the stack is a Last in first out stack.
if a priority ranking would be added a second stack could be made to go through the whole stack to find the highest priority.

the we would have the following time complexity:
enquene is still O(1)
dequeve becomes O(n) (worst case)