MONASH University

FIT2004 Algorithms and data structures - S1 2023

For constants b and c, consider the recurrence relation given by:

- $T(n) = b$, if n=1
- $T(n) = 25 * T(n/5) + c * n^3$, if n>1

Which of the following statements is true?

- ⦿ a. $T(n) = \Theta(n^3)$
- ○ b. $T(n) = \Theta(n^3 * \log n)$
- ○ c. $T(n) = \Theta(n^2)$
- ○ d. $T(n) = \Theta(n^2 * \log n * \log n)$
- ○ e. $T(n) = \Theta(n^2 * \log n)$

  Clear my choice

Consider the following algorithm, which returns the largest positive integer that divides two positive integers a, b.

```
myfunc(a, b):
    x = a
    y = b
    while (x != 0  &&  y != 0):
        ### Loop Invariant ###
        if (x > y):
            x = x % y
        else:
            y = y % x
    return max(x, y)
```

What is an appropriate loop invariant for this algorithm at the point specified?

- ⦿ a. The largest positive integer that divides both a,b is equal to the largest positive integer that divides both x, y
- ○ b. The largest positive integer that divides both a,b is equal to the largest positive integer that divides both x % y, y
- ○ c. The largest positive integer that divides both a,b is equal to the largest positive integer that divides both x, y % x
- ○ d. The largest positive integer that divides both a,b is equal to the largest positive integer that divides both x - y, y

  Clear my choice

Which of the following sorting algorithms runs in linear time when executed on an already sorted array (assuming that it is an array of integers)?

- ☐ a.   Heapsort

- ☐ b.   Selection Sort

- ☑ c.   Insertion Sort

- ☐ d.   Merge Sort

For a sequence of n integers in range [-2 ... n-2], which sorting algorithm can take a linear time to sort this sequence

- ☑ a.   Counting Sort

- ☐ b.   None of the answers

- ☐ c.   Radix Sort

- ☐ d.   Quick Sort

Which of the following statements are true regarding Karatsuba's multiplication algorithm?

- ☑ a.   Its time complexity is $O(n^{\log_2 3})$.

- ☐ b.   It uses 4 recursions to optimally solve sub-problems.

- ☑ c.   It uses 3 recursions to solve optimally sub-problems.

- ☐ d.   Its time complexity is $O(n^2)$.