School of Computing and Information Systems

COMP20007 Design of Algorithms

Semester 1, 2023

Sample Mid Semester Test

Solutions

Question 1 [2 Marks]

(a)

$$\begin{aligned} f(n) &= (n+1)^3 = n^3 + \text{lower order terms} \\ g(n) &= (2n)^3 = 8n^3 \end{aligned}$$

$$\implies f(n) \in \Theta(g(n))$$

(b)

$$\begin{aligned} f(n) &= 3^{n+1} = 3 \times 3^n = \text{const} \times 3^n \\ g(n) &= (3+1)^n = 4^n \end{aligned}$$

Since $4 > 3$ we have $f(n) \in O(g(n))$.

(c)

$$\begin{aligned} f(n) &= n^3 + 1.1^n \\ g(n) &= (n^3)^2 + 1.1^n = n^6 + 1.1^n \end{aligned}$$

Since $n^c \prec c^n$ we have $n^3 \prec n^6 \prec 1.1^n$, and so $f(n) \in \Theta(g(n))$.

(d)

$$\begin{aligned} f(n) &= \log(n^n) = n\log(n) \\ g(n) &= \sqrt{n} \end{aligned}$$

$\sqrt{n} \prec n$ and so $\sqrt{n} \prec n\log(n)$, therefore $f(n) \in \Omega(g(n))$.

Question 2 [2 Marks]

function IsPalindrome($input$, n)
  $S \leftarrow$ NewStack()
  for $i \leftarrow 0 \dots n/2-1$ inclusive **do**
    Push($S, input[i]$)
  for $i \leftarrow n/2 \dots n-1$ inclusive **do**
    if $input[i] \neq$ Pop(S) **then**
      return False
  return True

We chose a stack since a stack is first in last out (FILO) which corresponds to the order in which we want to check in a palindrome, *e.g.,* the $n/2$th letter must match the most recent letter put on the stack (index $n/2-1$) and the $(n-1)$th letter must match the first element pushed to the stack (the first letter of the string).

Question 3 [3 Marks]

(a) Use Dijkstra's algorithm starting from H since it is an SSSP algorithm and will give distances/paths from H to every other node.

After Dijkstra's, check each of the hospitals (B, D, or E) to see which is closest to H.

(b) Running Dijkstra's from H:

Node								
A	∞	∞	∞	12_B	12_B	11_D	11_D	10_C
B	∞	∞	4_F					
C	∞	∞	∞	∞	∞	8_D	8_D	
D	∞	∞	9_F	6_B	6_B			
E	∞	∞	∞	∞	6_G	6_G		
F	∞	3_H						
G	∞	6_H	5_F	5_F				
H	0							

So the costs are $(B,4)$, $(D,6)$ and $(E,6)$. B is the closest hospital with cost 4 and path HFB.

(c) BFS Order: $HFGBDEAC$

Question 4 [3 marks]

(a) We count the $n == \cdots$ as the basic operations here. So,

$$W(0) = 1, \quad W(1) = 2, \quad W(n) = 3W\left(\frac{n}{3}\right) \text{ for } n > 1$$

Solving this, assuming $n = 3^m$,

$$\begin{aligned}
W(n) &= 3W\left(\frac{n}{3}\right) + 2 \\
&= 3\left(3W\left(\frac{n}{3^2}\right) + 2\right) + 2 \\
&= 3\left(3W\left(3W\left(\frac{n}{3^3}\right) + 2\right) + 2\right) + 2 \\
&= 3^3 W\left(\frac{n}{3^3}\right) + 2 \times 3^2 + 2 \times 3^1 + 2 \times 3^0 \\
&\vdots \\
&= 3^k W\left(\frac{n}{3^k}\right) + 2 \times 3^{(k-1)} + \cdots + 2 \times 3^0 \\
\text{Now let } k &= \log_3(n) \\
&= 3^{\log_3(n)} W\left(\frac{n}{3^{\log_3(n)}}\right) + 2 \times 3^{(\log_3(n)-1)} + \cdots + 2 \times 3^0 \\
&= nW\left(\frac{n}{n}\right) + 2 \sum_{i=0}^{\log_3(n)-1} 3^i \\
&= 2n + 2\frac{3^{\log_3(n)} - 1}{2} \\
&= 2n + n - 1 \\
&= 3n - 1
\end{aligned}$$

(b) This algorithm is not input sensitive, so the expression for runtime applies to all inputs of size n. So the worst case and the best case are the same.

So $W(n) \in \Omega(n)$ and $W(n) \in O(n) \implies W(n) \in \Theta(n)$.

(c) We can check the result of a before evaluating the second recursive call to get b, if a is true then we don't need to evaluate b or c. Similarly with checking b before c.

The worst case is the same, $\Theta(n)$, for example consider k not existing in the array (or existing at index $n-1$).

The best case however is now $\Theta(\log_3(n))$, when we only evaluate the first branch each time.

So this new improved algorithm is $\Omega(\log(n))$ and $O(n)$.