Name _____ Roll number _____ Class /Batch _____ Marks _____

Q4, Given the following algorithms to implement a queue Q using two stacks S1 and S2 (of practically infinite sizes)

```
Enqueue (Q, x)
    1. Push (S1, x)
```

```
Dequeue (Q)
    1. If empty(S2) then
    2.                  while (not (empty(S1))
    3.                      do  x ← pop(S1)
    4.                          Push(S2,x)
    5. Pop(S2)
```

a. Prove that the queue operations have constant amortized time complexity using the accounting method (3)

Charge Enqueue operation as 4. Out of it 1 is used to push into S1, 1 for moving to S2 from S1 and 1 for popping from S2. Hence all operations are paid by the constant cost of 4. Dequeue can be undercharged as 0. Thus the cost of an average operation in large operation sequence is $O(1)$.

b. Prove that the queue operations have constant amortized time complexity using the potential method (3)

$\Phi = 3S_1 + S_2$ where S_1 & S_2 are no: of elements in stacks S_1 & S_2 respectively.

$= 0$ (initially).

Enqueue: $\hat{C} = C + (\Phi_i - \Phi_{i-1})$

$= 1 + (\Phi_{i-1} + 3) - \Phi_{i-1} = 4.$

Dequeue:

- Simple - $\hat{C} = C + (\Phi_i - \Phi_{i-1})$

$= 1 + (\Phi_{i-1} - 1) - \Phi_{i-1} = 0$

- Complex - $\hat{C} = C + (\Phi_i - \Phi_{i-1})$. Let S_1^{i-1} & S_2^{i-1} be the no: of elements originally in S_1 & S_2

$= 2S_1^{i-1} + 1 + (3.0 + (S_1^{i-1} + S_2^{i-1}) - 1 - (3S_1^{i-1} + S_2^{i-1}))$

$= 0$

$\therefore \hat{C} \leq 4$ for all ops

$\Rightarrow O(1)$ is Av cost

Name | Roll number | Class/Batch | Marks

Design and Analysis of Algorithms CS4050

Max Marks: 15 | Midsem Test I | Time: 30 min

Answer all questions in the space provided

Proofs are expected to be mathematically sound and formal

Q1. Prove that $o(g(n)) \subset O(g(n))$ (3)

To prove: (i) $\exists f(n) : f(n) \in O(g(n)) \wedge f(n) \notin \Theta(g(n))$

(ii) $f(n) \in o(g(n)) \Rightarrow f(n) \in O(g(n))$

(i) $g(n) \in O(g(n))$ because $g(n) \leq 1 \cdot g(n)$ for all $n \geq 0$

(ii) $\exists f(n) : f(n) \in \Theta(g(n))$.

$\forall c \; f(n) < c\, g(n)$, for $n \geq n_0$, $c > 0$

$\Rightarrow \exists c : f(n) \leq c g(n) \; \forall n > n_0 \Rightarrow f(n) \in O(g(n))$ (by definition)

Q2. Solve the recurrence $T(n) = T(n/2) + n$, where $T(1) = 1$ (3)

$n^{\log_b a} = n^0$.

$\therefore f(n) = n = \Omega(n^{0+0.5})$ [condition 1 of Master's theorem]

Case - III

$1 \cdot \frac{n}{2} < c \cdot n$. (for $c = 0.75$, for example).

So condition 2 of Master's theorem – Case III holds

Hence $T(n)$ is $\Theta(f(n)) = \Theta(n)$.

Q3. Prove that : If $T(n) = aT(n/b) + f(n)$ where $f(n)$ is a positive function of n and is $\Theta(n^{\log_b a})$, $a \geq 1$ and $b > 1$ then

$$\sum_{i=0}^{\log_2 n - 1} a^i f\left(\frac{n}{b^i}\right) = \Theta(n^{\log_b a} \log_b n) \qquad (3)$$

You can assume that n is a power of b

As $f(n) = \Theta(n^{\log_b a})$, let $f(n) = c \cdot n^{\log_b a}$.

Then $$\sum_{i=0}^{\log_2 n - 1} a^i f\left(\frac{n}{b^i}\right) = \sum_{i=0}^{\log_2 n - 1} a^i c \left(\frac{n}{b^i}\right)^{\log_b a}$$

$$= c\, n^{\log_b a} \sum_{i=0}^{\log_2 n - 1} \frac{a^i}{(b^i)^{\log_b a}} = c\, n^{\log_b a} \sum_{i=0}^{\log_2 n - 1} \frac{a^i}{b^{\log_b a^i}}$$

$$= c \cdot n^{\log_b a} \sum_{i=0}^{\log_2 n - 1} \frac{a^i}{a^i} = c \cdot n^{\log_b a} \sum_{i=0}^{\log_2 n - 1} 1 = c\, n^{\log_b a} \log_2 n$$

$$= c_2\, n^{\log_b a} (\log_b n) = \Theta(n^{\log_b a} \log_b n)$$