CS 201 Data Structures II – Spring 2024

Instructor: Dr. Muhammad Mobeen Movania

Quiz 2 - Solution

Name: ____ **Date:** ______________________

Regn. No. : _____________

There are two questions in this quiz. Each question carries 5 marks.

Q1) Suppose we perform a sequence of n operations on a data structure in which the i^{th} operation costs i if i is an exact power of 2, and 1 otherwise. Determine the amortized cost per operation using the aggregate analysis. (5 marks)

Aggregate:
Let c_i be the cost of i^{th} operation.

$$c_i = \begin{cases} i & if\ i\ is\ an\ exact\ power\ of\ 2 \\ 1 & otherwise \end{cases}$$

n operations will cost: $\sum_{i=1}^{n} c_i \leq n + \sum_{i=0}^{log\ n} 2^j = n + (2n - 1) < 3n$. Thus the average cost of operation T(n) = Total cost < 3n = O(n). And by aggregate analysis, the amortized cost per operation T(n)/n = 3n/n = O(1).

Accounting Method:
Charge each operation \$3 (amortized cost c^i).
If i is not an exact power of 2, pay \$1, and store \$2 as credit.
If i is an exact power of 2, pay \$i, using stored credit.

Operation	Cost	Actual Cost	Credit Remaining
1	3	1	2
2	3	2	3
3	3	1	5
4	3	4	4
5	3	1	6
6	3	1	8
7	3	1	10
8	3	8	5
9	3	1	7
10	3	1	9
…	…	…	…

Since the amortized cost is \$3 per operation, we have $\sum_{i=1}^{n} \hat{c_i}$=3n.
Moreover, from aggregate analysis, we know that the actual cost $\sum_{i=1}^{n} c_i$ < 3n.
So credit never goes -ve.
Since the amortized cost of each operation is O(1), and the amount of credit never goes negative, the total cost of n operations is O(n)

Q2) Suppose we perform a sequence of stack operations on a stack whose size never exceeds k. After every k operations, we make a copy of the entire stack for backup purposes. Using the accounting method, show that the cost of n stack operations, including copying the stack, is O(n) by assigning suitable amortized costs to the various stack operations. (5 marks)

Charge $2 for each PUSH and POP operation

$k for COPY of k items (the amortized cost is 0 since its already covered from the saved credit)

When we call PUSH, we use $1 to pay for the operation, and we store the other $1 on the item pushed. When we call POP, we again use $1 to pay for the operation, and we store the other $1 in the stack itself.

Because the stack size never exceeds k, the actual cost of a COPY operation is at most $k, which is paid by the $k found in the items in the stack and the stack itself.

Since there are k PUSH and POP operations between two consecutive COPY operations, there are $k of credit stored, either on individual items (from PUSH operations) or in the stack itself (from POP operations) by the time a COPY occurs.

Since the amortized cost of each operation is O(1) and the amount of credit never goes negative, the total cost of n operations is O(n).

-----X-----