15-451/651 Algorithm Design & Analysis, Spring 2024

Extra Review Problems

Amortized Analysis

1. **(Short answer / multiple choice)**

   (i) In the game of Fizzbin there are two operations: fizz and bin. Fizz involves putting a card on a stack, and costs 1. Bin involves taking all cards off the stack and has cost equal to the size of the stack plus 1, which is the same as the number of fizz operations since the most recent bin, plus 1.

   For example, the sequence of operations: fizz, bin, fizz, fizz, bin, bin, fizz would cost $1,2,1,1,3,1,1$. Which of the following is the tightest correct upper bound on the amortized cost per operation in a worst-case sequence of n operations?

   (a) 1

   (b) 2

   (c) 3

   (d) 4

   (e) $\log n$

   (f) n

   (ii) Consider a sequence of operations, where the i^{th} operation costs $\lceil \lg i \rceil$. Mark the best upper bound in terms of n on the amortized cost per operation of a sequence of n operations from the choices below:

   (a) $O(1)$

   (b) $O(\lg \lg n)$

   (c) $O(\lg n)$

   (d) $O((\lg n)^2)$

   (e) $O(n)$

   (iii) In lecture we considered the amortized cost of array resizing, where the cost to resize an array from size S to size S' was S. (We assume we start from an empty array.) We considered the strategy where $S' = 2S$ and showed that the amortized cost of a sequence of n appends was at most 3. Now suppose the cost of resize becomes kS for some $k > 0$, all other costs remain the same as in lecture. Choose the best upper bound on the amortized cost per operation for a sequence of n appends from the options below.

(a) 3

(b) $k+2$

(c) $2k+1$

(d) $3k$

(e) $3n$

(iv) Now, consider the strategy where you triple the size of the array once it is full, i.e., $S' = 3S$. Choose the best upper bound on the amortized cost per operation for a sequence of n pushes from the options below.

(a) 2.5

(b) 3

(c) 4.5

(d) 5.5

(e) 8.5

(f) 9

2. **(Ternary Counter)** Suppose that we want to maintain a *ternary counter*, i.e., a number written in ternary / base-3. The only operation we want to support is *increment*, which increases the counter to display the next number. The cost of the increment operation is the number of digits that change compared to its previous value.

   (a) What is the total cost of a sequence of n increments starting from zero?

   (b) Define a suitable potential function for analyzing the cost of each increment

   (c) Use your potential function to determine the amortized cost per increment and show that this matches the cost you got in Part 2a.

3. **(Simulating a queue)** A stack supports "push x" and "pop" and each operation has a cost of 1. Show how to implement a queue that supports enqueue and dequeue using two stacks. Prove using a potential function that the amortized cost of enqueue is 3 and the amortized cost of dequeue is 1.

4. **(Spray paint)** At the FTW Motor Company, there's an infinite line of cars, each of which are initially colored white. Associate the cars with the integers, both positive and negative. Management sends the paint crew a sequence of `Spray` commands: each command is of the form $\texttt{Spray}(x, y, c)$ (where $x \leq y$), which requires spray-painting all the cars/integers in the interval $[x, y]$ with the color c. The cost of each operation $\texttt{Spray}(x, y, c)$ is the number of distinct colors in the range $[x, y]$ before the operation is performed.

   (a) Show using a potential function that the cost of N paint operations is at most $3N$.

   (b) Show that any constant less than 3 would not work.