Lab05-Amortized Analysis

CS214-Algorithm and Complexity, Xiaofeng Gao, Spring 2018.

∗ If there is any problem, please contact TA Xinyu Wu.

∗ Name:Juncheng Wan Student ID:516021910620 Email: 578177149@qq.com

1. A **multistack** consists of an infinite series of stacks $S_0, S_1, S_2, \cdots$, where the i^{th} stack S_i can hold up to 3^i elements. Whenever a user attempts to push an element onto any full stack S_i, we first pop all the elements off S_i and push them onto stack S_{i+1} to make room. (Thus, if S_{i+1} is already full, we first recursively move all its members to S_{i+2} .) An illustrative example is shown in Figure 1. Moving a single element from one stack to the next takes $O(1)$ time. If we push a new element, we always intend to push it in stack S_0.

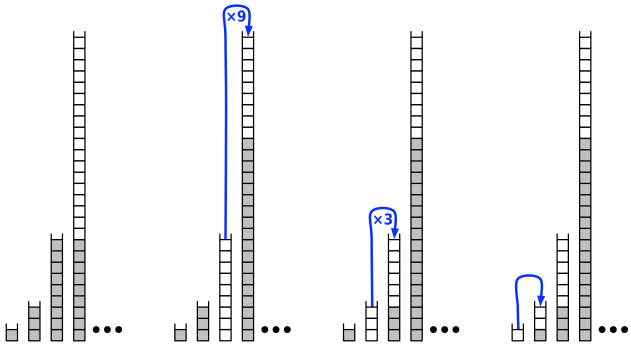

图 1: An example of making room for one new element in a multistack.

(a) In the worst case, how long does it take to push a new element onto a multistack containing n elements?

Solution. First, it is obvious that in the worst case the time complexity is less or equal to n, because no element will be moved twice. That is to say, n is the upper bound of the worst case.
Second, I prove that a case can reach it. Suppose that $3^0, 3^1, \cdots, 3^k, n - \frac{3^{k+1}}{2}$ are the numbers of elements in $S_0, S_1, \cdots, S_k, S_{k+1}$. It is obvious that $3^0+3^1+\cdots+3^k+n-\frac{3^{k+1}}{2} = n$. If $n - \frac{3^{k+1}}{2} > 2 \times 3^k$, we get $3^k + n - \frac{3^{k+1}}{2} > 3^{k+1}$. Therefore, when a new element is pushed in, all the elements in multistack have to move. The time complexity of the worst case is $O(n)$. □

(b) Prove that the amortized cost of a push operation is $O(\log n)$ by *Aggregation Analysis*.

Proof. Assume that $a_0, a_1, \cdots, a_k, \cdots$ are the numbers of elements in $S_0, S_1, \cdots, S_k, \cdots$ and $a_{k+1}, ak+2, \cdots$ are all equal to 0. It is obvious that if an element is in S_i, this element must have been moved $i+1$ times. According to amortized cost, we want to calculate $T(n) = \frac{1}{n}(1 \times a_0 + 2 \times a_1 + \cdots + (k+1) \times a_k)$.
First, it is obvious that $a_0 \leq 3^0, a_1 \leq 3^1, \cdots, a_k \leq 3^k$.

$$\begin{aligned} T(n) &= \frac{1}{n}[1 \times a_0 + 2 \times a_1 + \cdots + (k+1) \times a_k] \\ &\leq \frac{1}{n}[1 \times 3^0 + 2 \times 3^1 + \cdots + (k+1) \times 3^k] \\ &= \frac{1}{4n}[(2k+1)3^{k+1} + 1] \end{aligned}$$

Second, it is obvious that $a_0 \geq 1, a_1 \geq 3^0, \cdots, a_k \geq 3^{k-1}$. Because elements are puched when they are full in stack and they are puched together.

$$\begin{aligned} n &= a_0 + a_1 + \cdots + a_k \\ &\geq 1 + 3^0 + 3^1 + \cdots + 3^{k-1} \\ &= \frac{3^k + 1}{2} \end{aligned}$$

We get $k \leq log_3(2n-1)$.

$$\begin{aligned} T(n) &\leq \frac{1}{4n}[(2k+1)3^{k+1} + 1] \\ &\leq \frac{1}{4n}[(2log_3(2n-1)+1) \times 3 \times (2n-1) + 1] \\ &\leq \frac{1}{4n}[(2log_3(2n-1)+1) \times 3 \times (2n-1) + 1] \\ &= 3log_3(2n-1) - \frac{3log_3(2n-1)}{2n} + \frac{1}{4n} \\ &= O(logn) \end{aligned}$$

□

(c) (Optional Subquestion with Bonus) Prove that the amortized cost of a push operation is $O(\log n)$ by *Potential Method.*

Proof. In the ith operation, set $k(i)$ the maximal number of continuous stacks that is full and these $k(i)$ stacks are $S_0, S_1, \cdots, S_{k(i)}$.
Set the potential function as follows:

$$\Phi(i) = 3^0 + 3^1 + 3^2 + \cdots + 3^{k(i)} - k(i)$$

Originally, we have:

$$C(i) = 3^0 + 3^1 + 3^2 + \cdots + 3^{k(i)}$$

However, this potential function can only solove the preblem when the number of used stacks increaseds in the ith operation.
Therefore, I try to change it:

$$\Phi(i) = \max\{f(i), g(i)\} - k(i)$$

where $k(i)$ is the number of all the used stacks, $f(i)$ is the number of used slots in the $S_0, S_1, \cdots, S_{k(i)}$, $g(i)$ is the number of all the empty slots in the $S_0, S_1, \cdots, S_{k(i)}$.
It does not work.
Then I set:

$$\Phi(i) = i - 0.5g(i)$$

where $g(i)$ is the number of all the empty slots in the $S_0, S_1, \cdots, S_{k(i)}$.
However, it does not work. The above is my train of thought.

□

2. A factory needs to deliver a kind of product in 2 months. Suppose that for month i: the contract requires the factory to deliver d_i products; the selling price for a product is s_i; the capitalized cost for a product is c_i; the working time needed for a product is t_i. In month i, the normal working time is no more than T_i, and it is allowed to do extra work, but the extra working time is no more than T'_i, and each product produced in extra working time has an extra c'_i in its capitalized cost. If the products are stored (not delivered) in month i, the storage cost p_i is required to pay for each stored product.

Please design a production plan in the form of linear programming, which maximizes the overall profit under all possible constraints mentioned above.

(a) Please add some necessary explanations on your objective function and constraints, and finally write your LP in *standard* form.

Solution. Set x_1 the number of products produced in the normal working time in the 1st month, x_2 the number of products produced in the extra working time in the 1st month, x_3 the number of products produced in the normal working time in the 2nd month, x_4 the number of products produced in the extra working time in the 2nd month, z the overall profit of this factory.

Considering all possible constraints mentioned above, we want to maximize the function z.

$$
\begin{aligned}
\textbf{max} \quad z =& (s_1 - c_1)x_1 + (s_1 - c_1 - c'_1)x_2 + (s_2 - c_2)x_3 + (s_2 - c_2 - c'_2)x_4 \\
& - (x_1 + x_2 - d_1)p_1 - (x_3 + x_4 - d_2)p_2 \\
=& (s_1 - c_1 - p_1)x_1 + (s_1 - c_1 - c'_1 - p_1)x_2 + (s_2 - c_2 - p_2)x_3 \\
& + (s_2 - c_2 - c'_2 - p_2)x_4 + d_1p_1 + d_2p_2 \\
\textbf{s.t} \quad & (-1)x_1 + (-1)x_2 \le -d_1 \\
& (-1)x_3 + (-1)x_4 \le -d_2 \\
& t_1x_1 \le T_1 \\
& t_1x_2 \le T'_1 \\
& t_2x_3 \le T_2 \\
& t_2x_4 \le T'_2 \\
& x_1, x_2, x_3, x_4 \ge 0
\end{aligned}
$$

However, it is not standard form of LP. Set $x_5 = d_1p_1 + d_2p_2$. We get the standard form of LP as follows:

$$
\begin{aligned}
\textbf{max} \quad z =& (s_1 - c_1 - p_1)x_1 + (s_1 - c_1 - c'_1 - p_1)x_2 + \\
& + (s_2 - c_2 - c'_2 - p_2)x_4 + 1 \cdot x_5 \\
\textbf{s.t} \quad & (-1)x_1 + (-1)x_2 \le -d_1 \\
& (-1)x_3 + (-1)x_4 \le -d_2 \\
& t_1x_1 \le T_1 \\
& t_1x_2 \le T'_1 \\
& t_2x_3 \le T_2 \\
& t_2x_4 \le T'_2 \\
& x_5 \le d_1p_1 + d_2p_2 \\
& (-1)x_5 \le -d_1p_1 - d_2p_2 \\
& x_1, x_2, x_3, x_4, x_5 \ge 0
\end{aligned}
$$

□

(b) Transform your LP into its dual form.

Solution. The dual form of LP above is as follows:

$$\begin{aligned}
\textbf{min} \quad z = &(-d_1)y_1 + (-d_2)y_2 + T_1y_3 + T_1'y_4 + T_2y_5 + T_2'y_6 \\
&+ (d_1p_1 + d_2p_2)y_7 + (-d_1p_1 - d_2p_2)y_8 \\
\textbf{s.t} \quad &(-1)y_1 + t_1y_3 \geq s_1 - c_1 - p_1 \\
&(-1)y_1 + t_1y_4 \geq s_1 - c_1 - c_1' - p_1 \\
&(-1)y_2 + t_2y_5 \geq s_2 - c_2 - p_2 \\
&(-1)y_2 + t_2y_6 \geq s_2 - c_2 - c_2' - p_2 \\
&y_7 + (-1)y_8 \geq 1 \\
&y_1, y_2, y_3, y_4, y_5, y_6, y_7, y_8 \geq 0
\end{aligned}$$

□

Remark: Please include .pdf and .tex files in your .rar or .zip file with standard file names.