Lab07-Amortized Analysis

CS214-Algorithm and Complexity, Xiaofeng Gao, Spring 2020.

∗ If there is any problem, please contact TA Shuodian Yu.

∗ Name:Yijia Diao   Student ID:518030910146   Email: diao_yijia@sjtu.edu.cn

1. For the TABLE-DELETE Operation in Dynamic Tables, suppose we construct a table by multiplying its size by $\frac{2}{3}$ when the load factor drops below $\frac{1}{3}$. Using *Potential Method* to prove that the amortized cost of a TABLE-DELETE that uses this strategy is bounded above by a constant.

   Proof. Assume the potential function:

$$\Phi(T) = \begin{cases} 2 \times num[T] - size[T], & \alpha(T) \geq \frac{1}{2} \\ \frac{1}{2} size[T] - num[T], & \alpha(T) < \frac{1}{2} \end{cases}$$

   (1) $\frac{1}{3} \leq \alpha_i \leq \frac{1}{2}$, which indicates that the contraction is not triggered. We have

$$\begin{aligned} \widehat{C_i} &= C_i + \Phi_i - \Phi_{i-1} \\ &= 1 + (\frac{1}{2} size_i - num_i) - (\frac{1}{2} size_{i-1} - num_{i-1}) \\ &= 1 + (\frac{1}{2} size_{i-1} - (num_{i-1} - 1)) - (\frac{1}{2} size_{i-1} - num_{i-1}) \\ &= 2 \end{aligned}$$

   (2) A contraction is triggered, which means $\alpha_i < \frac{1}{2}$. We have

$$\begin{aligned} \widehat{C_i} &= C_i + \Phi_i - \Phi_{i-1} \\ &= num_i + 1 + (\frac{1}{2} size_i - num_i) - (\frac{1}{2} size_{i-1} - num_{i-1}) \\ &= num_i + 1 + (\frac{1}{2} size_i - num_i) - (\frac{3}{2} size_i - num_i - 1) \\ &= 2 + num_i - \frac{1}{2} size_i \\ &= 2 \end{aligned}$$

   (3) $\alpha_i > \frac{1}{2}$, which indicates that the contraction is not triggered. We have

$$\begin{aligned} \widehat{C_i} &= C_i + \Phi_i - \Phi_{i-1} \\ &= 1 + (2 \times num_i - size_i) - (2 \times num_{i-1} - size_{i-1}) \\ &= 1 + (2 \times num_{i-1} - 1 - size_{i-1}) - (2 \times num_{i-1} - size_{i-1}) \\ &= 0 \end{aligned}$$

   Therefore, we can conclude that the amortized cost of a TABLE-DELETE that uses this strategy is bounded above by a constant. □

2. 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.

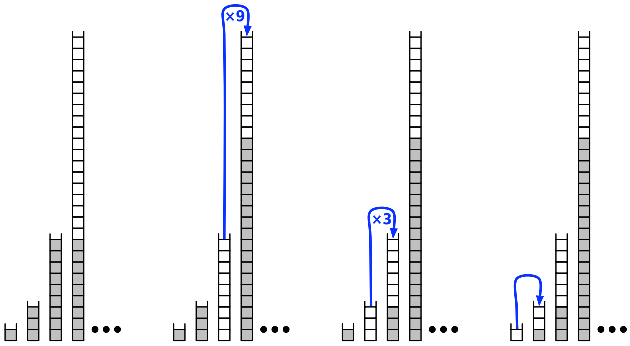

图 1: An example of making room for one new element in a multistack.

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

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

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

Solution. (a) The worst case is, when $n = 1+\sum_{i=0}^{k} 3^i$, all non-empty stacks are already full. Each element needs to be popped and pushed into its next stack, and it takes $T(n) = n$.

(b) Assume the total time of inserting n elements by S_n. This is no worse than the case: n^{th} insert is the worst case, which equals to $n = 1 + \sum_{i=0}^{k} 3^i$. So we have

$$S_n \leq 1 + 1 \times 3^0 + 2 \times 3^1 + \cdots + (k+1) \times 3^k \overset{def}{=} T_k \tag{1}$$

Since

$$3 \times T_k = 3 + 1 \times 3^1 + 2 \times 3^2 + \cdots + (k+1) \times 3^{k+1} \tag{2}$$

(Equ.2$-$Equ.1)/2, we have

$$T_k = \frac{(2k+1) * 3^{k+1} + 5}{4} \tag{3}$$

Since

$$n \leq 1 + (1 + 3 + 3^2 + \cdots + 3^k) = \frac{3^{k+1}+1}{2} \tag{4}$$

From inEqu.3 and inEqu.4, we have $k < \log_3(2n-1) \leq k+1$, which means $O(k) = O(\log n)$.
Thus, the amortized cost of a push operation is

$$\begin{aligned}\frac{S_n}{n} \leq \frac{T_k}{n} &= \frac{6k+3}{4} \times \frac{2n-1}{n} + \frac{5}{4n} \\ &= O(k) \\ &= O(\log n)\end{aligned}$$

(c) First assume **weight function**: $weight_i = \sum_{j=1}^{k_i} j \times |S_j|$, in which k_i is the number of non-empty stack after the ith insertion. Since $|S_j| < \log n$, we have $weight_i \leq k_i \log n$. Also, we have $c_i = weight_i - weight_{i-1}$.
Then assume the potential function:

$$\Phi(i) = \begin{cases} k_i \log n - weight_i, & i > 0 \\ 0, & i = 0 \end{cases}$$

Since $weight_i \leq k_i \log n$, we have $\forall i > 0 : \Phi(i) \geq 0 \Rightarrow \Phi(n) \geq \Phi(0)$. Thus,

$$\begin{aligned} \hat{C}_i &= C_i + \Phi_i - \Phi_{i-1} \\ &= C_i + (k_i \log n - weight_i) - (k_{i-1} \log n - weight_{i-1}) \\ &= C_i - (weight_i - weight_{i-1}) + (k_i - k_{i-1}) \log n \end{aligned}$$

Since the times of push and pops are $weight_i - weight_{i-1}$, we have

$$\hat{C}_i = (k_i - k_{i-1}) \log n = O(\log n)$$

Therefore, we can conclude that the amortized cost of a push operation is $O(\log n)$.

□

3. Given a graph $G = (V, E)$, and let V' be a strict subset of V. Prove the following propositions.

(a) Let T be a minimum spanning tree of a G. Let T' be the subgraph of T induced by V', and let G' be the subgraph of G induced by V'. Then T' is a minimum spanning tree of G' if T' is connected.

(b) Let e be a minimum weight edge which connects V' and $V \setminus V'$. There exists a minimum weight spanning tree which contains e.

Solution. (a) (Proof by Contradiction) Suppose that T' is not the minimum spanning tree of G'.
Assume the minimum spanning tree of G' is R. Since T' is a connected graph, and is the induced subgraph of T by V', we have $R \backslash T' \neq \emptyset$. Then we construct a subgraph of G: $S = R \cup (T \backslash T')$. Since R is the minimum spanning tree of G', and T is the minimum spanning tree of G, we have S is the minimum spanning tree of G. But $R \backslash T' \neq \emptyset \Rightarrow R \nsubseteq T$, which means $S \neq T$, contradiction.
Therefore we can conclude that T' is a minimum spanning tree.

(b) (Proof by Contradiction) Suppose that any minimum weight spanning tree does not contain e.

i. $|V'| = 1$ or $|V \backslash V'| = 1$: the minimum weight spanning tree must contain e', which makes connection with the single vertex and the other part. Suppose the weight of $\forall e \in G : w(e)$, then we have $w(e) < w(e')$. So if we change e' to e, we will get another spanning tree that has a smaller weight. Contradiction.

ii. $|V'| > 1$ and $|V \backslash V'| > 1$: According to this hypothesis, we can find $V_1 \subsetneq V'$, and the minimum spanning tree of V' does not contain the minimum weight edge between V_1 and $V' \backslash V_1$. We can do this division to case 3(b)i, and it's the same for $V \backslash V'$. So we still get the contradiction.

Therefore, we can conclude that there exists a minimum weight spanning tree which contains e.

□

Remark: Please include your .pdf, .tex files for uploading with standard file names.