Lab07-Amortized Analysis

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

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

$*$ Name:Zehao Wang   Student ID:518021910976   Email: davidwang200099@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.

   Solution. Define

   $$\Phi_i = |2num_i - size_i|$$

   as the potential function.

   If the i-th step is TABLE-DELETE, then

   - When $\alpha_{i-1} > \frac{1}{2}$, then $num_i = num_{i-1} - 1$ and $size_i = size_{i-1}$.

   $$\begin{aligned}\hat{C}_i &= 1 + |2num_i - size_i| - |2num_{i-1} - size_{i-1}| \\ &= 1 + |2num_i - size_i| - |2num_{i-1} - size_{i-1}| \\ &= 1 + 2num_i - size_i - 2num_{i-1} + size_{i-1} = -1\end{aligned}$$

   - When $\alpha_{i-1} = \frac{1}{2}$, namely $num_{i-1} = \frac{size_{i-1}}{2}$,
     then $num_i = num_{i-1} - 1$ and $size_i = size_{i-1}$.

   $$\begin{aligned}\hat{C}_i &= 1 + |2num_i - size_i| - |2num_{i-1} - size_{i-1}| \\ &= 1 + 0 + 2(num_i + 1) - size_i = 3\end{aligned}$$

   - When $\frac{1}{3} < \alpha_{i-1} < \frac{1}{2}$, then $num_i = num_{i-1} - 1$, $size_i = size_{i-1}$.

   $$\begin{aligned}\hat{C}_i &= 1 + |2num_i - size_i| - |2num_{i-1} - size_{i-1}| \\ &= 1 + size_i - 2num_i + 2num_{i-1} - size_{i-1} \\ &= 1 + 2(num_{i-1} - num_i) \\ &= 1 + 2 = 3\end{aligned} \tag{1}$$

   - When $\alpha_{i-1} = \frac{1}{3}$, then $num_i = num_{i-1} - 1$, $size_i = \frac{2size_{i-1}}{3}$, $num_i + 1 = \frac{size_i}{2}$.

   $$\begin{aligned}\hat{C}_i &= num_{i-1} + |2num_i - size_i| - |2num_{i-1} - size_{i-1}| \\ &= num_{i-1} + 1 - |2num_{i-1} - 3n_{i-1}| \\ &= num_{i-1} + 1 - num_{i-1} = 1\end{aligned}$$

   Therefore the amortized cost of TABLE-DELETE 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.

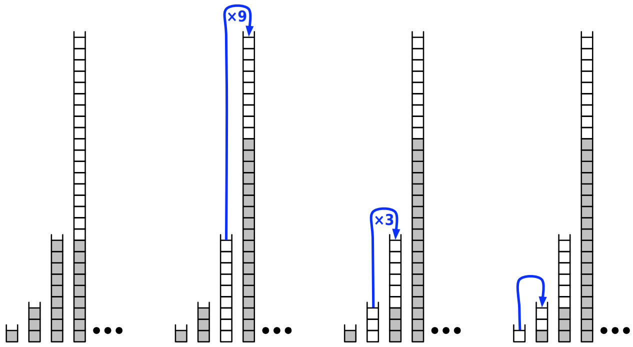

图 1: An example of making room for one new element in a multistack.

(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 happens when $n = \sum_{i=0}^{k} 3^i$. Every element in the multi-stack should be moved once.

Then the time complexity of the push operation is $O(n)$.

(b) Assume that we try to push $\sum_{i=0}^{k+1} 3^i + 1$ elements into the multi-stack.

Consider the process of pushing elements to make the size of stack change from $\sum_{i=0}^{k} 3^i + 1$ to $\sum_{i=0}^{k+1} 3^i + 1$. In this process, $n = 3^{k+1}$, namely $k = \log_3 n - 1$.

Assume when pushing the m-th element into the multi-stack, we need to move T_m elements from one stack to another.

By observation, we can see when m is a multiple of 3^i, $T_m = \sum_{j=0}^{i} 3^j$. Otherwise,$T_m = 1$.

Therefore the total number of movement is:

$$
\begin{aligned}
&1 + 1 + (3 + 1) + 1 + 1 + (3 + 1) + 1 + 1 + (9 + 3 + 1) + \cdots \\
&= 3^0 \times 3^{k+1} + 3^1 \times 3^k + 3^2 \times 3^{k-1} + \cdots + 3^{k+1} \times 1 \\
&= (k + 2)3^{k+1}
\end{aligned}
$$

3^{k+1} elements are pushed into the multi-stack while changing the size of multi-stack change from $\sum_{i=0}^{k} +1 = n - 3^{k+1}$ to $\sum_{i=0}^{k+1} 3^i + 1$.

We have

$$
\frac{(k + 2)3^{k+1}}{3^{k+1}} = k + 2 = \log_3 n + 1
$$

Taking pushing new elements into consideration, the average number of operations is $\log_3 n + 1 + 1$.

Therefore the time complexity is $O(\log n)$.

(c) Assume that the number of elements in the multi-stack is n.

By observation, we can know that there exists an $k(n)$ such that $\sum_{i=0}^{k(n)} 3^i < n \leq \sum_{i=0}^{k(n)+1} 3^i$.

Then we can get $k(n) = \lceil \log_3(2n-1) \rceil - 2$.

And we define $|S_j(i)|$ to be the number of elements in the j-th stack when we have pushed i elements into the multi-stack.$(i \geq 0,\ j \geq 0)$

Define

$$\Phi(i) = \begin{cases} (k(i) \log_3 n - \sum_{j=0}^{k(i)} j|S_j(i)| & i \neq \sum_{j=0}^{m} 3^j + 1, integer\ m \\ 0 & i = 0 \end{cases}$$

as potential function.

Define

$$C_i = \sum_{j=0}^{k(i)} j|S_j(i)| - \sum_{j=0}^{k(i-1)} j|S_j(i-1)|$$

Then

$$\hat{C}_i = C_i + \Phi(i) - \Phi(i-1)$$

$$= C_i + k(i)\log_3 i - \sum_{j=0}^{k(i)} j|S_j(i)| - k(i-1)\log_3(i-1) + \sum_{j=0}^{k(i-1)} j|S_j(i-1)|$$

$$\approx (k(i) - k(i-1)) \log_3 i$$

$$\leq log_3 i$$

Therefore we can know the time complexity of push is $O(\log n)$.

$\square$

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)

- **Statement: If T' is connected, it must be a tree.**
  Proof:
  Assume T' is connected but T' is not a tree. According to the definition of a tree, if T' is connected but not a tree, it must have a cycle in it.
  However, $T = T' \cup (T \setminus T')$. If T' has a cycle, T must have a cycle, which is contradictory to the condition that T is a tree. Therefore T' must be a tree if it is connected.

- Assume T is a minimum spanning tree but T' is not a minimum spanning tree, then there is a certain edge e_0 which connect two vertexes in V' and form a cycle C_0 with certain edges in E. What's more, $w(e_0) < w(a)$ for an edge $a_0 \in C_0$.
  In this case, the total weight of $T' \oplus (a_0, e_0)$ is smaller than T'. However, because $T' \subseteq T$, therefore $(T' \oplus (a_0, e_0)) \cup (T \setminus T')$ is a spanning tree with a smaller weight than T, which is contradictory to the condition that T is a minimal spanning tree. Therefore the assumption does not make sense. Then T' is a minimum spanning tree of G' if T' is connected.

(b) Assume that the minimum spanning tree T' does not contain e.

Edges in the minimum spanning tree T' can be divided into three parts:

- Edges connecting vertexes in V'
- Edges connecting vertexes in $V \setminus V'$
- A single edge e' which connects one vertex in V and another in $V \setminus V'$

Because the two vertexes connected by e' and e are in V and $V \setminus V'$ separately, neither e' nor e can form any circle with edges in E' or edges connecting vertexes in $V \setminus V'$.

According to the definition of e, its weight is smaller than e'.

Then replace e' with e and we can still get a spanning tree T. This new tree has a smaller weight than T', which is contradictory to the assumption that T' is minimum spanning tree.

Therefore the assumption does not make sense. There must exist a minimum weight spanning tree which contains e.

$\square$

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