3-2 Amortized Analysis

Hengfeng Wei

hfwei@nju.edu.cn

October 08, 2018

Robert Tarjan

John Hopcroft

For fundamental achievements
in the design and analysis of algorithms and data structures.

— *Turing Award*, 1986

SIAM J. ALG. DISC. METH.
Vol. 6, No. 2, April 1985

016

AMORTIZED COMPUTATIONAL COMPLEXITY*

ROBERT ENDRE TARJAN†

Abstract. A powerful technique in the complexity analysis of data structures is *amortization*, or averaging over time. Amortized running time is a realistic but robust complexity measure for which we can obtain surprisingly tight upper and lower bounds on a variety of algorithms. By following the principle of designing algorithms whose amortized complexity is low, we obtain "self-adjusting" data structures that are simple, flexible and efficient. This paper surveys recent work by several researchers on amortized complexity.

"Amortized Computational Complexity", 1985

Amortized analysis is

an algorithm analysis technique for

analyzing a sequence of operations

irrespective of the input to show that

the average cost per operation is small, even though

a single operation within the sequence might be expensive.

By averaging the cost per operation over a worst-case sequence,

amortized analysis can yield a time complexity that is

more robust than average-case analysis, since

its probabilistic assumptions on inputs may be false,

and more realistic than worst-case analysis, since it may be

impossible for every operation to take the worst-case time,

as occurs often in manipulation of data structures.

Accounting Method

Summation Method

Potential Method

Amortized Analysis

The Summation Method

$$\sum$$

$$o_1, o_2, \ldots, o_n$$

$$c_1, c_2, \ldots, c_n$$

$$o_1, o_2, \ldots, o_n$$

$$c_1, c_2, \ldots, c_n$$

$$\boxed{\forall i,\ \hat{c}_i = \frac{\left(\sum_{i=1}^{n} c_i\right)}{n}}$$

The Summation Method for Dynamic Tables

The Summation Method for Dynamic Tables

On any sequence of n TABLE-INSERT on an *initially empty* array.

The Summation Method for Dynamic Tables

On any sequence of n TABLE-INSERT on an *initially empty* array.

$$\begin{array}{ccccccccccc} o_i: & o_1 & o_2 & o_3 & o_4 & o_5 & o_6 & o_7 & o_8 & o_9 & o_{10} \\ c_i: & 1 & 2 & 3 & 1 & 5 & 1 & 1 & 1 & 9 & 1 \end{array}$$

The Summation Method for Dynamic Tables

On any sequence of n TABLE-INSERT on an *initially empty* array.

$$\begin{array}{lcccccccccc} o_i: & o_1 & o_2 & o_3 & o_4 & o_5 & o_6 & o_7 & o_8 & o_9 & o_{10} \\ c_i: & 1 & 2 & 3 & 1 & 5 & 1 & 1 & 1 & 9 & 1 \end{array}$$

$$c_i = \begin{cases} (i-1)+1 = i & \text{if } i-1 \text{ is an exact power of } 2 \\ 1 & \text{o.w.} \end{cases}$$

The Summation Method for Dynamic Tables

On any sequence of n TABLE-INSERT on an *initially empty* array.

$$\begin{array}{ccccccccccc} o_i: & o_1 & o_2 & o_3 & o_4 & o_5 & o_6 & o_7 & o_8 & o_9 & o_{10} \\ c_i: & 1 & 2 & 3 & 1 & 5 & 1 & 1 & 1 & 9 & 1 \end{array}$$

$$c_i = \begin{cases} (i-1)+1 = i & \text{if } i-1 \text{ is an exact power of 2} \\ 1 & \text{o.w.} \end{cases}$$

$$\sum_{i=1}^{n} c_i = n + \sum_{j=0}^{\lceil \log n \rceil - 1} 2^j = n + (2^{\lceil \log n \rceil} - 1) < n + 2n = 3n$$

The Summation Method for Dynamic Tables

On any sequence of n TABLE-INSERT on an *initially empty* array.

$$\begin{array}{lcccccccccc} o_i: & o_1 & o_2 & o_3 & o_4 & o_5 & o_6 & o_7 & o_8 & o_9 & o_{10} \\ c_i: & 1 & 2 & 3 & 1 & 5 & 1 & 1 & 1 & 9 & 1 \end{array}$$

$$c_i = \begin{cases} (i-1)+1 = i & \text{if } i-1 \text{ is an exact power of 2} \\ 1 & \text{o.w.} \end{cases}$$

$$\sum_{i=1}^{n} c_i = n + \sum_{j=0}^{\lceil \log n \rceil - 1} 2^j = n + (2^{\lceil \log n \rceil} - 1) < n + 2n = 3n$$

$$\boxed{\forall i,\ \hat{c}_i = 3}$$

The Accounting Method

$$o_1, o_2, \ldots, o_n$$

$$c_1, c_2, \ldots, c_n$$

$$a_1, a_2, \ldots, a_n$$

$$o_1, o_2, \ldots, o_n$$

$$c_1, c_2, \ldots, c_n$$

$$a_1, a_2, \ldots, a_n$$

$$\boxed{\hat{c}_i = c_i + a_i \quad (a_i >=< 0)}$$

Amortized Cost $=$ Actual Cost $+$ Accounting Cost

$$o_1, o_2, \ldots, o_n$$

$$c_1, c_2, \ldots, c_n$$

$$a_1, a_2, \ldots, a_n$$

$$\boxed{\hat{c}_i = c_i + a_i \quad (a_i >=< 0)}$$

Amortized Cost $=$ Actual Cost $+$ Accounting Cost

$$\forall n, \ \sum_{i=1}^{n} c_i \leq \sum_{i=1}^{n} \hat{c}_i$$

$$o_1, o_2, \ldots, o_n$$

$$c_1, c_2, \ldots, c_n$$

$$a_1, a_2, \ldots, a_n$$

$$\boxed{\hat{c}_i = c_i + a_i \ \ (a_i >=< 0)}$$

Amortized Cost $=$ Actual Cost $+$ Accounting Cost

$$\forall n, \ \sum_{i=1}^{n} c_i \le \sum_{i=1}^{n} \hat{c}_i \Longleftarrow \boxed{\forall n, \ \sum_{i=1}^{n} a_i \ge 0}$$

$$o_1, o_2, \ldots, o_n$$

$$c_1, c_2, \ldots, c_n$$

$$a_1, a_2, \ldots, a_n$$

$$\boxed{\hat{c}_i = c_i + a_i \quad (a_i >=< 0)}$$

Amortized Cost $=$ Actual Cost $+$ Accounting Cost

$$\forall n, \ \sum_{i=1}^{n} c_i \leq \sum_{i=1}^{n} \hat{c}_i \Longleftarrow \boxed{\forall n, \ \sum_{i=1}^{n} a_i \geq 0}$$

Key Point: Put the accounting cost on specific objects.

The Accounting Method for Dynamic Tables

$$Q : \hat{c}_i = 3 \textit{ vs. } \hat{c}_i = 2$$

The Accounting Method for Dynamic Tables

$$Q : \hat{c}_i = 3 \textit{ vs. } \hat{c}_i = 2$$

$$\hat{c}_i = 3 =$$

The Accounting Method for Dynamic Tables

$$Q: \hat{c}_i = 3 \text{ vs. } \hat{c}_i = 2$$

$$\hat{c}_i = 3 = \underbrace{1}_{\text{insert}} + \underbrace{1}_{\text{move itself}} + \underbrace{1}_{\text{help move another}}$$

The Accounting Method for Dynamic Tables

$$Q : \hat{c}_i = 3 \textit{ vs. } \hat{c}_i = 2$$

$$\hat{c}_i = 3 = \underbrace{1}_{\text{insert}} + \underbrace{1}_{\text{move itself}} + \underbrace{1}_{\text{help move another}}$$

	$\hat{c}_i$	c_i	a_i
TABLE-INSERT *(normal)*	3	1	2
TABLE-INSERT *(expansion)*	3	$1+t$	$-t+2$

The Potential Method

$$D_0,\ o_1,\ D_1,\ o_2,\ \cdots,\ \underbrace{D_{i-1},\ o_i,\ D_i}_{\text{the } i\text{-th operation}},\ \cdots,\ D_{n-1},\ o_n,\ D_n$$

$$D_0,\ o_1,\ D_1,\ o_2,\ \cdots,\ \underbrace{D_{i-1},\ o_i,\ D_i}_{\text{the } i\text{-th operation}},\ \cdots,\ D_{n-1},\ o_n,\ D_n$$

$$\Phi : \left\{ D_i \mid 0 \le i \le n \right\} \to \mathcal{R}$$

$$D_0,\ o_1,\ D_1,\ o_2,\ \cdots,\ \underbrace{D_{i-1},\ o_i,\ D_i}_{\text{the } i\text{-th operation}},\ \cdots,\ D_{n-1},\ o_n,\ D_n$$

$$\Phi : \Big\{ D_i \mid 0 \le i \le n \Big\} \to \mathcal{R}$$

$$\boxed{\hat{c}_i = c_i + \Big(\Phi(D_i) - \Phi(D_{i-1}) \Big)}$$

$$D_0,\ o_1,\ D_1,\ o_2,\ \cdots,\ \underbrace{D_{i-1},\ o_i,\ D_i}_{\text{the } i\text{-th operation}},\ \cdots,\ D_{n-1},\ o_n,\ D_n$$

$$\Phi : \Big\{ D_i \mid 0 \le i \le n \Big\} \to \mathcal{R}$$

$$\boxed{\hat{c}_i = c_i + \Big(\Phi(D_i) - \Phi(D_{i-1}) \Big)}$$

$$\sum_{1 \le i \le n} c_i = \left(\sum_{1 \le i \le n} \hat{c}_i \right) + \Big(\underbrace{\Phi(D_0) - \Phi(D_n)}_{\text{net decrease in potential}} \Big)$$

$$\sum_{1 \le i \le n} c_i = \left(\sum_{1 \le i \le n} \hat{c}_i \right) + \Big(\underbrace{\Phi(D_0) - \Phi(D_n)}_{\text{net decrease in potential}} \Big)$$

$$\sum_{1 \le i \le n} c_i = \left(\sum_{1 \le i \le n} \hat{c}_i \right) + \Big(\underbrace{\Phi(D_0) - \Phi(D_n)}_{\text{net decrease in potential}} \Big)$$

$$\underbrace{\Phi(D_0) - \Phi(D_n)}_{\text{net decrease in potential}} \le \square \implies \boxed{\sum_{1 \le i \le n} c_i \le \left(\sum_{1 \le i \le n} \hat{c}_i \right) + \square}$$

$$\sum_{1 \le i \le n} c_i = \left(\sum_{1 \le i \le n} \hat{c}_i \right) + \Big(\underbrace{\Phi(D_0) - \Phi(D_n)}_{\text{net decrease in potential}} \Big)$$

$$\underbrace{\Phi(D_0) - \Phi(D_n)}_{\text{net decrease in potential}} \le \square \implies \boxed{\sum_{1 \le i \le n} c_i \le \left(\sum_{1 \le i \le n} \hat{c}_i \right) + \square}$$

$$\square = 0 \; \big(\forall i,\; \Phi(D_i) \ge \Phi(D_0)\big) \implies \forall n, \; \sum_{1 \le i \le n} c_i \le \sum_{1 \le i \le n} \hat{c}_i$$

$$\sum_{1 \le i \le n} c_i = \left(\sum_{1 \le i \le n} \hat{c}_i \right) + \Big(\underbrace{\Phi(D_0) - \Phi(D_n)}_{\text{net decrease in potential}} \Big)$$

$$\underbrace{\Phi(D_0) - \Phi(D_n)}_{\text{net decrease in potential}} \le \square \implies \boxed{\sum_{1 \le i \le n} c_i \le \left(\sum_{1 \le i \le n} \hat{c}_i \right) + \square}$$

$$\square = 0 \; (\forall i,\ \Phi(D_i) \ge \Phi(D_0)) \implies \forall n,\ \sum_{1 \le i \le n} c_i \le \sum_{1 \le i \le n} \hat{c}_i$$

$$\Phi(D_0) = 0, \quad \forall 1 \le i \le n : \ \Phi(D_i) \ge 0$$

$$\sum_{1 \le i \le n} c_i = \left(\sum_{1 \le i \le n} \hat{c}_i \right) + \Big(\underbrace{\Phi(D_0) - \Phi(D_n)}_{\text{net decrease in potential}} \Big)$$

$$\underbrace{\Phi(D_0) - \Phi(D_n)}_{\text{net decrease in potential}} \le \square \implies \boxed{\sum_{1 \le i \le n} c_i \le \left(\sum_{1 \le i \le n} \hat{c}_i \right) + \square}$$

$$\square = 0 \; (\forall i,\; \Phi(D_i) \ge \Phi(D_0)) \implies \forall n, \sum_{1 \le i \le n} c_i \le \sum_{1 \le i \le n} \hat{c}_i$$

$$\Phi(D_0) = 0, \quad \forall 1 \le i \le n : \; \Phi(D_i) \ge 0 \quad \text{(Typically)}$$

The Potential Method for Dynamic Tables

$$\boxed{\alpha = \frac{T.num}{T.size}}$$

The Potential Method for Dynamic Tables

$$\boxed{\alpha = \frac{T.num}{T.size}}$$

$$\textsc{Expansion}: \begin{cases} \text{When to expand?} \\ \text{How large to expand to?} \end{cases}$$

The Potential Method for Dynamic Tables

$$\boxed{\alpha = \frac{T.num}{T.size}}$$

$$\textsc{Expansion}: \begin{cases} \text{When to expand?} & \alpha = 1 \\ \text{How large to expand to?} & \alpha = 1/2 \end{cases}$$

The Potential Method for Dynamic Tables

$$\boxed{\alpha = \frac{T.num}{T.size}}$$

$$\text{EXPANSION} : \begin{cases} \text{When to expand?} & \alpha = 1 \\ \text{How large to expand to?} & \alpha = 1/2 \end{cases}$$

$$\text{CONTRACTION} : \begin{cases} \text{When to contract?} \\ \text{How small to contract to?} \end{cases}$$

The Potential Method for Dynamic Tables

$$\boxed{\alpha = \frac{T.num}{T.size}}$$

$$\textsc{Expansion} : \begin{cases} \text{When to expand?} & \alpha = 1 \\ \text{How large to expand to?} & \alpha = 1/2 \end{cases}$$

$$\textsc{Contraction} : \begin{cases} \text{When to contract?} & \alpha = 1/4 \\ \text{How small to contract to?} & \alpha = 1/2 \end{cases}$$

The Potential Method for Dynamic Tables

$$\boxed{\alpha = \frac{T.num}{T.size}}$$

$$\text{Expansion}: \begin{cases} \text{When to expand?} & \alpha = 1 \\ \text{How large to expand to?} & \alpha = 1/2 \end{cases}$$

$$\text{Contraction}: \begin{cases} \text{When to contract?} & \alpha = 1/4 \\ \text{How small to contract to?} & \alpha = 1/2 \end{cases}$$

$$\boxed{\frac{1}{4} \leq \alpha \leq 1}$$

$$\Phi(T) = \begin{cases} 2 \cdot T.num - T.size & \text{if } \alpha(T) \geq 1/2 \\ T.size/2 - T.num & \text{if } \alpha(T) < 1/2 \end{cases}$$

$$\Phi(T) = \begin{cases} 2 \cdot T.num - T.size & \text{if } \alpha(T) \geq 1/2 \\ T.size/2 - T.num & \text{if } \alpha(T) < 1/2 \end{cases}$$

$$\Phi(T_0) = 0, \quad \Phi(T_i) \geq 0$$

$$\Phi(T) = \begin{cases} 2 \cdot T.num - T.size & \text{if } \alpha(T) \geq 1/2 \\ T.size/2 - T.num & \text{if } \alpha(T) < 1/2 \end{cases}$$

$$\Phi(T_0) = 0, \quad \Phi(T_i) \geq 0$$

$$\alpha = 1/2 \implies \Phi(T) = 0$$

$$\Phi(T) = \begin{cases} 2 \cdot T.num - T.size & \text{if } \alpha(T) \geq 1/2 \\ T.size/2 - T.num & \text{if } \alpha(T) < 1/2 \end{cases}$$

$$\Phi(T_0) = 0, \quad \Phi(T_i) \geq 0$$

$$\alpha = 1/2 \implies \Phi(T) = 0$$

$$\alpha = 1/2 \rightsquigarrow \alpha = 1 \implies \Phi(T) : 0 \rightsquigarrow T.num$$

$$\Phi(T) = \begin{cases} 2 \cdot T.num - T.size & \text{if } \alpha(T) \geq 1/2 \\ T.size/2 - T.num & \text{if } \alpha(T) < 1/2 \end{cases}$$

$$\Phi(T_0) = 0, \quad \Phi(T_i) \geq 0$$

$$\alpha = 1/2 \implies \Phi(T) = 0$$

$$\alpha = 1/2 \leadsto \alpha = 1 \implies \Phi(T) : 0 \leadsto T.num$$

$$\alpha = 1/2 \leadsto \alpha = 1/4 \implies \Phi(T) : 0 \leadsto T.num$$

$$\Phi(T) = \begin{cases} 2 \cdot T.num - T.size & \text{if } \alpha(T) \geq 1/2 \\ T.size/2 - T.num & \text{if } \alpha(T) < 1/2 \end{cases}$$

$$\hat{c}_i = c_i + \Big(\Phi_i - \Phi_{i-1}\Big)$$

$$\Phi(T) = \begin{cases} 2 \cdot T.num - T.size & \text{if } \alpha(T) \geq 1/2 \\ T.size/2 - T.num & \text{if } \alpha(T) < 1/2 \end{cases}$$

$$\hat{c}_i = c_i + \Big(\Phi_i - \Phi_{i-1}\Big)$$

By Case Analysis.

$$\Phi(T) = \begin{cases} 2 \cdot T.num - T.size & \text{if } \alpha(T) \geq 1/2 \\ T.size/2 - T.num & \text{if } \alpha(T) < 1/2 \end{cases}$$

$$\hat{c}_i = c_i + \Big(\Phi_i - \Phi_{i-1}\Big)$$

By Case Analysis.

TABLE-INSERT

$$\begin{cases} \alpha_{i-1} < 1/2 \begin{cases} \alpha_i < 1/2 \\ \alpha_i \geq 1/2 \end{cases} \\ \alpha_{i-1} \geq 1/2 \begin{cases} \alpha_{i-1} < 1 \\ \alpha_{i-1} = 1 \end{cases} \end{cases}$$

$$\Phi(T) = \begin{cases} 2 \cdot T.num - T.size & \text{if } \alpha(T) \geq 1/2 \\ T.size/2 - T.num & \text{if } \alpha(T) < 1/2 \end{cases}$$

$$\hat{c}_i = c_i + \Big(\Phi_i - \Phi_{i-1}\Big)$$

By Case Analysis.

TABLE-INSERT

$$\begin{cases} \alpha_{i-1} < 1/2 \begin{cases} \alpha_i < 1/2 \\ \alpha_i \geq 1/2 \end{cases} \\ \alpha_{i-1} \geq 1/2 \begin{cases} \alpha_{i-1} < 1 \\ \alpha_{i-1} = 1 \end{cases} \end{cases}$$

TABLE-DELETE

$$\begin{cases} \alpha_{i-1} < 1/2 \begin{cases} \frac{num_{i-1} - 1}{size_{i-1}} \geq \frac{1}{4} \\ \frac{num_{i-1} - 1}{size_{i-1}} < \frac{1}{4} \end{cases} \\ \alpha_{i-1} \geq 1/2 \begin{cases} \alpha_i < 1/2 \\ \alpha_i \geq 1/2 \end{cases} \end{cases}$$

$$\Phi(T) = \begin{cases} 2 \cdot T.num - T.size & \text{if } \alpha(T) \ge 1/2 \\ T.size/2 - T.num & \text{if } \alpha(T) < 1/2 \end{cases}$$

$$\hat{c}_i = c_i + \Big(\Phi_i - \Phi_{i-1}\Big)$$

By Case Analysis.

TABLE-INSERT

$$\begin{cases} \alpha_{i-1} < 1/2 \begin{cases} \alpha_i < 1/2 \\ \alpha_i \ge 1/2 \end{cases} \\ \alpha_{i-1} \ge 1/2 \begin{cases} \alpha_{i-1} < 1 \\ \alpha_{i-1} = 1 \end{cases} \end{cases}$$

TABLE-DELETE

$$\begin{cases} \alpha_{i-1} < 1/2 \begin{cases} \frac{num_{i-1}-1}{size_{i-1}} \ge \frac{1}{4} \\ \frac{num_{i-1}-1}{size_{i-1}} < \frac{1}{4} \end{cases} \\ \alpha_{i-1} \ge 1/2 \begin{cases} \alpha_i < 1/2 \; \left(\frac{num_{i-1}-1}{size_{i-1}} < \frac{1}{4}?\right) \\ \alpha_i \ge 1/2 \end{cases} \end{cases}$$

TABLE-DELETE

$$\alpha_{i-1} < 1/2 \ \wedge \ \frac{num_{i-1} - 1}{size_{i-1}} \geq \frac{1}{4}$$

$$\hat{c}_i = c_i + \Big(\Phi_i - \Phi_{i-1}\Big)$$

TABLE-DELETE

$$\alpha_{i-1} < 1/2 \ \wedge \ \frac{num_{i-1} - 1}{size_{i-1}} \geq \frac{1}{4}$$

$$\begin{aligned}
\hat{c}_i &= c_i + \Big(\Phi_i - \Phi_{i-1}\Big) \\
&= 1 + (size_i/2 - num_i) - (size_{i-1}/2 - num_{i-1})
\end{aligned}$$

TABLE-DELETE

$$\alpha_{i-1} < 1/2 \;\wedge\; \frac{num_{i-1} - 1}{size_{i-1}} \geq \frac{1}{4}$$

$$\begin{aligned}
\hat{c}_i &= c_i + \Big(\Phi_i - \Phi_{i-1}\Big) \\
&= 1 + (size_i/2 - num_i) - (size_{i-1}/2 - num_{i-1}) \\
&= 1 + (size_i/2 - num_i) - (size_i/2 - (num_i + 1)) \\
&= 2
\end{aligned}$$

Table-Delete

$$\alpha_{i-1} < 1/2 \ \wedge \ \frac{num_{i-1} - 1}{size_{i-1}} \geq \frac{1}{4}$$

$$\begin{aligned}
\hat{c}_i &= c_i + \Big(\Phi_i - \Phi_{i-1}\Big) \\
&= 1 + (size_i/2 - num_i) - (size_{i-1}/2 - num_{i-1}) \\
&= 1 + (size_i/2 - num_i) - (size_i/2 - (num_i + 1)) \\
&= 2
\end{aligned}$$

TABLE-DELETE

$$\alpha_{i-1} \geq 1/2 \ \wedge \ \alpha_i \geq 1/2$$

$$\hat{c}_i = c_i + \Big(\Phi_i - \Phi_{i-1}\Big)$$

TABLE-DELETE

$$\alpha_{i-1} \geq 1/2 \ \wedge \ \alpha_i \geq 1/2$$

$$\begin{aligned} \hat{c}_i &= c_i + \Big(\Phi_i - \Phi_{i-1}\Big) \\ &= 1 + (2 \cdot num_i - size_i) - (2 \cdot num_{i-1} - size_{i-1}) \end{aligned}$$

TABLE-DELETE

$$\alpha_{i-1} \geq 1/2 \;\wedge\; \alpha_i \geq 1/2$$

$$\begin{aligned}
\hat{c}_i &= c_i + \Big(\Phi_i - \Phi_{i-1}\Big) \\
&= 1 + (2 \cdot num_i - size_i) - (2 \cdot num_{i-1} - size_{i-1}) \\
&= 1 + (2 \cdot num_i - size_i) - (2 \cdot (num_i + 1) - size_i) \\
&= {\color{red}-1}
\end{aligned}$$

Table-Delete

$$\alpha_{i-1} \geq 1/2 \ \wedge \ \alpha_i \geq 1/2$$

$$\begin{aligned}
\hat{c}_i &= c_i + \Big(\Phi_i - \Phi_{i-1}\Big) \\
&= 1 + (2 \cdot num_i - size_i) - (2 \cdot num_{i-1} - size_{i-1}) \\
&= 1 + (2 \cdot num_i - size_i) - (2 \cdot (num_i + 1) - size_i) \\
&= -1
\end{aligned}$$

TABLE-INSERT

$$\begin{cases} \alpha_{i-1} < 1/2 \begin{cases} \alpha_i < 1/2 \ (0) \\ \alpha_i \geq 1/2 \ (3) \end{cases} \\ \alpha_{i-1} \geq 1/2 \begin{cases} \alpha_{i-1} < 1 \ (3) \\ \alpha_{i-1} = 1 \ (3) \end{cases} \end{cases}$$

TABLE-DELETE

$$\begin{cases} \alpha_{i-1} < 1/2 \begin{cases} \frac{num_{i-1}-1}{size_{i-1}} \geq \frac{1}{4} \ (1) \\ \frac{num_{i-1}-1}{size_{i-1}} < \frac{1}{4} \ (2) \end{cases} \\ \alpha_{i-1} \geq 1/2 \begin{cases} \alpha_i < 1/2 \ (1/2) \\ \alpha_i \geq 1/2 \ (-1) \end{cases} \end{cases}$$

TABLE-INSERT

$$\begin{cases} \alpha_{i-1} < 1/2 & \begin{cases} \alpha_i < 1/2 \ \textcolor{red}{(0)} \\ \alpha_i \geq 1/2 \ \textcolor{red}{(3)} \end{cases} \\ \alpha_{i-1} \geq 1/2 & \begin{cases} \alpha_{i-1} < 1 \ \textcolor{red}{(3)} \\ \alpha_{i-1} = 1 \ \textcolor{red}{(3)} \end{cases} \end{cases}$$

TABLE-DELETE

$$\begin{cases} \alpha_{i-1} < 1/2 & \begin{cases} \frac{num_{i-1}-1}{size_{i-1}} \geq \frac{1}{4} \ \textcolor{red}{(1)} \\ \frac{num_{i-1}-1}{size_{i-1}} < \frac{1}{4} \ \textcolor{red}{(2)} \end{cases} \\ \alpha_{i-1} \geq 1/2 & \begin{cases} \alpha_i < 1/2 \ \textcolor{red}{(1/2)} \\ \alpha_i \geq 1/2 \ \textcolor{red}{(-1)} \end{cases} \end{cases}$$

The Summation Method for "Power of 2" (Problem 17.1-3)

$$c_i = \begin{cases} i & \text{if } i \text{ is an exact power of 2} \\ 1 & \text{o.w.} \end{cases}$$

o_i :	o_1	o_2	o_3	o_4	o_5	o_6	o_7	o_8	o_9	o_{10}
c_i :	1	2	1	4	1	1	1	8	1	1

The Summation Method for "Power of 2" (Problem 17.1-3)

$$c_i = \begin{cases} i & \text{if } i \text{ is an exact power of 2} \\ 1 & \text{o.w.} \end{cases}$$

o_i :	o_1	o_2	o_3	o_4	o_5	o_6	o_7	o_8	o_9	o_{10}
c_i :	1	2	1	4	1	1	1	8	1	1

$$\begin{aligned}
\sum_{i=1}^{n} c_i &= (n - \lfloor \log n \rfloor - 1) + \sum_{j=0}^{\lfloor \log n \rfloor} 2^j \\
&= (n - \lfloor \log n \rfloor - 1) + (2^{\lfloor \log n \rfloor + 1} - 1) \\
&\leq (n - \lfloor \log n \rfloor - 1) + (2n - 1) \\
&< 3n
\end{aligned}$$

The Summation Method for "Power of 2" (Problem 17.1-3)

$$c_i = \begin{cases} i & \text{if } i \text{ is an exact power of 2} \\ 1 & \text{o.w.} \end{cases}$$

o_i :	o_1	o_2	o_3	o_4	o_5	o_6	o_7	o_8	o_9	o_{10}
c_i :	1	2	1	4	1	1	1	8	1	1

$$\begin{aligned} \sum_{i=1}^{n} c_i &= (n - \lfloor \log n \rfloor - 1) + \sum_{j=0}^{\lfloor \log n \rfloor} 2^j \\ &= (n - \lfloor \log n \rfloor - 1) + (2^{\lfloor \log n \rfloor + 1} - 1) \\ &\le (n - \lfloor \log n \rfloor - 1) + (2n - 1) \\ &< 3n \end{aligned}$$

$$\boxed{\forall i,\ \hat{c}_i = 3}$$

The Accounting Method for "Power of 2" (Problem 17.2-2)

$$c_i = \begin{cases} i & \text{if } i \text{ is an exact power of 2} \\ 1 & \text{o.w.} \end{cases}$$

o_i :	o_1	o_2	o_3	o_4	o_5	o_6	o_7	o_8	o_9	o_{10}
c_i :	1	2	1	4	1	1	1	8	1	1

The Accounting Method for "Power of 2" (Problem 17.2-2)

$$c_i = \begin{cases} i & \text{if } i \text{ is an exact power of 2} \\ 1 & \text{o.w.} \end{cases}$$

o_i :	o_1	o_2	o_3	o_4	o_5	o_6	o_7	o_8	o_9	o_{10}
c_i :	1	2	1	4	1	1	1	8	1	1

$$\boxed{\forall i,\ \hat{c}_i = 3}$$

$$\hat{c}_i = c_i + a_i \implies a_i = 3 - c_i$$

The Accounting Method for "Power of 2" (Problem 17.2-2)

$$c_i = \begin{cases} i & \text{if } i \text{ is an exact power of 2} \\ 1 & \text{o.w.} \end{cases}$$

o_i :	o_1	o_2	o_3	o_4	o_5	o_6	o_7	o_8	o_9	o_{10}
c_i :	1	2	1	4	1	1	1	8	1	1
a_i :	2	1	2	-1	2	2	2	-5	2	2

$$\boxed{\forall i,\ \hat{c}_i = 3}$$

$$\hat{c}_i = c_i + a_i \implies a_i = 3 - c_i$$

The Accounting Method for "Power of 2" (Problem 17.2-2)

$$c_i = \begin{cases} i & \text{if } i \text{ is an exact power of } 2 \\ 1 & \text{o.w.} \end{cases}$$

o_i :	o_1	o_2	o_3	o_4	o_5	o_6	o_7	o_8	o_9	o_{10}
c_i :	1	2	1	4	1	1	1	8	1	1
a_i :	2	1	2	-1	2	2	2	-5	2	2

$$\boxed{\forall i,\ \hat{c}_i = 3}$$

$$\hat{c}_i = c_i + a_i \implies a_i = 3 - c_i$$

$$\forall n, \sum_{1 \le i \le n} a_i \ge 0.$$

The Accounting Method for "Power of 2" (Problem 17.2-2)

$$c_i = \begin{cases} i & \text{if } i \text{ is an exact power of 2} \\ 1 & \text{o.w.} \end{cases}$$

o_i :	o_1	o_2	o_3	o_4	o_5	o_6	o_7	o_8	o_9	o_{10}
c_i :	1	2	1	4	1	1	1	8	1	1
a_i :	2	1	2	-1	2	2	2	-5	2	2

$$\boxed{\forall i,\ \hat{c}_i = 3}$$

$$\hat{c}_i = c_i + a_i \implies a_i = 3 - c_i$$

$$\forall n, \sum_{1 \le i \le n} a_i \ge 0.$$

Prove by Mathematical Induction on n.

The Accounting Method for "Power of 2" (Problem 17.2-2)

$$c_i = \begin{cases} i & \text{if } i \text{ is an exact power of 2} \\ 1 & \text{o.w.} \end{cases}$$

o_i :	o_1	o_2	o_3	o_4	o_5	o_6	o_7	o_8	o_9	o_{10}
c_i :	1	2	1	4	1	1	1	8	1	1
a_i :	2	1	2	-1	2	2	2	-5	2	2

$$\boxed{\forall i,\ \hat{c}_i = 3}$$

$$2^k \quad (2^k, 2^{k+1}) \quad 2^{k+1}$$

$$\hat{c}_i = c_i + a_i \implies a_i = 3 - c_i$$

$$\forall n, \sum_{1 \le i \le n} a_i \ge 0.$$

Prove by Mathematical Induction on n.

The Accounting Method for "Power of 2" (Problem 17.2-2)

$$c_i = \begin{cases} i & \text{if } i \text{ is an exact power of 2} \\ 1 & \text{o.w.} \end{cases}$$

$$\begin{array}{ccccccccccc} o_i: & o_1 & o_2 & o_3 & o_4 & o_5 & o_6 & o_7 & o_8 & o_9 & o_{10} \\ c_i: & 1 & 2 & 1 & 4 & 1 & 1 & 1 & 8 & 1 & 1 \\ a_i: & 2 & 1 & 2 & -1 & 2 & 2 & 2 & -5 & 2 & 2 \end{array}$$

$$\boxed{\forall i,\ \hat{c}_i = 3}$$

$$2^k \quad (2^k, 2^{k+1}) \quad 2^{k+1}$$

$$\hat{c}_i = c_i + a_i \implies a_i = 3 - c_i$$

$$\Big(\sum_{1 \le i \le 2^k} a_i\Big) + 2(2^k - 1) + (3 - 2^{k+1}) \ge 0$$

$$\forall n, \sum_{1 \le i \le n} a_i \ge 0.$$

Prove by Mathematical Induction on n.

The Potential Method for "Power of 2" (Problem 17.1-3)

$$c_i = \begin{cases} i & \text{if } i \text{ is an exact power of 2} \\ 1 & \text{o.w.} \end{cases}$$

o_i :	o_1	o_2	o_3	o_4	o_5	o_6	o_7	o_8	o_9	o_{10}
c_i :	1	2	1	4	1	1	1	8	1	1
a_i :	2	1	2	-1	2	2	2	-5	2	2

The Potential Method for "Power of 2" (Problem 17.1-3)

$$c_i = \begin{cases} i & \text{if } i \text{ is an exact power of } 2 \\ 1 & \text{o.w.} \end{cases}$$

o_i :	o_1	o_2	o_3	o_4	o_5	o_6	o_7	o_8	o_9	o_{10}
c_i :	1	2	1	4	1	1	1	8	1	1
a_i :	2	1	2	-1	2	2	2	-5	2	2

$$\Phi(D_i) =$$

The Potential Method for "Power of 2" (Problem 17.1-3)

$$c_i = \begin{cases} i & \text{if } i \text{ is an exact power of 2} \\ 1 & \text{o.w.} \end{cases}$$

o_i :	o_1	o_2	o_3	o_4	o_5	o_6	o_7	o_8	o_9	o_{10}
c_i :	1	2	1	4	1	1	1	8	1	1
a_i :	2	1	2	-1	2	2	2	-5	2	2

$$\Phi(D_i) = \sum_{j=1}^{i} a_j$$

The Potential Method for "Power of 2" (Problem 17.1-3)

$$c_i = \begin{cases} i & \text{if } i \text{ is an exact power of 2} \\ 1 & \text{o.w.} \end{cases}$$

o_i :	o_1	o_2	o_3	o_4	o_5	o_6	o_7	o_8	o_9	o_{10}
c_i :	1	2	1	4	1	1	1	8	1	1
a_i :	2	1	2	-1	2	2	2	-5	2	2

$$\Phi(D_i) = \sum_{j=1}^{i} a_j = 2(i - \lfloor \log i \rfloor - 1) + \sum_{j=0}^{\lfloor \log i \rfloor} (3 - 2^j)$$

The Potential Method for "Power of 2" (Problem 17.1-3)

$$c_i = \begin{cases} i & \text{if } i \text{ is an exact power of } 2 \\ 1 & \text{o.w.} \end{cases}$$

o_i :	o_1	o_2	o_3	o_4	o_5	o_6	o_7	o_8	o_9	o_{10}
c_i :	1	2	1	4	1	1	1	8	1	1
a_i :	2	1	2	-1	2	2	2	-5	2	2

$$\begin{aligned}\Phi(D_i) = \sum_{j=1}^{i} a_j &= 2(i - \lfloor \log i \rfloor - 1) + \sum_{j=0}^{\lfloor \log i \rfloor} (3 - 2^j) \\ &= 2(i - 2^{\lfloor \log i \rfloor} + 1) + \lfloor \log i \rfloor\end{aligned}$$

The Potential Method for "Power of 2" (Problem 17.1-3)

$$c_i = \begin{cases} i & \text{if } i \text{ is an exact power of 2} \\ 1 & \text{o.w.} \end{cases}$$

o_i :	o_1	o_2	o_3	o_4	o_5	o_6	o_7	o_8	o_9	o_{10}
c_i :	1	2	1	4	1	1	1	8	1	1
a_i :	2	1	2	-1	2	2	2	-5	2	2

$$\begin{aligned}\Phi(D_i) = \sum_{j=1}^{i} a_j &= 2(i - \lfloor \log i \rfloor - 1) + \sum_{j=0}^{\lfloor \log i \rfloor} (3 - 2^j) \\ &= 2(i - 2^{\lfloor \log i \rfloor} + 1) + \lfloor \log i \rfloor\end{aligned}$$

$$\Phi(D_0) \triangleq 0, \quad \Phi(D_i) \geq 0$$

The Potential Method for "Power of 2" (Problem 17.1-3)

$$c_i = \begin{cases} i & \text{if } i \text{ is an exact power of 2} \\ 1 & \text{o.w.} \end{cases}$$

o_i :	o_1	o_2	o_3	o_4	o_5	o_6	o_7	o_8	o_9	o_{10}
c_i :	1	2	1	4	1	1	1	8	1	1
a_i :	2	1	2	-1	2	2	2	-5	2	2

$$\Phi(D_i) = \sum_{j=1}^{i} a_j = 2(i - \lfloor \log i \rfloor - 1) + \sum_{j=0}^{\lfloor \log i \rfloor} (3 - 2^j)$$

$$= 2(i - 2^{\lfloor \log i \rfloor} + 1) + \lfloor \log i \rfloor$$

$$\Phi(D_0) \triangleq 0, \quad \Phi(D_i) \geq 0$$

$$\hat{c}_i = c_i + \Big(\Phi(D_i) - \Phi(D_{i-1})\Big) = 3$$

Array Merging Dictionary (Additional Problem)

Array Merging Dictionary (Additional Problem)

i	s_i
A_0	1
A_1	2
A_2	4
A_3	8
$\vdots$	$\ldots$
A_i	2^i

Array Merging Dictionary (Additional Problem)

i	s_i
A_0	1
A_1	2
A_2	4
A_3	8
$\vdots$	$\ldots$
A_i	2^i

$$11 = 2^0 + 2^1 + 2^3$$

i	e_i
A_0	$[5]$
A_1	$[4, 8]$
A_2	$[\ ]$
A_3	$[2, 6, 9, 12, 13, 16, 20, 25]$

Array Merging Dictionary (Additional Problem)

i	s_i
A_0	1
A_1	2
A_2	4
A_3	8
$\vdots$	$\dots$
A_i	2^i

$$11 = 2^0 + 2^1 + 2^3$$

i	e_i
A_0	$[5]$
A_1	$[4, 8]$
A_2	$[\,]$
A_3	$[2, 6, 9, 12, 13, 16, 20, 25]$

$$\textsc{Create} : 1 \quad \textsc{Merge}(A_i, A_i) : 2 \cdot 2^i$$

Array Merging Dictionary (Additional Problem)

i	s_i
A_0	1
A_1	2
A_2	4
A_3	8
$\vdots$	$\ldots$
A_i	2^i

$$11 = 2^0 + 2^1 + 2^3$$

i	e_i
A_0	$[5]$
A_1	$[4, 8]$
A_2	$[\,]$
A_3	$[2, 6, 9, 12, 13, 16, 20, 25]$

$$\textsc{Create} : 1 \quad \textsc{Merge}(A_i, A_i) : 2 \cdot 2^i$$

$\textsc{Insert}(10) : 1 + 2 + 4;$

Array Merging Dictionary (Additional Problem)

i	s_i
A_0	1
A_1	2
A_2	4
A_3	8
$\vdots$	$\dots$
A_i	2^i

$$11 = 2^0 + 2^1 + 2^3$$

i	e_i
A_0	$[5]$
A_1	$[4, 8]$
A_2	$[\,]$
A_3	$[2, 6, 9, 12, 13, 16, 20, 25]$

$$\textsc{Create} : 1 \quad \textsc{Merge}(A_i, A_i) : 2 \cdot 2^i$$

$$\textsc{Insert}(10) : 1 + 2 + 4; \quad \textsc{Insert}() : 1;$$

Array Merging Dictionary (Additional Problem)

i	s_i
A_0	1
A_1	2
A_2	4
A_3	8
$\vdots$	$\dots$
A_i	2^i

$$11 = 2^0 + 2^1 + 2^3$$

i	e_i
A_0	$[5]$
A_1	$[4, 8]$
A_2	$[\ ]$
A_3	$[2, 6, 9, 12, 13, 16, 20, 25]$

$$\textsc{Create} : 1 \quad \textsc{Merge}(A_i, A_i) : 2 \cdot 2^i$$

$$\textsc{Insert}(10) : 1 + 2 + 4; \quad \textsc{Insert}() : 1; \quad \textsc{Insert}() : 1 + 2$$

The Summation Method for "Array Merging Dictionary"

$$\textsc{Create} : 1 \quad \textsc{Merge}(A_i, A_i) : 2 \cdot 2^i$$

The Summation Method for "Array Merging Dictionary"

$$\textsc{Create} : 1 \quad \textsc{Merge}(A_i, A_i) : 2 \cdot 2^i$$

i	c_i
1	1
2	$1+2$
3	1
4	$1+2+4$
5	1
6	$1+2$
7	1
8	$1+2+4$
$\vdots$	$\ldots$

The Summation Method for "Array Merging Dictionary"

$$\textsc{Create} : 1 \quad \textsc{Merge}(A_i, A_i) : 2 \cdot 2^i$$

i	c_i
1	1
2	$1+2$
3	1
4	$1+2+4$
5	1
6	$1+2$
7	1
8	$1+2+4$
$\vdots$	$\dots$

$$\sum_{i=1}^{n} c_i = \sum_{j=0}^{\lfloor \log n \rfloor} \lfloor \frac{n}{2^j} \rfloor 2^j \le n(\lfloor \log n \rfloor + 1)$$

The Summation Method for "Array Merging Dictionary"

$$\textsc{Create} : 1 \quad \textsc{Merge}(A_i, A_i) : 2 \cdot 2^i$$

i	c_i
1	1
2	$1+2$
3	1
4	$1+2+4$
5	1
6	$1+2$
7	1
8	$1+2+4$
$\vdots$	$\ldots$

$$\sum_{i=1}^{n} c_i = \sum_{j=0}^{\lfloor \log n \rfloor} \lfloor \frac{n}{2^j} \rfloor 2^j \le n(\lfloor \log n \rfloor + 1)$$

$$\forall i, \ \hat{c}_i = 1 + \lfloor \log n \rfloor$$

The Accounting Method for "Array Merging Dictionary"

$$\textsc{Create} : 1 \quad \textsc{Merge}(A_i, A_i) : 2 \cdot 2^i$$

The Accounting Method for "Array Merging Dictionary"

$$\textsc{Create}: 1 \quad \textsc{Merge}(A_i, A_i): 2 \cdot 2^i$$

$$\hat{c}_i = 1 + \lfloor \log n \rfloor$$

The Accounting Method for "Array Merging Dictionary"

$$\textsc{Create} : 1 \quad \textsc{Merge}(A_i, A_i) : 2 \cdot 2^i$$

$$\hat{c}_i = 1 + \lfloor \log n \rfloor$$

The Accounting Method for "Array Merging Dictionary"

$$\textsc{Create} : 1 \quad \textsc{Merge}(A_i, A_i) : 2 \cdot 2^i$$

$$\hat{c}_i = 1 + \lfloor \log n \rfloor$$

$$\forall n, \ \sum_{i=1}^{n} a_i \geq 0$$

The Potential Method for "Array Merging Dictionary"

$$\text{Create} : 1 \quad \text{Merge}(A_i, A_i) : 2 \cdot 2^i$$

The Potential Method for "Array Merging Dictionary"

$$\textsc{Create}: 1 \quad \textsc{Merge}(A_i, A_i): 2 \cdot 2^i$$

$$j = \sum_{i=1}^{k} 2^{x_i} \implies$$

The Potential Method for "Array Merging Dictionary"

$$\textsc{Create} : 1 \quad \textsc{Merge}(A_i, A_i) : 2 \cdot 2^i$$

$$j = \sum_{i=1}^{k} 2^{x_i} \implies \boxed{\Phi(D_j) = \sum_{i=1}^{k} 2^{x_i} \Big(\lfloor \log n \rfloor - x_i \Big)}$$

The Potential Method for "Array Merging Dictionary"

$$\textsc{Create}: 1 \quad \textsc{Merge}(A_i, A_i): 2 \cdot 2^i$$

$$j = \sum_{i=1}^{k} 2^{x_i} \implies \boxed{\Phi(D_j) = \sum_{i=1}^{k} 2^{x_i} \Big(\lfloor \log n \rfloor - x_i \Big)}$$

$$\textsc{Insert}_j : A_0, A_1, \cdots, A_t \rightsquigarrow A_{t+1}$$

The Potential Method for "Array Merging Dictionary"

$$\textsc{Create} : 1 \quad \textsc{Merge}(A_i, A_i) : 2 \cdot 2^i$$

$$j = \sum_{i=1}^{k} 2^{x_i} \implies \boxed{\Phi(D_j) = \sum_{i=1}^{k} 2^{x_i} \Big(\lfloor \log n \rfloor - x_i \Big)}$$

$$\textsc{Insert}_j : A_0, A_1, \cdots, A_t \rightsquigarrow A_{t+1}$$

$$\hat{c}_j = c_j + \Big(\Phi(D_j) - \Phi(D_{j-1}) \Big)$$

The Potential Method for “Array Merging Dictionary”

$$\textsc{Create} : 1 \quad \textsc{Merge}(A_i, A_i) : 2 \cdot 2^i$$

$$j = \sum_{i=1}^{k} 2^{x_i} \implies \boxed{\Phi(D_j) = \sum_{i=1}^{k} 2^{x_i} \Big(\lfloor \log n \rfloor - x_i \Big)}$$

$$\textsc{Insert}_j : A_0, A_1, \cdots, A_t \rightsquigarrow A_{t+1}$$

$$\hat{c}_j = c_j + \Big(\Phi(D_j) - \Phi(D_{j-1}) \Big)$$

$$= 1 + \sum_{i=0}^{t} 2^{i+1} - \Big(\sum_{i=0}^{t} 2^i (\lfloor \log n \rfloor - i) \Big) + 2^{t+1} \Big(\lfloor \log n \rfloor - (t+1) \Big)$$

The Potential Method for "Array Merging Dictionary"

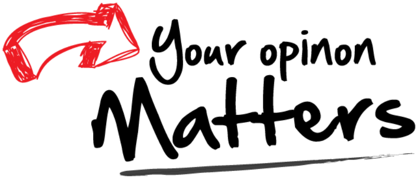
$$\textsc{Create} : 1 \quad \textsc{Merge}(A_i, A_i) : 2 \cdot 2^i$$

$$j = \sum_{i=1}^{k} 2^{x_i} \implies \boxed{\Phi(D_j) = \sum_{i=1}^{k} 2^{x_i}\Big(\lfloor \log n \rfloor - x_i\Big)}$$

$$\textsc{Insert}_j : A_0, A_1, \cdots, A_t \rightsquigarrow A_{t+1}$$

$$\begin{aligned}\hat{c}_j &= c_j + \Big(\Phi(D_j) - \Phi(D_{j-1})\Big) \\ &= 1 + \sum_{i=0}^{t} 2^{i+1} - \Big(\sum_{i=0}^{t} 2^i(\lfloor \log n \rfloor - i)\Big) + 2^{t+1}\Big(\lfloor \log n \rfloor - (t+1)\Big) \\ &= 1 + \lfloor \log n \rfloor\end{aligned}$$


THANK YOU



Office 302

Mailbox: H016

hfwei@nju.edu.cn