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

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

$$\sum$$

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

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

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

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

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

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.

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

The Summation Method for Dynamic Tables

On any sequence of n Table-Insert on an *initially empty* array.

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

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

The Summation Method for Dynamic Tables

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

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

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

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

The Summation Method for Dynamic Tables

On any sequence of n Table-Insert on an *initially empty* array.

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

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

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

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

The Accounting Method

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

TABLE-DELETE

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

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

TABLE-DELETE

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

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

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

$$
\begin{cases}
\alpha_{i-1} < 1/2 & \begin{cases} \alpha_i < 1/2 \; (0) \\ \alpha_i \ge 1/2 \; (3) \end{cases} \\[2ex]
\alpha_{i-1} \ge 1/2 & \begin{cases} \alpha_{i-1} < 1 \; (3) \\ \alpha_{i-1} = 1 \; (3) \end{cases}
\end{cases}
$$

Table-Delete

$$
\begin{cases}
\alpha_{i-1} < 1/2 & \begin{cases} \frac{num_{i-1}-1}{size_{i-1}} \ge \frac{1}{4} \; (1) \\ \frac{num_{i-1}-1}{size_{i-1}} < \frac{1}{4} \; (2) \end{cases} \\[2ex]
\alpha_{i-1} \ge 1/2 & \begin{cases} \alpha_i < 1/2 \; (1/2) \\ \alpha_i \ge 1/2 \; (-1) \end{cases}
\end{cases}
$$

What work are you proudest of?

What work are you proudest of?

Proudest? It's hard to choose.

What work are you proudest of?

Proudest? It's hard to choose.

*I like the **self-adjusting search tree** data structure*

that Danny Sleator and I developed.

Self-Adjusting Binary Search Trees

DANIEL DOMINIC SLEATOR AND ROBERT ENDRE TARJAN

AT&T Bell Laboratories, Murray Hill, NJ

Abstract. The *splay* tree, a self-adjusting form of binary search tree, is developed and analyzed. The binary search tree is a data structure for representing tables and lists so that accessing, inserting, and deleting items is easy. On an n-node splay tree, all the standard search tree operations have an amortized time bound of $O(\log n)$ per operation, where by "amortized time" is meant the time per operation averaged over a worst-case sequence of operations. Thus splay trees are as efficient as balanced trees when total running time is the measure of interest. In addition, for sufficiently long access sequences, splay trees are as efficient, to within a constant factor, as static optimum search trees. The efficiency of splay trees comes not from an explicit structural constraint, as with balanced trees, but from applying a simple restructuring heuristic, called *splaying*, whenever the tree is accessed. Extensions of splaying give simplified forms of two other data structures: lexicographic or multidimensional search trees and link/cut trees.

"Self-Adjusting Binary Search Trees – Splay Tree", JACM, 1985

Self-Adjusting Binary Search Trees

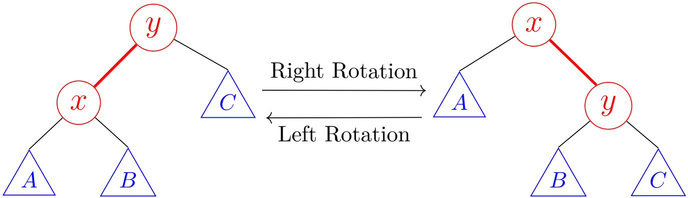
vs. Balanced Binary Search Trees

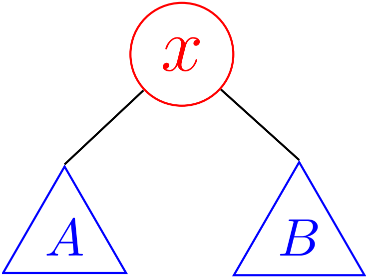
$\textsc{Splay}(x)$:

Moving node x to the root of the tree by performing a sequence of rotations along the path from x to the root.

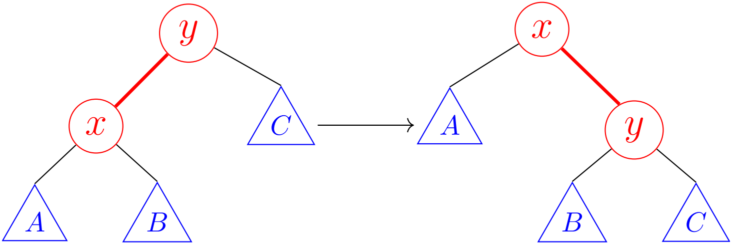
$\textsc{Splay}(x)$:

Moving node x to the root of the tree by performing a sequence of rotations along the path from x to the root.



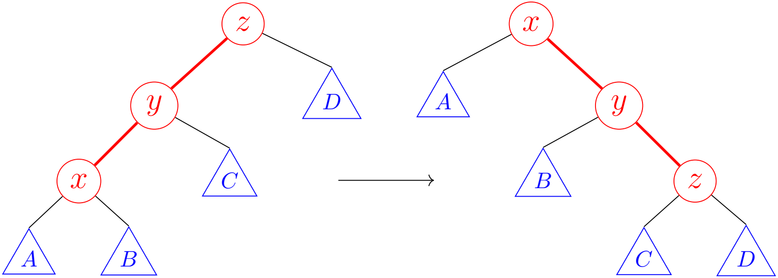


CASE 0: x is the root



CASE 1: zig

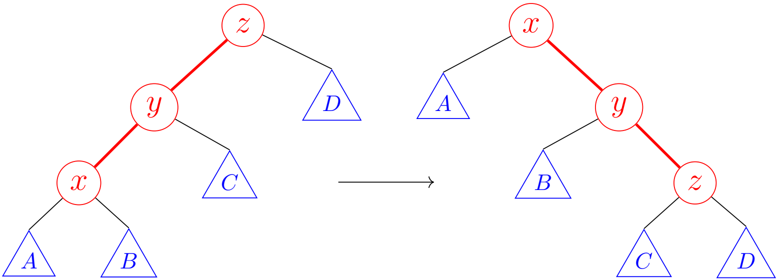
$y = p(x)$ is the root



CASE 2: zig-zig

$$y = p(x) \qquad z = p(y)$$

$$x = lc(y) \qquad y = lc(z)$$

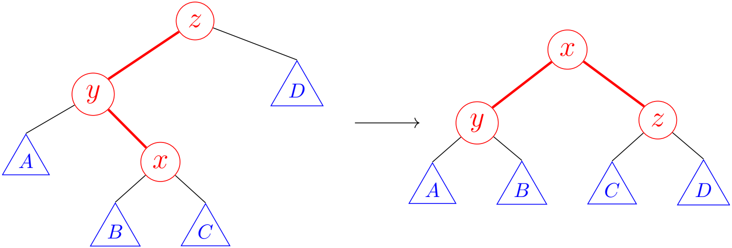


Case 2: zig-zig

$$y = p(x) \qquad z = p(y)$$

$$x = lc(y) \qquad y = lc(z)$$

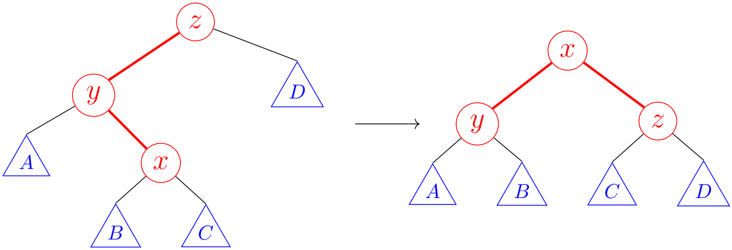
$$(1) : y - z \qquad (2) : x - y$$



Case 3: zig-zag

$$y = p(x) \qquad z = p(y)$$

$$x = rc(y) \qquad y = lc(z)$$

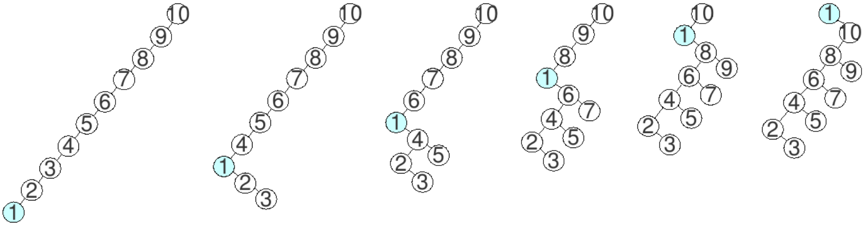


CASE 3: zig-zag

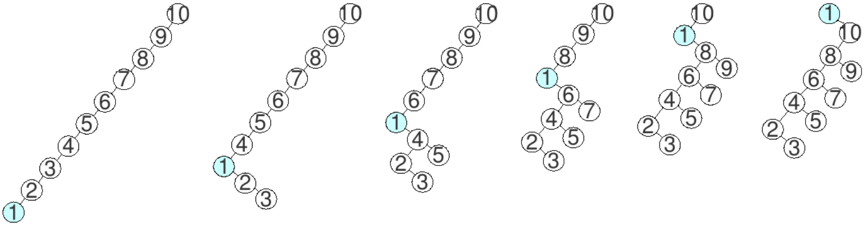
$$y = p(x) \qquad z = p(y)$$

$$x = rc(y) \qquad y = lc(z)$$

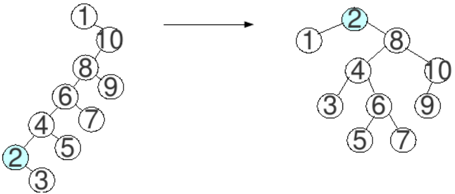
$$(1): x - y \qquad (2): x - z$$



SPLAY(1)



SPLAY(1)



SPLAY(2)

Amortized analysis of SPLAY

Amortized analysis of Splay

A splay tree T of n-node

An arbitrary sequence of m Splay operations

Amortized analysis of $\textsc{Splay}$

A splay tree T of n-node

An arbitrary sequence of m $\textsc{Splay}$ operations

$\#$ of rotations

Amortized analysis of SPLAY

A splay tree T of n-node

An arbitrary sequence of m SPLAY operations

$\#$ of rotations

Theorem

$$\hat{c}_{\text{SPLAY}} = O(\log n).$$

$$\Phi_0 \;\; \textsc{Splay}_1 \;\; \Phi_1 \;\; \textsc{Splay}_2 \;\; \Phi_2 \;\; \cdots \;\; \underbrace{\Phi_{i-1} \;\; \textsc{Splay}_i \;\; \Phi_i}_{\text{the } i\text{-th } \textsc{Splay}} \cdots \;\; \textsc{Splay}_m \;\; \Phi_m$$

$$\hat{c}_{\textsc{Splay}_i} = c_{\textsc{Splay}_i} + \left(\Phi_{\textsc{Splay}_i} - \Phi_{\textsc{Splay}_{i-1}}\right)$$

$$\Phi_0 \ \textsc{Splay}_1 \ \Phi_1 \ \textsc{Splay}_2 \ \Phi_2 \ \cdots \ \underbrace{\Phi_{i-1} \ \textsc{Splay}_i \ \Phi_i}_{\text{the } i\text{-th } \textsc{Splay}} \cdots \ \textsc{Splay}_m \ \Phi_m$$

$$\hat{c}_{\textsc{Splay}_i} = c_{\textsc{Splay}_i} + \left(\Phi_{\textsc{Splay}_i} - \Phi_{\textsc{Splay}_{i-1}}\right)$$

How to define Φ?

$s(x)$: # of nodes in the subtree rooted at x

$s(x)$: # of nodes in the subtree rooted at x

$r(x) = \log s(x)$

$s(x)$: # of nodes in the subtree rooted at x

$$r(x) = \log s(x)$$

$$\Phi = \sum_{x \in T} r(x)$$

$s(x)$: # of nodes in the subtree rooted at x

$$r(x) = \log s(x)$$

$$\Phi = \sum_{x \in T} r(x)$$

$s(x)$: # of nodes in the subtree rooted at x

$r(x) = \log s(x)$

$\Phi = \sum_{x \in T} r(x)$

$$\hat{c}_{\text{SPLAY}_i} = c_{\text{SPLAY}_i} + (\Phi_{\text{SPLAY}_i} - \Phi_{\text{SPLAY}_{i-1}})$$

How to calculate $(\Phi_{\text{SPLAY}_i} - \Phi_{\text{SPLAY}_{i-1}})$ *and* c_{SPLAY_i}*?*

$$\Phi_0 \ \ \textsc{Splay}_1 \ \ \Phi_1 \ \ \textsc{Splay}_2 \ \ \Phi_2 \ \cdots \ \underbrace{\Phi_{i-1} \ \ \textsc{Splay}_i \ \ \Phi_i}_{\text{the } i\text{-th } \textsc{Splay}} \cdots \ \textsc{Splay}_m \ \ \Phi_m$$

$$\Phi_0 \;\; \textsc{Splay}_1 \;\; \Phi_1 \;\; \textsc{Splay}_2 \;\; \Phi_2 \;\cdots\; \underbrace{\Phi_{i-1} \;\; \textsc{Splay}_i \;\; \Phi_i}_{\text{the } i\text{-th } \textsc{Splay}} \cdots\; \textsc{Splay}_m \;\; \Phi_m$$

$$\underbrace{\Phi_{i-1} \;\; \textsc{Splay}_i \;\; \Phi_i}_{\text{the } i\text{-th } \textsc{Splay}} :$$

$$\Phi_{i-1} \;\triangleq\; \Phi_{0'} \; \textsc{Iter}_1 \; \Phi_{1'} \;\cdots\; \underbrace{\Phi_{k-1} \; \textsc{Iter}_k \; \Phi_k}_{\text{the } k\text{-th } \textsc{Iteration}} \cdots\; \textsc{Iter}_l \; \Phi_l \;\triangleq\; \Phi_i$$

$$\Phi_0 \;\; \textsc{Splay}_1 \;\; \Phi_1 \;\; \textsc{Splay}_2 \;\; \Phi_2 \; \cdots \; \underbrace{\Phi_{i-1} \;\; \textsc{Splay}_i \;\; \Phi_i}_{\text{the } i\text{-th } \textsc{Splay}} \cdots \; \textsc{Splay}_m \;\; \Phi_m$$

$$\underbrace{\Phi_{i-1} \;\; \textsc{Splay}_i \;\; \Phi_i}_{\text{the } i\text{-th } \textsc{Splay}} :$$

$$\Phi_{i-1} \;\triangleq\; \Phi_{0'} \; \textsc{Iter}_1 \; \Phi_{1'} \; \cdots \; \underbrace{\Phi_{k-1} \; \textsc{Iter}_k \; \Phi_k}_{\text{the } k\text{-th } \textsc{Iteration}} \cdots \; \textsc{Iter}_l \; \Phi_l \;\triangleq\; \Phi_i$$

$$\begin{aligned} \hat{c}_{\textsc{Splay}_i} &= \sum_{1 \le j \le l} \hat{c}_{\textsc{Iter}_j} \\ &= \sum_{1 \le j \le l} c_{\textsc{Iter}_j} + \left(\Phi_{\textsc{Iter}_j} - \Phi_{\textsc{Iter}_{j-1}} \right) \end{aligned}$$

$$\hat{c}_{\text{ITER}_j} = c_{\text{ITER}_j} + \left(\Phi_{\text{ITER}_j} - \Phi_{\text{ITER}_{j-1}}\right)$$

$$\hat{c}_{\textsc{Iter}_j} = c_{\textsc{Iter}_j} + \left(\Phi_{\textsc{Iter}_j} - \Phi_{\textsc{Iter}_{j-1}}\right)$$

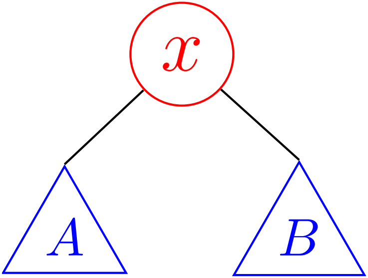
By Case Analysis.

$$\hat{c}_{\textsc{Iter}_j} = c_{\textsc{Iter}_j} + \left(\Phi_{\textsc{Iter}_j} - \Phi_{\textsc{Iter}_{j-1}}\right)$$

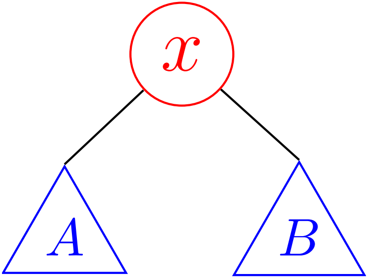
By Case Analysis.

$$\hat{c}_j = c_j + \left(\Phi_j - \Phi_{j-1}\right)$$

Remember: ITER

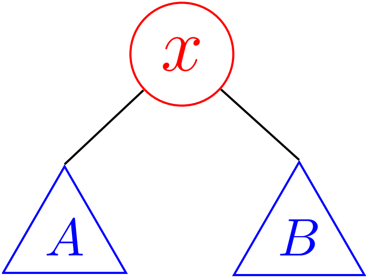


CASE 0



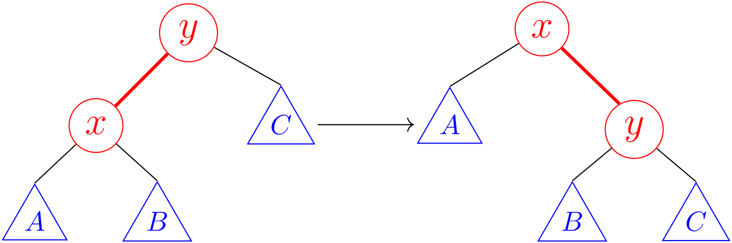
Case 0

$$\hat{c}_j = c_j + (\Phi_j - \Phi_{j-1})$$



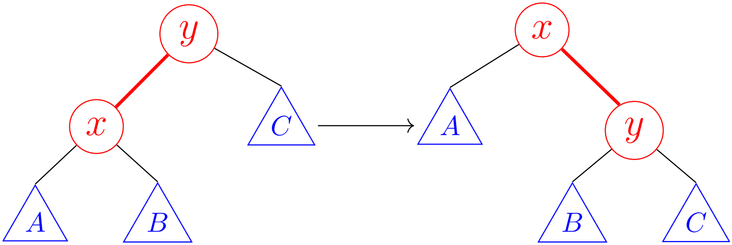
CASE 0

$$
\begin{aligned}
\hat{c}_j &= c_j + (\Phi_j - \Phi_{j-1}) \\
&= 0 + 0 \\
&= 0
\end{aligned}
$$



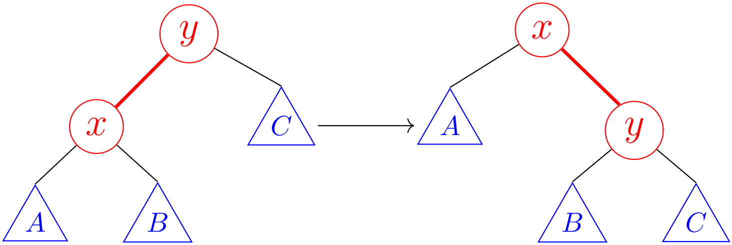
CASE 1: zig

$$\hat{c}_j = c_j + (\Phi_j - \Phi_{j-1})$$



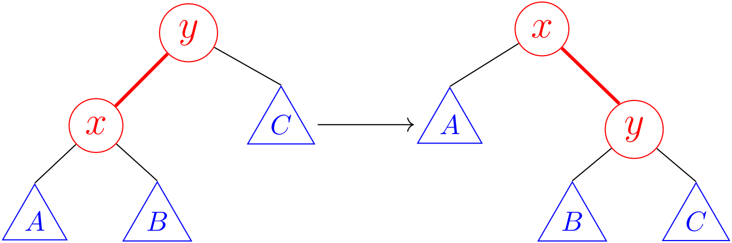
CASE 1: zig

$$\begin{aligned}\hat{c}_j &= c_j + (\Phi_j - \Phi_{j-1}) \\ &= 1 + r_j(x) + r_j(y) - r_{j-1}(x) - r_{j-1}(y)\end{aligned}$$



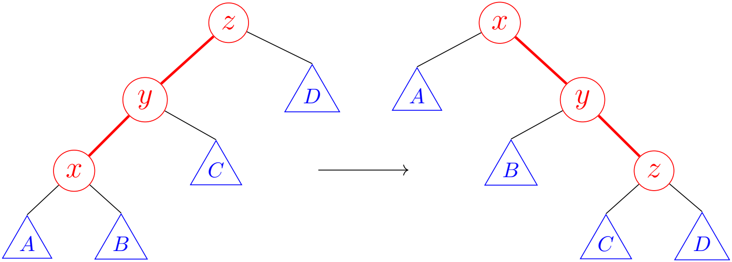
CASE 1: zig

$$
\begin{aligned}
\hat{c}_j &= c_j + (\Phi_j - \Phi_{j-1}) \\
&= 1 + r_j(x) + r_j(y) - r_{j-1}(x) - r_{j-1}(y) \\
&\leq 1 + r_j(x) - r_{j-1}(x)
\end{aligned}
$$



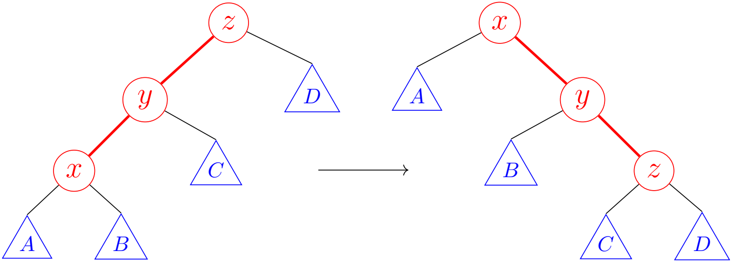
CASE 1: zig

$$
\begin{aligned}
\hat{c}_j &= c_j + (\Phi_j - \Phi_{j-1}) \\
&= 1 + r_j(x) + r_j(y) - r_{j-1}(x) - r_{j-1}(y) \\
&\leq 1 + r_j(x) - r_{j-1}(x) \\
&\leq 1 + 3\big(r_j(x) - r_{j-1}(x)\big)
\end{aligned}
$$



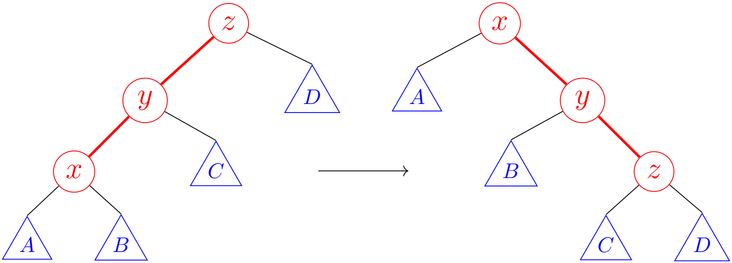
Case 2: zig-zig

$$\hat{c}_j = c_j + (\Phi_j - \Phi_{j-1})$$



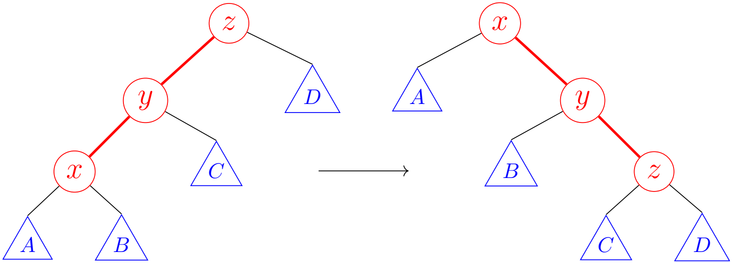
Case 2: zig-zig

$$
\begin{aligned}
\hat{c}_j &= c_j + (\Phi_j - \Phi_{j-1}) \\
&= 2 + r_j(x) + r_j(y) + r_j(y) - r_{j-1}(x) - r_{j-1}(y) - r_{j-1}(z)
\end{aligned}
$$



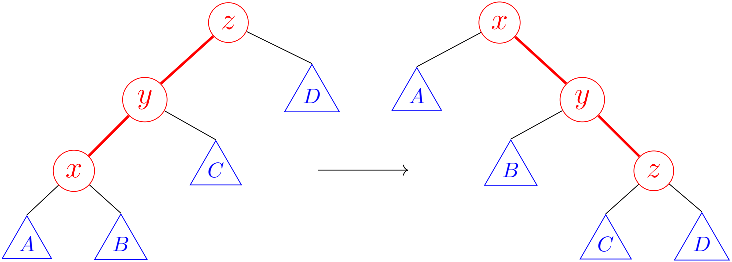
CASE 2: zig-zig

$$
\begin{aligned}
\hat{c}_j &= c_j + (\Phi_j - \Phi_{j-1}) \\
&= 2 + r_j(x) + r_j(y) + r_j(y) - r_{j-1}(x) - r_{j-1}(y) - r_{j-1}(z) \\
&= 2 + r_j(y) + r_j(y) - r_{j-1}(x) - r_{j-1}(y)
\end{aligned}
$$



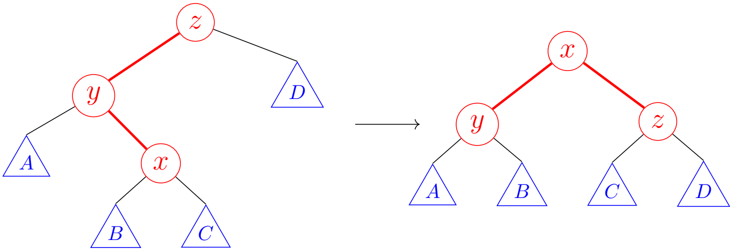
CASE 2: zig-zig

$$
\begin{aligned}
\hat{c}_j &= c_j + (\Phi_j - \Phi_{j-1}) \\
&= 2 + r_j(x) + r_j(y) + r_j(y) - r_{j-1}(x) - r_{j-1}(y) - r_{j-1}(z) \\
&= 2 + r_j(y) + r_j(y) - r_{j-1}(x) - r_{j-1}(y) \\
&\leq 2 + r_j(x) + r_j(z) - 2r_{j-1}(x)
\end{aligned}
$$



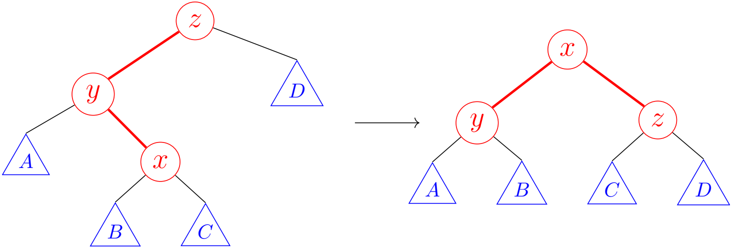
CASE 2: zig-zig

$$\begin{aligned}
\hat{c}_j &= c_j + (\Phi_j - \Phi_{j-1}) \\
&= 2 + r_j(x) + r_j(y) + r_j(y) - r_{j-1}(x) - r_{j-1}(y) - r_{j-1}(z) \\
&= 2 + r_j(y) + r_j(y) - r_{j-1}(x) - r_{j-1}(y) \\
&\leq 2 + r_j(x) + r_j(z) - 2r_{j-1}(x) \\
&\leq 3(r_j(x) - r_{j-1}(x))
\end{aligned}$$



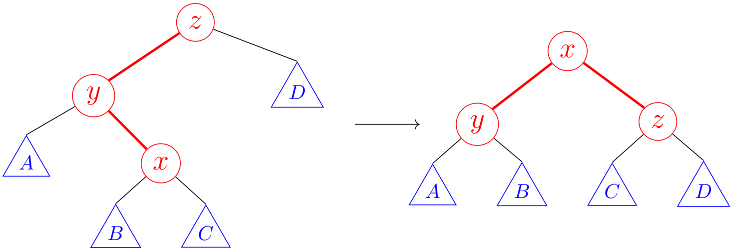
Case 3: zig-zag

$$\hat{c}_j = c_j + (\Phi_j - \Phi_{j-1})$$



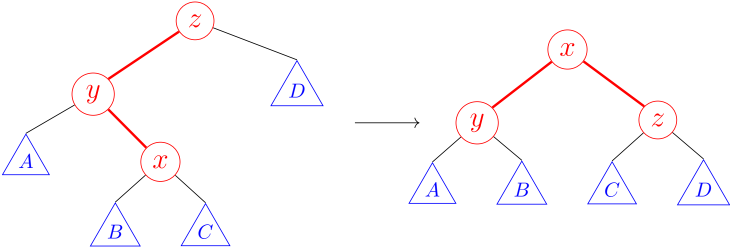
Case 3: zig-zag

$$
\begin{aligned}
\hat{c}_j &= c_j + (\Phi_j - \Phi_{j-1}) \\
&= 2 + r_j(x) + r_j(y) + r_j(y) - r_{j-1}(x) - r_{j-1}(y) - r_{j-1}(z)
\end{aligned}
$$



CASE 3: zig-zag

$$\begin{aligned}\hat{c}_j &= c_j + (\Phi_j - \Phi_{j-1}) \\ &= 2 + r_j(x) + r_j(y) + r_j(y) - r_{j-1}(x) - r_{j-1}(y) - r_{j-1}(z) \\ &\leq 2 + r_j(y) + r_j(z) - 2r_{j-1}(x)\end{aligned}$$



CASE 3: zig-zag

$$
\begin{aligned}
\hat{c}_j &= c_j + (\Phi_j - \Phi_{j-1}) \\
&= 2 + r_j(x) + r_j(y) + r_j(y) - r_{j-1}(x) - r_{j-1}(y) - r_{j-1}(z) \\
&\leq 2 + r_j(y) + r_j(z) - 2r_{j-1}(x) \\
&\leq 3\big(r_j(x) - r_{j-1}(x)\big)
\end{aligned}
$$

$$\hat{c}_{\textsc{Iter}_j} \le \begin{cases} 0, & \textsc{Case } 0 \\ 1 + 3\big(r_j(x) - r_{j-1}(x)\big), & \textsc{Case } 1 \\ 3\big(r_j(x) - r_{j-1}(x)\big), & \textsc{Case } 2 \\ 3\big(r_j(x) - r_{j-1}(x)\big), & \textsc{Case } 3 \end{cases}$$

$$\hat{c}_{\text{ITER}_j} \le \begin{cases} 0, & \text{CASE } 0 \\ 1 + 3\big(r_j(x) - r_{j-1}(x)\big), & \text{CASE } 1 \\ 3\big(r_j(x) - r_{j-1}(x)\big), & \text{CASE } 2 \\ 3\big(r_j(x) - r_{j-1}(x)\big), & \text{CASE } 3 \end{cases}$$

$$\begin{aligned} \hat{c}_{\text{SPLAY}_i} &= \sum_{1 \le j \le l} \hat{c}_{\text{ITER}_j} \\ &= \sum_{1 \le j \le l} c_{\text{ITER}_j} + \big(\Phi_{\text{ITER}_j} - \Phi_{\text{ITER}_{j-1}}\big) \end{aligned}$$

$$\hat{c}_{\textsc{Iter}_j} \leq \begin{cases} 0, & \textsc{Case } 0 \\ 1 + 3\big(r_j(x) - r_{j-1}(x)\big), & \textsc{Case } 1 \\ 3\big(r_j(x) - r_{j-1}(x)\big), & \textsc{Case } 2 \\ 3\big(r_j(x) - r_{j-1}(x)\big), & \textsc{Case } 3 \end{cases}$$

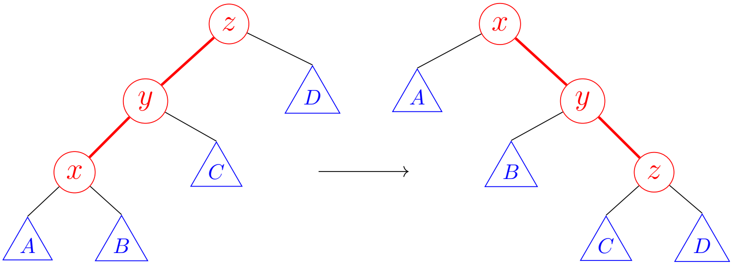
$$\begin{aligned} \hat{c}_{\textsc{Splay}_i} &= \sum_{1 \leq j \leq l} \hat{c}_{\textsc{Iter}_j} \\ &= \sum_{1 \leq j \leq l} c_{\textsc{Iter}_j} + \left(\Phi_{\textsc{Iter}_j} - \Phi_{\textsc{Iter}_{j-1}}\right) \\ &\leq 3\big(r_{\textsc{Iter}_l}(x) - r_{\textsc{Iter}_0}(x)\big) \color{red}{+1} \end{aligned}$$

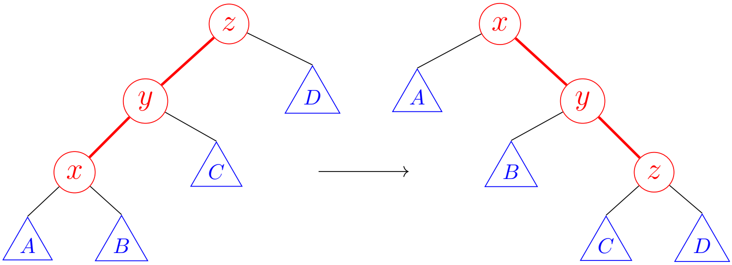
$$\hat{c}_{\text{ITER}_j} \le \begin{cases} 0, & \text{CASE } 0 \\ 1 + 3\big(r_j(x) - r_{j-1}(x)\big), & \text{CASE } 1 \\ 3\big(r_j(x) - r_{j-1}(x)\big), & \text{CASE } 2 \\ 3\big(r_j(x) - r_{j-1}(x)\big), & \text{CASE } 3 \end{cases}$$

$$\begin{aligned} \hat{c}_{\text{SPLAY}_i} &= \sum_{1 \le j \le l} \hat{c}_{\text{ITER}_j} \\ &= \sum_{1 \le j \le l} c_{\text{ITER}_j} + \big(\Phi_{\text{ITER}_j} - \Phi_{\text{ITER}_{j-1}}\big) \\ &\le 3\big(r_{\text{ITER}_l}(x) - r_{\text{ITER}_0}(x)\big) + 1 \\ &= 3\big(\log n - r_{\text{ITER}_0}(x)\big) + 1 \end{aligned}$$

$$\hat{c}_{\text{ITER}_j} \le \begin{cases} 0, & \text{CASE } 0 \\ 1 + 3(r_j(x) - r_{j-1}(x)), & \text{CASE } 1 \\ 3(r_j(x) - r_{j-1}(x)), & \text{CASE } 2 \\ 3(r_j(x) - r_{j-1}(x)), & \text{CASE } 3 \end{cases}$$

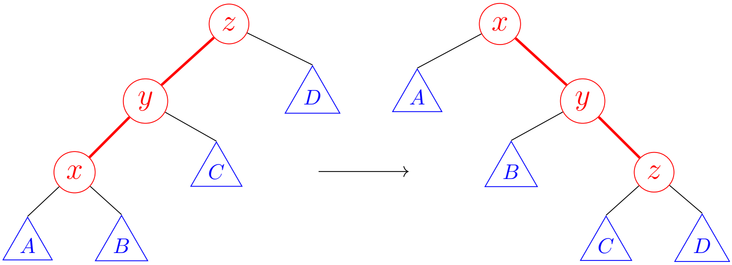
$$\begin{aligned} \hat{c}_{\text{SPLAY}_i} &= \sum_{1 \le j \le l} \hat{c}_{\text{ITER}_j} \\ &= \sum_{1 \le j \le l} c_{\text{ITER}_j} + (\Phi_{\text{ITER}_j} - \Phi_{\text{ITER}_{j-1}}) \\ &\le 3(r_{\text{ITER}_l}(x) - r_{\text{ITER}_0}(x)) \; {\color{red}+1} \\ &= 3(\log n - r_{\text{ITER}_0}(x)) + 1 \\ &\le 3 \log n + 1 \\ &= O(\log n) \end{aligned}$$


z
y
D
x
C
A
B
x
A
y
B
z
C
D



MTR (Move To Root) heuristic:

Keeping rotate the edge joining x to its parent.



MTR (Move To Root) heuristic:

Keeping rotate the edge joining x to its parent.

Does this work?

$$\Phi = \sum_{x \in T} r(x)$$

$$\Phi = \sum_{x \in T} r(x)$$

$\textsc{Splay}(x)$

$\textsc{Splay}(x)$

$\textsc{Search}(x, t)$

$\textsc{Insert}(x, t)$

$\textsc{Delete}(x, t)$

$\textsc{Join}(t_1, t_2)$

$\textsc{Split}(x, t)$

$\textsc{Splay}(x)$

$\textsc{Search}(x, t)$

$\textsc{Insert}(x, t)$

$\textsc{Delete}(x, t)$

$\textsc{Join}(t_1, t_2)$

$\textsc{Split}(x, t)$

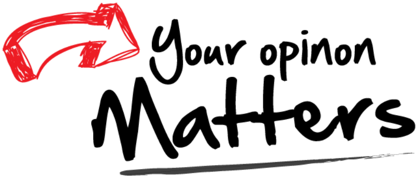
Self-Adjusting Binary Search Trees

DANIEL DOMINIC SLEATOR AND ROBERT ENDRE TARJAN

AT&T Bell Laboratories, Murray Hill, NJ

Abstract. The *splay* tree, a self-adjusting form of binary search tree, is developed and analyzed. The binary search tree is a data structure for representing tables and lists so that accessing, inserting, and deleting items is easy. On an *n*-node splay tree, all the standard search tree operations have an amortized time bound of $O(\log n)$ per operation, where by "amortized time" is meant the time per operation averaged over a worst-case sequence of operations. Thus splay trees are as efficient as balanced trees when total running time is the measure of interest. In addition, for sufficiently long access sequences, splay trees are as efficient, to within a constant factor, as static optimum search trees. The efficiency of splay trees comes not from an explicit structural constraint, as with balanced trees, but from applying a simple restructuring heuristic, called *splaying*, whenever the tree is accessed. Extensions of splaying give simplified forms of two other data structures: lexicographic or multidimensional search trees and link/cut trees.


THANK YOU



Office 302

Mailbox: H016

hfwei@nju.edu.cn