Amortization Analysis

16

Amortized Analysis

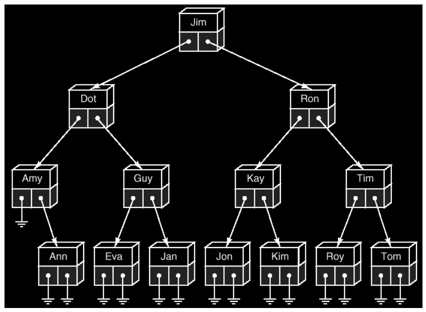
DESIGN & ALALYSIS OF ALGORITHM

- When one event in a sequence affects the cost of later events:
  - One particular task may be expensive.
  - But it may leave data structure in a state that next few tasks becomes easier.
- Example:
  - Analysis of single sort? (Quick sort may be better)
  - Analysis of a continual sort? (Quick sort may be worst)

LECT-11, S-17
ALG00S, javed@kent.edu

Amortized Cost of Tree Traversal

- Consider in-order traversal of BT with n nodes:
  - cost is number of links visited to reach the vertex from the last vertex visited.



Amortized Cost:

2(n-1)/n < 2

Best-case=1 (child-to-parent)

worst-case=1-n (parent-to-left child in a lefti-chain BT)

What is the amortized cost?

DESIGN & ALALYSIS OF ALGORITHM

Amortized Cost of Incrementing Binary Integers

- The actual cost varies from step to step.
  - So we will use a cerdit-balance function which smooths out the cost.
- Choose the credit-balance function c_i so as to make the amortized costs a_i as nearly equal as possible, no matter how the actual costs t_i may vary.

step i	*integer*	t_i	c_i	a_i
0	0 0 0 0		0	
1	0 0 0 1	1	1	2
2	0 0 1 0	2	1	2
3	0 0 1 1	1	2	2
4	0 1 0 0	3	1	2
5	0 1 0 1	1	2	2
6	0 1 1 0	2	2	2
7	0 1 1 1	1	3	2
8	1 0 0 0	4	1	2
9	1 0 0 1	1	2	2
10	1 0 1 0	2	2	2
11	1 0 1 1	1	3	2
12	1 1 0 0	3	2	2
13	1 1 0 1	1	3	2
14	1 1 1 0	2	3	2
15	1 1 1 1	1	4	2
16	0 0 0 0	4	0	0

t_i = actual cost = number of digits changed
c_i = credit-balance function = number of 1's in integer
a_i = amortized cost = $t_i + c_i - c_{i-1}$

Amortized Analysis

DEFINITION The ***amortized cost*** a_i of each operation is defined to be $a_i = t_i + c_i - c_{i-1}$ for $i = 1, 2, \ldots, m$, where t_i is the actual cost and c_i is a credit balance.

Choose the credit-balance function c_i so as to make the amortized costs a_i as nearly equal as possible, no matter how the actual costs t_i may vary.

LEMMA 9.5 The total actual cost and total amortized cost of a sequence of m operations on a data structure are related by

$$\sum_{i=1}^{m} t_i = \left(\sum_{i=1}^{m} a_i\right) + c_0 - c_m.$$

Cost of One Insertion/ Retrieval in Splay Tree

- One Insertion can result in a series of zig-zag, zig-zig, zag-zag, zag, zig operations.
- We will assume a credit-balance function for each of these operations.
- Based on it we will try to estimate the cost of m operations which makes one insertion by evaluating: $\left(\sum_{i=1}^{m} a_i\right)$ and $c_0 - c_m.$
- and then: $$\sum_{i=1}^{m} t_i = \left(\sum_{i=1}^{m} a_i\right) + c_0 - c_m.$$
- Finally we will estimate the cost of m sequential insertion/searches.

Today's Math:

Geometric Mean is Smaller than Arithmetic Mean

DESIGN & ALALYSIS OF ALGORITHM

LEMMA 9.6 If α, β, and γ are positive real numbers with $\alpha + \beta \leq \gamma$, then $\lg \alpha + \lg \beta \leq 2 \lg \gamma - 2$.

$$(\sqrt{\alpha} - \sqrt{\beta})^2 \geq 0$$

$$\alpha + \beta \geq 2\sqrt{\alpha.\beta}$$

$$2\log(\alpha + \beta) \geq 2.\log 2 + \log \alpha + \log \beta$$

$$2\log \gamma - 2 \geq \log \alpha + \log \beta$$

ALG00S, javed@kent.edu

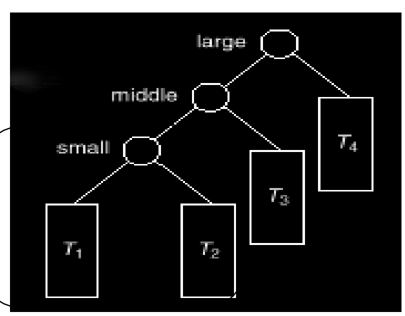
Credit Balance Function for Splaying

DESIGN & ALALYSIS OF ALGORITHM

Let T be a binary search tree, T_i be T as it is after step i of splaying, $T_i(x)$ be the subtree with root x in T_i, $|T_i(x)|$ be the number of nodes in $T_i(x)$, and define the ***rank*** of x to be $r_i(x) = \lg |T_i(x)|$.

The Credit Invariant

For every node x of T and after every step i of splaying, node x has credit equal to its rank $r_i(x)$.



T(x)

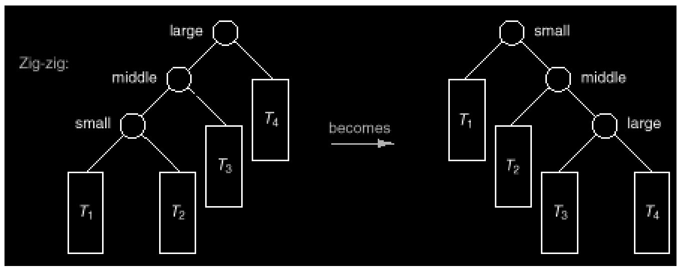
r(x)=0 when the tree has only one node.

ALG00S, javed@kent.edu

Amortization Cost in Zig-Zag or Zag-Zag

DESIGN & ALALYSIS OF ALGORITHM

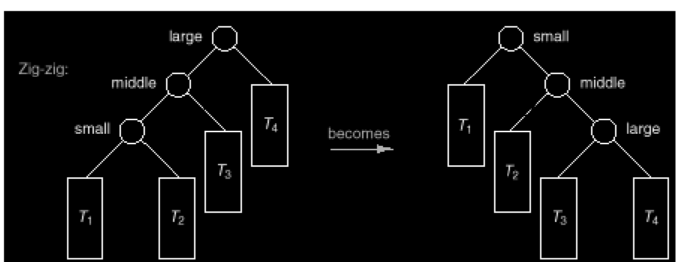
LEMMA 9.7 If the i^{th} splaying step is a zig-zig or zag-zag step at node x, then its amortized complexity a_i satisfies the inequality $a_i < 3\big(r_i(x) - r_{i-1}(x)\big)$.



$$a_i = t_i + c_i - c_{i-1}$$
$$= 2 + r_i(x) + r_i(y) + r_i(z) - r_{i-1}(x) - r_{i-1}(y) - r_{i-1}(z)$$
$$= 2 + r_i(y) + r_i(z) - r_{i-1}(x) - r_{i-1}(y)$$

Amortization Cost in Zig-Zag or Zag-Zag

DESIGN & ALALYSIS OF ALGORITHM



$$a_i = t_i + c_i - c_{i-1}$$
$$= 2 + r_i(x) + r_i(y) + r_i(z) - r_{i-1}(x) - r_{i-1}(y) - r_{i-1}(z)$$
$$= 2 + r_i(y) + r_i(z) - r_{i-1}(x) - r_{i-1}(y)$$

$r_{i-1}(y) \le r_{i-1}(x)$

$$A = |T_{i-1}(x)|,\ B = |T_i(z)|,\ and\ C = |T_i(x)|$$
$$A + B < C$$
$$r_{i-1}(x) + r_i(z) \le 2r_i(x) - 2$$
$$r_i(z) \le 2r_i(x) - 2 - r_{i-1}(x)$$
$$or\ a_i < 3r_i(x) - 3r_{i-1}(x)$$

LEMMA 9.6 If α, β, and γ are positive real numbers with $\alpha + \beta \le \gamma$, then $\lg\alpha + \lg\beta \le 2\lg\gamma - 2$.

Amortization Cost for Other Operations

DESIGN & ALALYSIS OF ALGORITHM

Lemma 9.7 If the i^{th} splaying step is a zig-zig or zag-zag step at node x, then its amortized complexity a_i satisfies the inequality $a_i < 3\big(r_i(x) - r_{i-1}(x)\big)$.

Lemma 9.8 If the i^{th} splaying step is a zig-zag or zag-zig step at node x, then its amortized complexity a_i satisfies

$$a_i < 2\big(r_i(x) - r_{i-1}(x)\big).$$

Lemma 9.9 If the i^{th} splaying step is a zig or a zag step at node x, then its amortized complexity a_i satisfies

$$a_i < 1 + \big(r_i(x) - r_{i-1}(x)\big).$$

Total Amortization Cost for one Retrieval/ Insertion

DESIGN & ALALYSIS OF ALGORITHM

$$\sum_{i=1}^{m} a_i = \sum_{i=1}^{m-1} a_i + a_m$$

$$\leq \sum_{i=1}^{m-1} (3r_i(x) - 3r_{i-1}(x)) + (1 + 3r_m(x) - 3r_{m-1}(x))$$

$$= 1 + 3r_m(x) - 3r_0(x)$$

$$\leq 1 + 3r_m(x)$$

$$= 1 + 3\log n$$

Theorem 9.10 The amortized cost of an insertion or retrieval with splaying in a binary search tree with n nodes does not exceed $1 + 3\lg n$ upward moves of the target node in the tree.

Amortization cost of a sequence of m insertions/retrievals

DESIGN & ALALYSIS OF ALGORITHM

- The total actual cost of a sequence of m splay differs from the total amortizatized cost only by c_0-c_m.
- C_m is at most log n.

LEMMA 9.5 The total actual cost and total amortized cost of a sequence of m operations on a data structure are related by

$$\sum_{i=1}^{m} t_i = \left(\sum_{i=1}^{m} a_i\right) + c_0 - c_m.$$

COROLLARY 9.11 The total complexity of a sequence of m insertions or retrievals with splaying in a binary search tree which never has more than n nodes does not exceed

$$m(1 + 3\lg n) + \lg n$$

upward moves of a target node in the tree.

Next Class

Heap Sort
Priority Queue