Dynamic Tables

Amortized Analyses: Dynamic Table

- A nice use of amortized analysis
- Table-insertion, table-deletion.
- Scenario:
  - A table –maybe a hash table
  - Do not know how large in advance
  - May expend with insertion
  - May contract with deletion
  - Detailed implementation is not important
- Goal:
  - $O(1)$ amortized cost.
  - Unused space always $\leq$ constant fraction of allocated space.

Dynamic Table

- ***Load factor*** $\alpha = num/size$, where num = # items stored, $size$ = allocated size.
- If $size = 0$, then $num = 0$. Call $\alpha = 1$.
- Never allow $\alpha > 1$.
- Keep α >a constant fraction → goal (2).

Dynamic Table: Expansion with Insertion

- **Table expansion**
- Consider only insertion.
- When the table becomes full, double its size and reinsert all existing items.
- Guarantees that $\alpha \geq 1/2$.
- Each time we actually insert an item into the table, it's an ***elementary insertion***.

```
TABLE-INSERT(T, x)
if size[T] = 0
   then allocate table[T] with 1 slot
        size[T] ← 1
if num[T] = size[T]
   then allocate new-table with 2 · size[T] slots
        insert all items in table[T] into new-table
        free table[T]
        table[T] ← new-table
        size[T] ← 2 · size[T]
insert x into table[T]
num[T] ← num[T] + 1
```

Num[t] ele. insertion

1 ele. insertion

Initially, *num*[*T*] = *size*[*T*] = 0.

Aggregate Analysis

- ***Running time:*** Charge 1 per elementary insertion. Count only elementary insertions,
- since all other costs together are constant per call.
- ci = actual cost of ith operation
  - If not full, $ci = 1$.
  - If full, have $i - 1$ items in the table at the start of the ith operation. Have to copy all $i - 1$ existing items, then insert ith item, $\Rightarrow ci = i$
- *Cursory analysis:* n operations $\Rightarrow ci = O(n) \Rightarrow O(n^2)$ time for n operations.
- Of course, we don't always expand:
  - $ci = \begin{cases} i & \text{if } i - 1 \text{ is exact power of } 2\,, \\ 1 & \text{otherwise}\,. \end{cases}$
- *So total cost* $= \sum_{i=1}^{n} ci \le n + \sum_{i=0}^{\log(n)} 2^i \le n + 2n = 3n$
- Therefore, **aggregate analysis** says amortized cost per operation = 3.

Accounting Analysis

- Charge \$3 per insertion of x.
  - \$1 pays for x's insertion.
  - \$1 pays for x to be moved in the future.
  - \$1 pays for some other item to be moved.
- Suppose we've just expanded, $size = m$ before next expansion, $size = 2m$ after next expansion.
- Assume that the expansion used up all the credit, so that there's no credit stored after the expansion.
- Will expand again after another m insertions.
- Each insertion will put \$1 on one of the m items that were in the table just after expansion and will put \$1 on the item inserted.
- Have \$$2m$ of credit by next expansion, when there are $2m$ items to move. Just enough to pay for the expansion, with no credit left over!

Potential Method

- **Potential method**
- $\Phi(T) = 2 \cdot num[T] - size[T]$
- Initially, $num = size = 0 \Rightarrow \Phi = 0$.
- • Just after expansion, $size = 2 \cdot num \Rightarrow \Phi = 0$.
- Just before expansion, $size = num \Rightarrow \Phi = num \Rightarrow$ have enough potential to pay for moving all items.
- Need $\Phi \geq 0$, always.
- Always have
  - $size \geq num \geq ½\, size \Rightarrow 2 \cdot num \geq size \Rightarrow \Phi \geq 0$.

Potential Method

- ***Amortized cost of ith operation:***
  - $num_i = num$ after ith operation ,
  - $size_i = size$ after ith operation ,
  - $\Phi_i = \Phi$ after ith operation .
- If no expansion:
  - $size_i = size_{i-1}$,
  - $num_i = num_{i-1} + 1$,
  - $ci = 1$.
- Then we have
  - $C_i' = c_i + \Phi_i - \Phi_{i-1} = 1 + (2num_i - size_i) - (2num_{i-1} - size_{i-1}) = 3.$
- If expansion:
  - $size_i = 2size_{i-1}$,
  - $size_{i-1} = num_{i-1} = num_i - 1$,
  - $c_i = num_{i-1} + 1 = num_i.$
- Then we have
- $C_i' = c_i + \Phi_i - \Phi_{i-1} = num_i + (2num_i - size_i) - (2num_{i-1} - size_{i-1}) = num_i + (2num_i - 2(num_i - 1)) - (2(num_i - 1) - (num_i - 1)) = num_i + 2 - (num_i - 1) = 3$

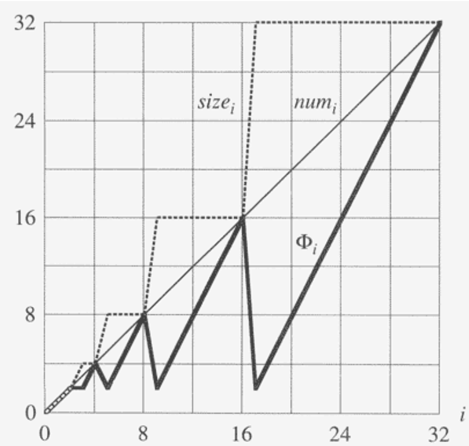

Figure 17.3 The effect of a sequence of n TABLE-INSERT operations on the number num_i of items in the table, the number $size_i$ of slots in the table, and the potential $\Phi_i = 2 \cdot num_i - size_i$, each being measured after the ith operation. The thin line shows num_i, the dashed line shows $size_i$, and the thick line shows Φ_i. Notice that immediately before an expansion, the potential has built up to the number of items in the table, and therefore it can pay for moving all the items to the new table. Afterwards, the potential drops to 0, but it is immediately increased by 2 when the item that caused the expansion is inserted.

Expansion and Contraction

- **Expansion and contraction**
- When α drops too low, contract the table.
  - Allocate a new, smaller one.
  - Copy all items.
- Still want
  - α bounded from below by a constant,
  - amortized cost per operation = $O(1)$.
- Measure cost in terms of elementary insertions and deletions.

Obvious strategy

- Double size when inserting into a full table (when $\alpha = 1$, so that after insertion α would become <1).
- Halve size when deletion would make table less than half full (when $\alpha = 1/2$, so that after deletion α would become >= 1/2).
- Then always have $1/2 \le \alpha \le 1$.
- Suppose we fill table.
  - Then insert $\Rightarrow$ double
  - 2 deletes $\Rightarrow$ halve
  - 2 inserts $\Rightarrow$ double
  - 2 deletes $\Rightarrow$ halve
  - . . .
  - Cost of each expansion or contraction is $\Theta(n)$, so total n operation will be $\Theta(n^2)$.
- Problem is that: Not performing enough operations after expansion or contraction to pay for the next one.

Simple Solution

- Double as before: when inserting with $\alpha = 1 \Rightarrow$ after doubling, $\alpha = 1/2$.
- Halve size when deleting with $\alpha = 1/4 \Rightarrow$ after halving, $\alpha = 1/2$.
- Thus, immediately after either expansion or contraction, have $\alpha = 1/2$.
- Always have $1/4 \leq \alpha \leq 1$.
- ***Intuition:***
- Want to make sure that we perform enough operations between consecutive expansions/contractions to pay for the change in table size.
- Need to delete half the items before contraction.
- Need to double number of items before expansion.
- Either way, number of operations between expansions/contractions is at least a constant fraction of number of items copied.

Potential function

- $\Phi(T) = \quad 2num[T] - size[T]$ if $\alpha \geq ½$
  $size[T]/2 - num[T]$ if $\alpha < ½$.
- T empty $\Rightarrow \Phi = 0$.
- $\alpha \geq 1/2 \Rightarrow num \geq 1/2 size \Rightarrow 2num \geq size \Rightarrow \Phi \geq 0$.
- $\alpha < 1/2 \Rightarrow num < 1/2 size \Rightarrow \Phi \geq 0$.

Intuition

- measures how far from $\alpha = 1/2$ we are.
  - $\alpha = 1/2 \Rightarrow \Phi = 2num - 2num = 0$.
  - $\alpha = 1 \Rightarrow \Phi = 2num - num = num$.
  - $\alpha = 1/4 \Rightarrow \Phi = size/2 - num = 4num/2 - num = num$.
- Therefore, when we double or halve, have enough potential to pay for moving all *num* items.
- Potential increases linearly between $\alpha = 1/2$ and $\alpha = 1$, and it also increases linearly between $\alpha = 1/2$ and $\alpha = 1/4$.
- Since α has different distances to go to get to 1 or 1/4, starting from 1/2, rate of increase differs.
- For α to go from 1/2 to 1, *num* increases from *size* /2 to *size*, for a total increase of *size* /2. Φ increases from 0 to *size*. Thus, Φ needs to increase by 2 for each item inserted. That's why there's a coefficient of 2 on the $num[T]$ term in the formula for when $\alpha \geq 1/2$.
- For α to go from 1/2 to 1/4, *num* decreases from *size* /2 to *size* /4, for a total decrease of *size* /4. Φ increases from 0 to *size* /4. Thus, Φ needs to increase by 1 for each item deleted. That's why there's a coefficient of −1 on the $num[T]$ term in the formula for when $\alpha < 1/2$.

Amortized Cost for Each Operation

- Amortized costs: more cases
  - insert, delete
  - $\alpha \geq 1/2$, $\alpha < 1/2$ (use α_i, since α can vary a lot)
  - *size* does/doesn't change

The End