Insert/Delete into a table (array)

How do you fix the size of the array when knowing how much data you will see.

- Table is too small - Reallocation
- Table is too big - Wasting memory

Reallocation ⇒ Re-allocating all data \( \Rightarrow \)

Time proportional to the data you have
Strategy when full - double size

Some inserts $O(1)$ time.

Some inserts are $O(\text{size}(T))$ time.

$O(n)$
Let $C_i = \text{cost of } i^{\text{th}} \text{ insertion}$

$$C_i = \begin{cases} 
i & \text{if } i \text{ is a power of 2} \\ 1 & \text{otherwise} \end{cases}$$

$$\sum_{i=1}^{n} C_i = \sum_{i=1}^{\log n} 2^i + \sum_{j=0}^{n} 2^j$$
$$\leq n + (1 + 2 + 4 + \cdots + 2^n)$$
$$\leq n + 2n = 3n$$

Each insertion is $O(1)$ amortized time.
Each insert should build up credit (potential) potential should be used when doing a reallocation.

\[
\phi \begin{cases} 
\text{num} = \text{size}/2 & \phi = 0 \\
\text{num} \leq \text{size} & \phi \leq \text{num}
\end{cases}
\]

\[
\phi(T) = 2\text{num}(t) - \text{size}(T)
\]
\[ \Phi(T) = 2 \text{num}(T) - \text{size}(T) \]

Show: \[ \hat{c}_c = c_i + \Delta \Phi \leq 3 \]

Case 1: table size does not change

\[ \hat{c}_c = 1 + 2 \text{num}_i - \text{size}_c - 2 \text{num}_{i-1} + \text{size}_{i-1} \]

\[ \text{size}_c = \text{size}_{i-1} \]

\[ \text{num}_i = \text{num}_{i-1} + 1 \]

\[ = 1 + 2 (\text{num}_{i-1} + 1) - 2 \text{num}_{i-1} \]

\[ = 3 \]
case 2: table size does change

\[ \hat{C}_i = C_i + \Delta \Phi_i \]

\[ \quad = \text{num}_i + 2 \text{num}_i - \text{size}_i - 2 \text{num}_{i-1} + \text{size}_{i-1} \]

\[ \quad = 3(\text{num}_{i-1} + 1) - 2 \text{size}_{i-1} - 2 \text{num}_{i-1} + \text{size}_{i-1} \]

\[ \quad = \text{num}_{i-1} + 3 - \text{size}_{i-1} \]

\[ \quad = 3 \]
Deletion

Shrink the table if
num < size

Idea

double when full

quarter

halfsize when table is half full

alt. ms., del, this makes every op expensive
\[ C \]

\[ \frac{1}{2} \text{ full} \quad \frac{1}{4} \text{ full} \quad \text{full} \]

\text{Amortized}

\[ O(1) \text{ time/operation Ins.} \& \text{Delete} \]
Disjoint Sets

Items: $X=\{A, B, C, D, E, F, G, H\}$

Maintain sets $S_1, \ldots, S_k$

Each $x \in X$ is in exactly one set $S_i$

$S_i \cap S_j = \emptyset \quad \forall i, j, i \neq j$

Make Set $(x)$ - makes a one element set

Find $(x)$ - return the "name" of the set $x$ is in

Union $(x, y)$ - merges $x$'s set and $y$'s set
maintain equivalence relation

Network "is connected to"

\[ A \quad B \quad C \quad D \quad E \quad F \quad G \quad H \]

A - B - C - D - E - F - G - H
<table>
<thead>
<tr>
<th>Items name</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>R</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>G</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>

Find $O(1)$
Union $O(n)$
name: last elt.
Find: O(n)
Union: add 1 ptr.
but have to find
beginning and end
of list

Add ptr to end of list
Find: O(1)  Union: O(n)
Union by size
(put shorter list before the longer list)

Union: time = |Shorter list|

D→D
D→D→D→D→... →D

Union has $O(\log n)$ amortized time
smaller set

\[ \text{Union}_1, A \]
\[ \text{Union}_2, B \]
\[ \text{Union}_3, C \cap D \]
\[ \text{Union}_4, B \cap D \]
\[ \text{Union}_{n-1} \]

How many times can one element be in the smaller set of a union?
A \setminus B

A \setminus B \subseteq C \cup D

\text{every time an elt. is in the smaller side of a union, its set size at least doubles}

\Rightarrow \text{smallest side } \leq \log n \text{ times}

\text{Total time for all unions is } O(n \log n)
Find $O(1)$

Union $O(\log n)$
Find: $O(\lg n)$
Union: $O(1) + \text{Find}$

$O(\lg^* n)$