Dynamic Tables

table `T`, $|T|$ denotes max number of items that can be stored

Operations

- `insert(T, x)`
- `delete(T, x)`

let $n = \#\text{ of items currently stored in }T$, let $\alpha(T) = \frac{n}{|T|}$

`insert` using table doubling

if `T = [a, b, c, d]` and $|T| = 4$, then `insert(T, e)` results in `T = [a, b, c, d, e, _, _, _]` and $|T| = 8$

- cost is $\mathcal{O}(|T|)$ because every element must be copied

Aggregate analysis

Suppose n items need to be inserted into empty table, then following table doubles happen:

`T = [a]`

`T = [a, b]`

`T = [a, b, c, d]`

`...`

Each double takes $\mathcal{O}(|T|)$ time and table needs to be doubled for every power of 2 that is $\leq n$, so inserting n items takes $\mathcal{O}\left(n + \sum_{i=1}^{\lfloor \log_2(n) \rfloor} 2^i\right)$ time

$$\mathcal{O}\left(n + \sum_{i=1}^{\lfloor \log_2(n) \rfloor} 2^i\right) = \mathcal{O}\left(n + 2^{\lfloor \log_2(n) \rfloor + 1}\right) = \mathcal{O}(n)$$

Accounting analysis

- charge \$3 for each insert:
  - \$ 1 for inserting the item
  - \$ 1 credit for later
  - \$ 1 credit to add to an item in the first half of the list
- when table is full, inserts into second half have filled up credit in first half, so each item has \$1 credit
- now we have enough credit to double the table, since copying takes \$1 per item

`delete` with table shrinking

- if $\alpha(T)$ gets too small, we can reduce memory waste by shrinking table

Naive approach: halve table when $\alpha(T) = \frac{1}{2}$

- when # of elements falls below $|T|/2$ (when $\alpha(T) \leq 1/2$), copy to new table with size $|T|/2$
- bad sequence of `insert` and `delete` can be expensive!
  - suppose table is full, then we `insert` , then we `delete` , then `insert` , then `delete` , `...`
  - each insert doubles table and each delete halves it
  - results in total cost being $\Omega(n^2)$

Better approach: halve table when $\alpha(T) = \frac{1}{4}$ - amortized analysis

- note that after size change, table is always half full
  - we only double table when it is full, so resulting table is half full
  - we only halve table when it is quarter full, so resulting table is half full

Charging scheme:

- charge \$2 for each delete
  - \$ 1 for deleting item
  - \$ 1 credit for future contraction
- table must be halved when there are $|T|/4$ items, which requires at least $|T|/4$ deletes since last size change, so there is always enough credit to halve the table when necessary