Continuing with semi-dynamic (only insertions) dictionary

Given a set of elements $S$ with $|S| = n$, we maintain sorted arrays according to the binary representation of $n$. (Almost log $n$ array)

**Search**: $O((\log n \cdot \log n) = O(\log^2 n)$

**Insertion**: Create a new array $A_j$ with $|A_j| = 2^d$, given that all the arrays $A_0, A_1, ..., A_{j-1}$ were full when the insertion occurred.

$A_j$ is created by combining all the elements from $A_0, ..., A_{j-1}$ ($\leq |A_j| = 2^d + 1 \leq 2^d \text{ elements}$)

Can be done in $O(2^d)$ comparisons
$2^d$ can be large (upto $n$)

How often do we pay the price for filling up $A_j$?

Observation: Only once after every $2^d$ insertions, $A_j$ is affected.

$\Rightarrow$ Over a sequence of $m$ insertions, the number of times, we incur a cost for $A_j = O\left(\frac{m}{2^d}\right)$

$\Rightarrow$ Total cost $= \sum_{j=0}^{\log m} O\left(\frac{m \cdot 2^d}{2^j}\right) = O(m \log m)$

So, the "amortised" cost of insertion in $O(\log m)$ (amortised over a "large"
Goal is to get a worst-case bound over a sequence of operations and look at the amortised cost.

\[ \text{Countercan count to } N \]
\[ b \log n \]
\[ 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \]

modulo \( N \) counts:

\[ \ldots \quad 0 \quad 1 \quad 0 \quad 1 \]
\[ \ldots \quad 0 \quad 1 \quad 1 \quad 0 \]

\[ \ldots \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \]

Worst case cost of a single operation say \( T \).

Then worst case cost of \( m \) operations in \( \Theta(T \cdot m) \).

\[ T = \log N \quad m = N \quad \Theta(N \log N) \]
Total # of bits shifted

\[ \sum \frac{N}{2^d} \cdot (j \cdot \frac{j}{2^i}) = N \cdot \sum \frac{j}{2^i} = O(N) \]

Amortized Analysis

Potential based amortized analysis

Associate a "potential function" \( \phi(i) \) with the state of the data structure. \( \phi(i) \) is the potential at state \( i \).

Amortised cost at step \( i \)

\[ \text{Amortised cost at step } i \]

\[ = \text{Change of potential + actual cost} \]

\[ = \phi(i+1) - \phi(i) + W_i \]

Total amortised cost : \( \sum \text{Amortised cost in step } i \)
\[ = \sum_{i} \phi(i+1) - \phi(i) + \sum_{i} w_i \]

\[ = \phi_f - \phi_{\text{initial}} + W \]

Total Amortised cost = \(\phi_f - \phi_i + W\)

\[ \Rightarrow W = \text{Total amortised cost} - (\phi_f - \phi_i) \]

\[ W \leq \text{Total amortised cost} \]

if \(\phi_f - \phi_i\) is non-negative

For the counter example:
what is a good potential function
# if \(h_i = 1\)

Amortised cost of incrementally the costs
\[ 10110111 \rightarrow 10111000 \]
Potential change = -3 + 1
Amortized cost = O(1)

Over a sequence of N increments, ToAmortized cost = O(N)