## → Bit Flips

Worst case → $O(\log_2 n)$ bit flips, and for $n$ numbers → $O(n \log_2 n)$ ✗

$f(i)$ = no. of times $i^{th}$ bit flips during $n$ increments.

$$T(n) = \sum_{i=0} f(i)$$

$f(0) = n$ $\quad f(1) = n/2$ $\quad f(2) = n/4$

$f(i) = n/2^i$ $\Rightarrow \Sigma = n + \frac{n}{2} + \frac{n}{4} + \dots\dots$

$\Rightarrow \Sigma < n\left(\frac{1}{1/2}\right) = 2n$

$\Rightarrow T(n) = O(n)$

## → Stack with Multi pop

pop of $i$ elements $\Rightarrow ic$

Worst case $= O(n^2)$ ✗

$T(n) = c(\#\text{push}) + c(\#\text{pop total})$

$\leq 2c(\#\text{push})$

$= 2cn$

## → Amortised Cost

- $\phi$ : potential func associated with the algo / data structure
- $\phi(i)$ = Potential at the end of $i^{th}$ oper$^n$
- Amortised cost of $i^{th}$ oper$^n$ = actual cost of $i^{th}$ oper$^n$ + $(\phi(i) - \phi(i-1))$

Actual cost of $i^{th}$ oper + $(\phi(i) - \phi(i-1))$

- Amortised cost of $n$ operations = $\sum$ Amortised cost of $i^{th}$ operation
  = Actual cost of $n^{th}$ oper$^n$ + $(\phi(n) - \phi(0))$

$\boxed{\phi(0) = 0}$ $\boxed{\phi(i) \geq 0 \ \forall i}$

Amortised cost of $n$ oper$^n$ $\geq$ Actual cost of $n$ oper$^n$

- Select a suitable $\phi$ so that for the costly operation, $\Delta(\phi)$ is negative to such an extent that it nullifies or reduces the effect of actual cost.

Find some quantity that is decreasing during the operation

→ $\phi$ : Bit Flips

$\phi(i)$ = # of 1's in the counter after $i^{th}$ increment.

| Actual cost | $\Delta(\phi)$ | Amortised cost |
|---|---|---|
| $k+1$ | $-k+1$ | 2 |

$\Rightarrow$ $i^{th} \Rightarrow 2$

$n \Rightarrow 2n = O(n)$

→ $\phi$ : Stack multi pop

| | Actual Cost | $\Delta(\phi)$ | Amortised cost |
|---|---|---|---|
| Push | $c$ | $c$ | $2c$ |
| Pop | $ic$ | $c(-i)$ | $0$ |

$\phi$ = c(# elements in stack after $i^{th}$ oper)

Amortised cost of each oper$^n$ $\leq 2c$
Amortised cost of n oper$^n$ $\leq 2cn$

→ $\phi$ : Binary heap (H)

| | Actual | $\Delta(\phi)$ | Amortised |
|---|---|---|---|
| ExtractMin | c Height(H) | −c Height(H) | 0 |
| Insert | c Height(H) | c Height(H) | 2c Height(H) |

$$\phi(i) = c \sum_{v \in H} depth(v)$$

⇒ Extract min ⇒ Amortised = O(1)
Insert → O(log n) still.

→ Dynamic Tables

n = no. of elements in table T.
createTable (K) → syscall that creates table of size K & returns its pointer

Size (T) size of table T

copy (T, T') copies contents of T into T'

free (T) : free the space occ. by T.

→ efficient way to insert: if table is full, create new table of size 2n. So the table is at least half full

always.

```
if (n==0) createTable (2);
else if (n == size(T){
    T' ← createTable (2n);
    copy (T, T'); free (T); T ← T';}
insert x into T; n++;
}
```

O(n)

| | Actual | $\Delta(\phi)$ | Amortised |
|---|---|---|---|
| Table not full | c | 2c | 3c |
| Table full | cn + c | c(2 − n) | 3c |

$$\phi(i) = c\left(n - (size(T) - n)\right) = (2n - size(T))$$

Amortised cost of each insert oper$^n$ = O(1)
⇒ Actual cost of n insert oper$^n$ = O(n)

→ **Delete**

```
n--;
if (n==0) free (T);
else if (n == size(T)/2) {
    T' ← createTable (n);
    copy (T, T'); free (T); T ← T';}
```

| | Actual | $\Delta(\phi)$ | Amortised |
|---|---|---|---|
| Table not half | c | c | 2c |
| Table half | cn + c | −cn + c | 2c |

$$\phi(i) = c(size(T) - n)$$

## → Both insert and delete

Insertion ⇒ double when full.
Delete ⇒ shrink when quarter full.
and make of size half
Size = 4n, ele = n ⇒ new table = 2n size

every time a table is created, it is half full. Now for any costly oper$^n$:
i) it becomes full ⇒ n insertions
ii) it becomes quarter full ⇒ $\frac{n}{2}$ deletions

| Delete(x) | Actual | Δ(ϕ) | Amortised |
|---|---|---|---|
| Table not shrink | c | 2c | 3c |
| Table shrink | cn+c | c(2−2n) | 3c − cn |

$\phi(i) = |c\,(2n - size\,(T))|$

3c − cn < 3c

Amortised cost < 3c ⇒ 3nc for n oper$^n$

| Delete(x) | Actual | Δ(ϕ) | Amortised |
|---|---|---|---|
| Table not shrink | c | | |
| Table shrink | cn+c | | |

$\phi(i) = c\,(4n - size\,(T))$

Amortised cost < ⇒ n oper$^n$ for

| Delete(x) | Actual | $\Delta(\phi)$ | Amortized |
|---|---|---|---|
| Table not shrink | $c$ | | |
| Table shrink | $cn+c$ | | |

$$\phi(i) = \begin{cases} c\,(2n - size(T)) & 2n \geq size(T) \\ c\,(size(T)/2 - n) & 2n < size(T) \end{cases}$$