Dynamic Graph Algorithms

CSL-531

2026

Introduction

Dr. Shahbaz Khan

Department of Computer Science and Engineering,

Indian Institute of Technology Roorkee

shahbaz.khan@cs.iitr.ac.in

Amortized Analysis

Methods:

1- **Aggregate**

Upper bounds total cost of n operations T(n),
so average cost <= T(n)/n

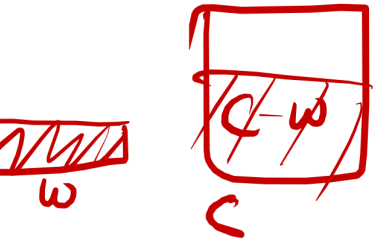
Textbook Reference

CLRS
Chapter 17

2- **Accounting**

Each operation is assigned an amortized cost to store extra credit to pay for the expensive operation later on.

Amortized Cost = Actual Cost + Extra Credit



3- **Potential**

Credit is computed in form of a potential which is used in heavy operations

Amortized Cost = Actual Cost + Change in Potential

$$W + \Delta\Phi \leq 2$$

Aggregate Method

WC time = Logn
Amortized =

BIT COUNTER

Consider the binary representation of the numbers. Incrementing by one unit, several bits may change in the binary representation.

Compute the AVERAGE NUMBER OF BITS FLIPPED per increment.

Upper bounds total cost of n operations T(n), so

Am. cost <= T(n)/n

T(n) = Number of bit flips after n increments

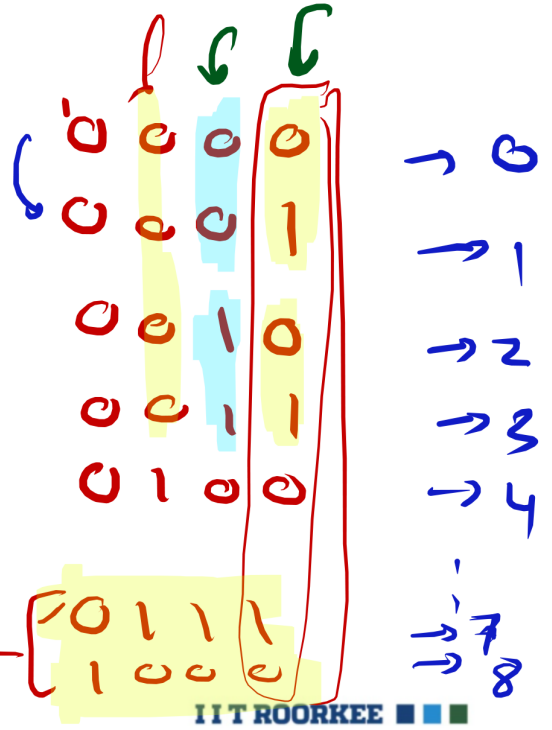
WC # flip = O(logn)

Start from 0 to n

1 ≤ Avg. flips ≤ O(logn)

2

logn



$\leftarrow 1+\lfloor \log n \rfloor \rightarrow$

$T(n) = \underbrace{\frac{n}{2^0}}_{\text{LSB}} + \frac{n}{2} + \frac{n}{4} + \cdots + \frac{n}{2^{\lfloor \log n \rfloor}}$

$\hookrightarrow \leq n$

$\#\text{bits} = 1+\lfloor \log n \rfloor$

$7 = 1 + 2$

$8 = 1 + 3$

$2^{1+\lfloor \log n \rfloor} = 2 \times 2^{\lfloor \log n \rfloor} \leq 2n$

$= n + \frac{n}{2} + \cdots 2 + 1$

$\leq 2n - 1$

Amortized flips per increment $= \frac{\text{Total}}{\#\text{increments}} = \frac{2n-1}{n}$

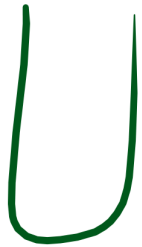
$= 2 - \frac{1}{n}$

Accounting Method

BIT COUNTER

Consider the binary representation of the numbers. Incrementing by one unit, several bits may change in the binary representation.

Compute the AVERAGE NUMBER OF BITS FLIPPED per increment.



Each operation is assigned an amortized cost to store extra credit to pay for the expensive operation later on.

Am. Cost ≤ Actual Cost + Extra Credit

Credit for increment:

Cost of flipping LSB

Credits acquired to flip remaining bits

$= (1 - \log n)$

$= 1 + \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \cdots + \frac{1}{2^{\lfloor \log n \rfloor}} \leq 2 - \frac{1}{n}$

(Flip LSB)

Potential Method

BIT COUNTER
Consider the binary representation of the numbers. Incrementing by one unit, several bits may change in the binary representation.

Compute the AVERAGE NUMBER OF BITS FLIPPED per increment.

Credit is computed in form of a potential which is used in heavy operations

Amortized Cost = Actual Cost + Change in Potential

Φ = Potential

≤ 2

$w + \Delta\Phi \leq$ Am. Cost

$\leq k$

	w	$\Delta\phi$	Am. Cost $(w+\Delta\phi)$
$0 \leftarrow 000000$	1	+1	2 $\leq$ (V)
$1 \leftarrow 000001$	2	0	2
$1 \leftarrow 000010$			
⋮			
$4 \leftarrow 011111$	5	-3	2
$1 \leftarrow 100000$	1	1	2
$2 \leftarrow 100001$			

000011

↓

000011110

100011101

ϕ = function of State of system

ϕ = #1's

$w + \Delta\phi = (k+1) + (1-k) = 2$

#Consecutive 1's in LSB

(W

Φ_0 → Very large

$w_1 + \Phi_1 - \Phi_0 \leq k$

$w_2 + \Phi_2 - \Phi_1 \leq k$

$\vdots$

$w_n + \Phi_n - \Phi_{n-1} \leq k$

$(w_1 + w_2 + \cdots + w_n) + \Phi_n - \Phi_0 \leq nk$

$k \left(\frac{w_1 + w_2 + \cdots + w_n}{n} \right) \leq k - \frac{(\Phi_n - \Phi_0)}{n}$

$\leq k - \frac{\Phi_n}{n} + \frac{\Phi_0}{n}$

Properties

① $\Phi_i \geq 0$

② Φ_0/n negligible compared to k

Robert
Gedre
Tarjan

- DFS DSU??
- Bridge/Articulation
- SCC's
- DS$_2$
- Amortized Analysis

Amortized Analysis

Stack = Push → WC O(1)
Pop → O(1)

2- Multi Pop Stack: **Supports following operations:**

a) Push (x) : Push x to top of Stack

b) Pop (k) : Pop k elements from Stack (if exists)

WC
O(1)
O(k)

Amortized time

Accounting method

Push = (1 + 1) ≥ 2
Push, Future Pop

0

Am Cost ≤ 2

Aggregate

T = #Cost Push + #Cost Pops

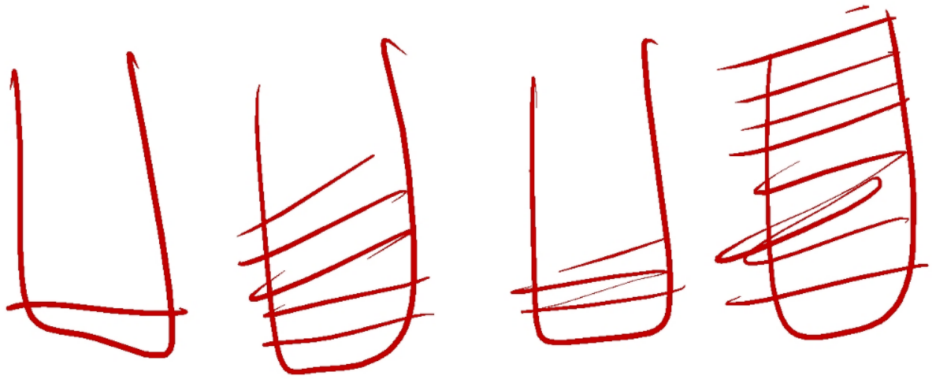
#Cost Pops ≤ #Cost Push

≤ 2 #Cost Push

≤ 2n

$\frac{T}{n} \leq 2$

Potential



Φ = # Elements in the stack

	$W \neq \Delta\Phi$	Am	
Push	1 + 1	= 2	≤ 2
Pop (k)	k − k	→ 0	

n push
pop

$$\frac{2n+2-2}{n+1} \leq 2 - \frac{2}{n+1}$$

Homework

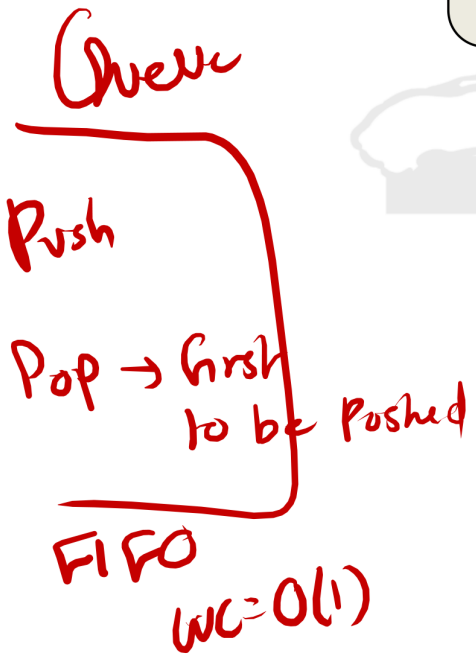
Compute the
AVERAGE NUMBER OF LITERALS CHANGED
per increment for

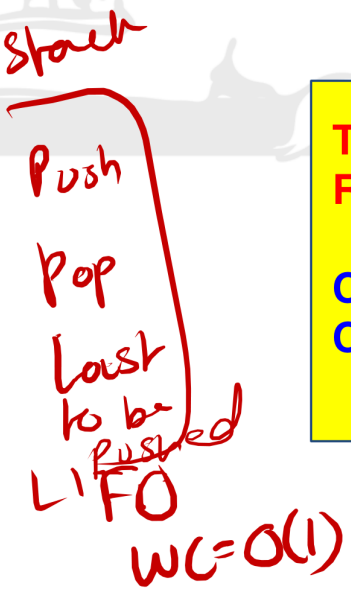
1- Digit Counter (0,...,9)
2- B-ary Counter (0,...,b-1)

→ HW ←

QUEUE USING TWO STACKS
Formally Analyze the implementation of a
Queue using two Stacks

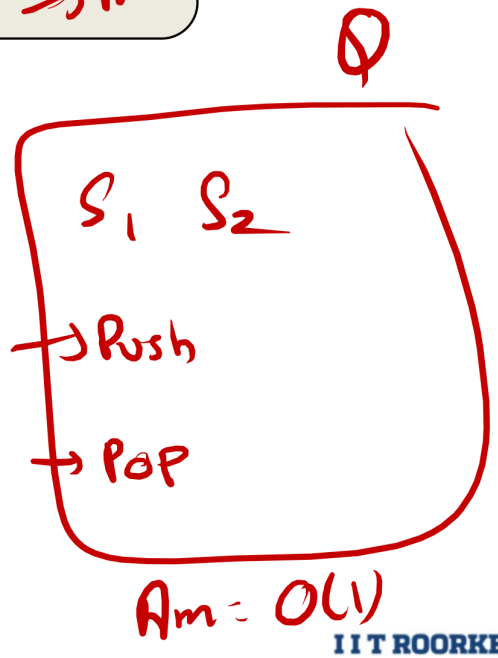
→ HW ←





Textbook Reference

CLRS
Chapter 17



Recap Questions

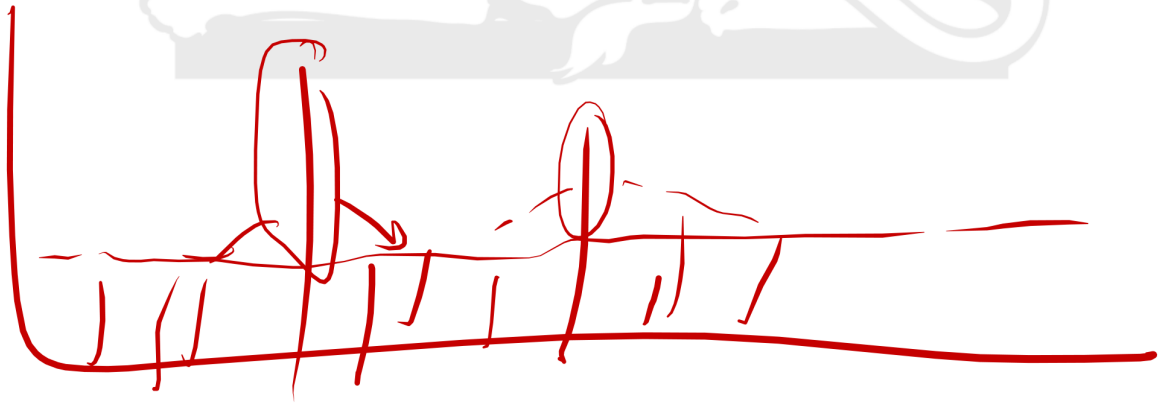
AMORTIZED ANALYSIS IS USED WHEN

a Different Operations take different times

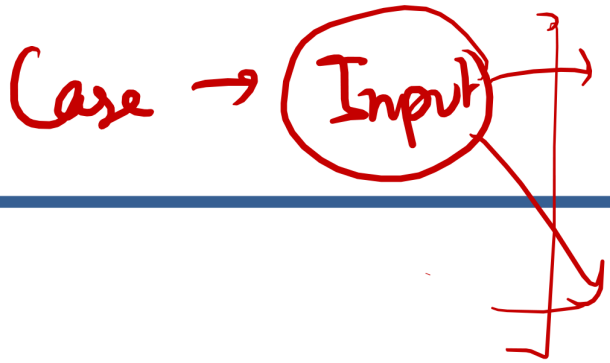
b Same operations take different time

c Expensive operations are rare

d All of the above



Recap Questions



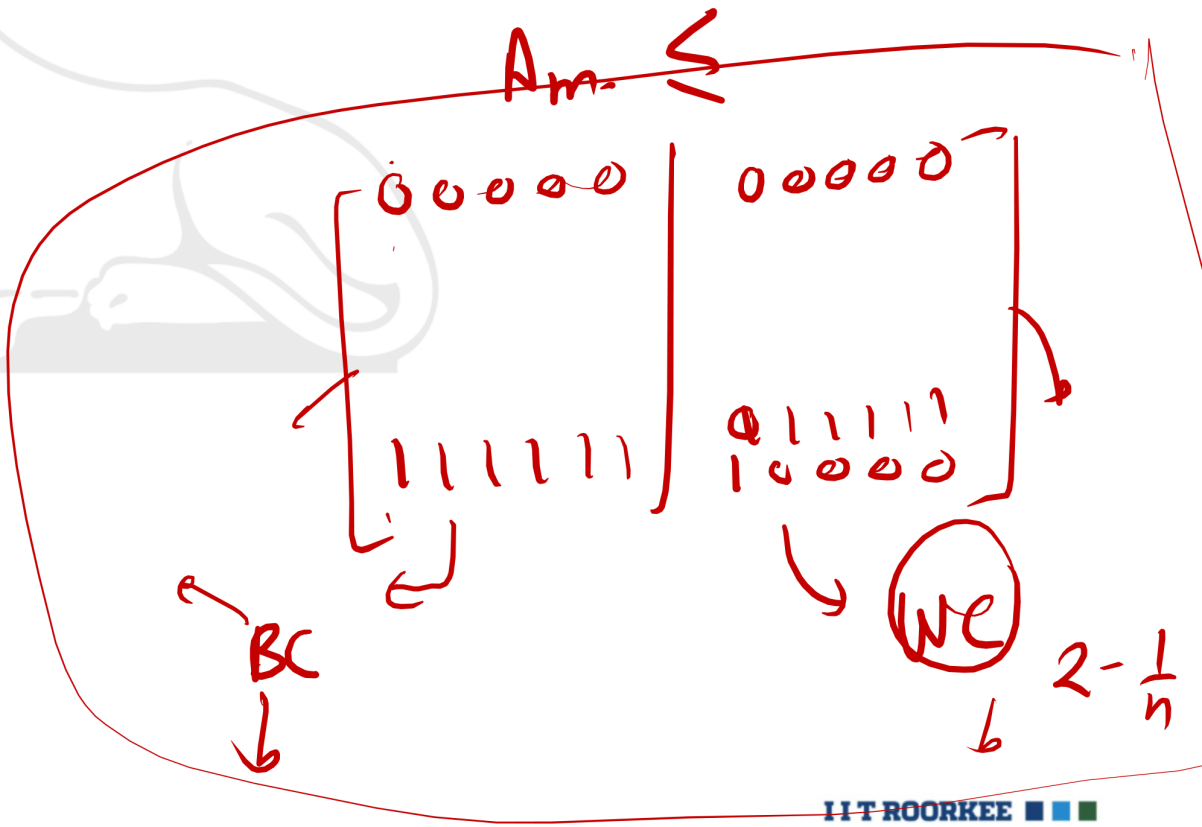
Amortized Analysis can Compute

a Best Case Time Complexity

b Average Case Time Complexity

c Worst Case Time Complexity

d All of the above



Worst case
Best case
Upper Bound
Lower Bound

Classical Problem

Incremental Reachability
Given a graph under edge insertions, maintain all vertices reachable from s

What is optimal update and query time?

Static

$O(m+n)$

Dynamic

$O(1)$ time

↳ Ω

Amortized?

w.c. time?

s-t Reachability (Directed)

↳ ∃ path from s to t

Connectivity (Undirected)

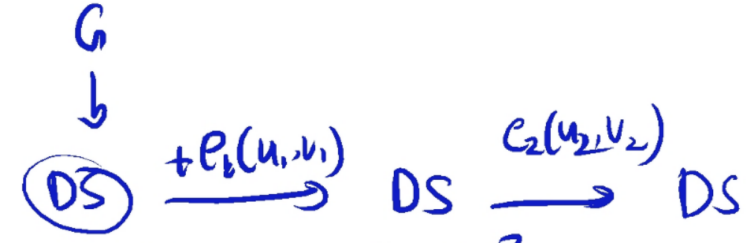
Single Source Reachability (S)

↳ Graph $G(V,E)$ having n vertices and m edges
Directed
with source $s \in V$

→ Static: DFS / BFS (Reachability tree)

↳ $O(m+n)$

Incremental:



Decremental:

$$R[v] = \begin{cases} 1 & \text{if } s \rightsquigarrow v \\ 0 & \text{if } s \not\rightsquigarrow v \end{cases}$$

Preprocessing

Initialize $R[v] = 0$ for all

Start DFS/BFS from s $\Rightarrow$ Tree T

$\forall v \in T \Rightarrow R[v] = 1$

Update:

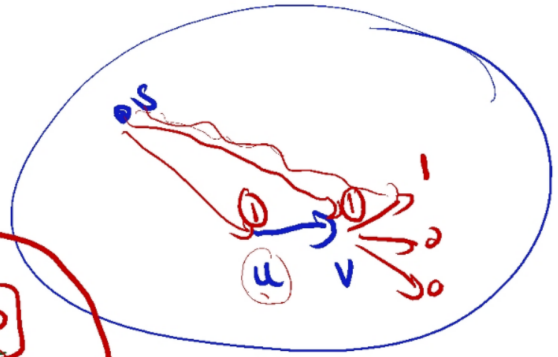
(u, v)

If $R[u] = 1$ && $R[v] = 0$

→ $R[v] = 1$;

→ for all $(v, w) \in E$

Update (v, w);



R[u]	R[v]	
0	0	X
0	1	X
1	0	
1	1	X

Space $G + R$ → $O(n)$

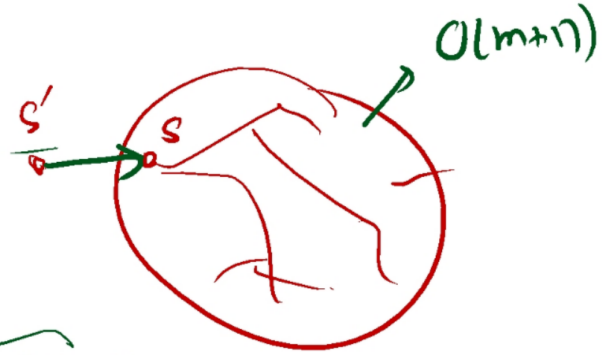
Time: $WC \Rightarrow O(m+1)$

$Am = O(1)$

Query (v)

return $R[v]$ ← $O(1)$

WC example



If a vertex u

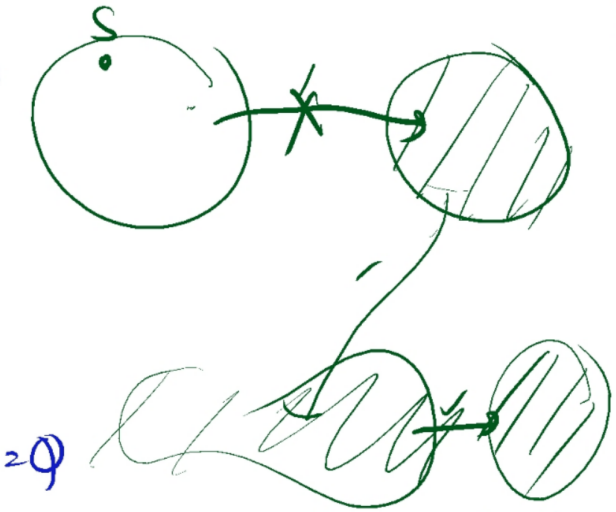
$R[u] = 0 \rightarrow R[u] = 1$

$O(\deg(u))$

Total $\sum_{u \in R[\cdot]=0 \rightarrow R[\cdot]=1} 1 + \deg(u)$

WC $\sum_{u \in V} 1 + \deg(u) = O(m+n)$

Total time



e_1 (u,v) V_1 $\sum_{v \in V_1} \deg(v)$

e_2 V_2 " V_2

e_3 V_3 " V_3

$V_1 \cap V_2 = \phi$, $V_2 \cap V_3 = \phi$

Total $\sum_{\substack{v \in V \\ \text{if } R[v]=1}} 1 + \deg(v) = O(m+n)$ | Amortized time $= \frac{O(m+n)}{m} = O(1)$

Aggregate = $\frac{\text{Total}}{\#\text{ Updates}} = \frac{O(m+n)}{n} = O(1)$

Accounting:

R[u]	R[v]	w	Δφ	(u,v) w	+ Credit
0	0	1	+1	0	+ O(1)
0	1	1	+1	0	+ O(1)
1	1	1	0	0	+ O(1)
1	0		$-\sum \deg(v)$ $R[v]=0$	1	+ 0
			$\downarrow$		$= O(1)$
			$R[v]=1$		

$O(1) + \sum \deg(v)$ $R[v]=0$ $\rightarrow$ $R[v]=1$

Processing (u,v) $\begin{bmatrix} R[u]=1, R[v]=0 \\ R[v]=1 \end{bmatrix}$

Potential $\Phi = \sum_{v \in R[v]=0} \deg(v)$