Unit - 4 - Amortized Analysis

→ If the are ~~some~~ series of operations where the cost of a single operation is very large and rest of operation cost less, worst case running asymtotic time complexity may not give us a tighter bound.

→ It is used for algorithm have a operation is very slow and but most of operation are faster. And we analysis algorithm and guarantee a worst case average time is lower than the ~~worst~~ worst case time complexity.

eg Hash Table, Disjoint Set.

→ The amortized analysis does not say anything about the cost of individual operation. rather it average the cost of all the operation in the worst case.

1) Aggregate Method
→ Simple Method.
→ Amortized cost per operation: Total cost of all operation / No. of operation

T(n)/n
→ count the complexity of each operation.

Stack operation (Augmented Stack).

Asymtotic Analysis

POP(S) = O(1)
Push(S,x) = O(1)
MultiPOP(k) = O(n) min(n, k)

for n operation.
complexity will be O(n) × n = O(n²)

Aggregate method

Total cost of pop, or multipop ≤ no of pushes which is at most n.

So, let have n operation of push and pop
Amortized Analysis = O(n)

for single operation and O(n)/n = O(1)

Binary counter

let we have array B[K] - K bit
So, also MSB = B[K-1]
LSB = B[0]

It Stores: no. = $\sum_{i=0}^{k-1} B[i] \cdot 2^i$ [eg, B[0]×2⁰ + B[1]×2¹ + B[2]×2².

2022.12.30

cost = no of flip

Counter value					cost	cumulative cost
0	0	0	0	0	0	0
1	0	0	0	1	1	1
2	0	0	1	0	2	3
3	0	0	1	1	1	4
4	0	1	0	0	3	7
5	0	1	0	1	1	8
6	0	1	1	0	2	10
7	0	1	1	1	1	11
8	1	0	0	0	4	15
9	1	0	0	1	1	16
10	1	0	1	0	2	18
11	1	0	1	1	1	19
12	1	1	0	0	3	22
13	1	1	0	1	1	23
14	1	1	1	0	2	25
15	1	1	1	1	1	26

```
i=0
while (i < length and B[i] == 1)
    B[i] = 0
    i = i+1
if (i < length)
    B[i] = 1
```

total no. of flip: $n + \frac{n}{2} + \frac{n}{4} + \frac{n}{8} \dots + \frac{n}{2^{k-1}}$

$\leq 2n = O(n)$.

$= n\left(1 + \frac{1}{2} + \frac{1}{4} + \dots \frac{1}{2^{\log n}}\right)$

$= n\left(\frac{1 - (1/2)^n}{1/2}\right) = 2$

So, for each operation $\frac{O(n)}{n} = O(1)$

Accounting method

→ Assign diff. charge to diff operation called amortized cost, which may less or more than its actual cost

→ If the amortized cost is more than its actual cost the diff. is called credit

Total credit = $\sum_{i=1}^{n} \hat{c}_i - \sum_{i=1}^{n} c_i$ (always non negative)

$\hat{c}$ = amortized cost

c_i = actual cost.

Stack operation

let assign amortized cost

push = 2

Pop = 0

Multipop = 0

On every push we will get 1 as credit and for pop and multipop cost is paid from credit.

So, $O(n)/n = O(1)$

Binary counter

let, for setting cost $(0 \to 1) = 2$ unit
resetting cost $(1 \to 0) = 0$ unit

Here also
I added 2 in credit and paid when reseting

Potential method

Similar to accounting method

But in this uses potential or energy to pay costly operation

- At any given point, work is potential in data structure
→ Initial potential of DS $= D_0$
→ D_i will get after i^{th} operation on D_{i-1}
→ ϕ Potential function $\Phi : \{D_i\} \to R$

$\Phi(D_0) = 0$ and $\phi(D_i) \geq 0$ for all i

* Amortized cost = Actual cost + Change in Potential

$$\hat{C}_i = C_i + \Phi(D_i) - \Phi(D_{i-1})$$
$$= C_i + \Delta\Phi_i$$

if $\Delta\Phi_i > 0$ overcharged
$\Delta\Phi_i < 0$ undercharged.

Augmented Stack. let $\phi(D_i)$ = no of element in stack

Push

Amortized cost = Actual + potential diff

$= 1 + [(n+1) - (n)]$
$= 1 + 1 = 2$

pop

$= 1 + [(n-1) - n]$
$= 0$

multipop $= k + [(n-k) - n]$
$= 0$.

Ans,
$O(n)$ for n operation & $O(1)$ for one operation.

Binary counter.

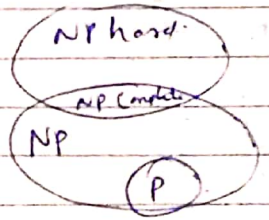
$\Phi(D_i)$ = no. of 1's in counter after i^{th} operation.
$\phi(D_i) \geq 0$

		Actual cost	$\Phi(D_i)$	$\Phi(D_{i-1})$	$\Delta\Phi$	Amortized
0	0 0 0 0	0	0	0	0	0
1	0 0 0 1	1	1	0	1	2
2	0 0 1 0	2	1	1	0	2
3	0 0 1 1	1	2	1	1	2
4	0 1 0 0	3	1	2	-1	2
5	0 1 0 1	1	2	1	1	2

we are get 2 for every operation, So,
$2+2+2 \ldots +2 = O(2n) \cdot O(n)$ for n operation

Scanned with CamScanner

* Tractable problem	Non tractable problem
→ Can be solved in polynomial time	→ Can't solve in polynomial time
→ Can be verified in polynomial time	→ Can verify in polynomial time
→ easy to solve	→ not easy to solve
→ Bubble sort $O(n^2)$	→ TSP, towers of hanoi, Eg $O(2^n)$
Class - P	Class NP
	→ Solved by guessing and verification



Randomized Algorithm

- An algorithm that uses a random number at least once during the computation to make a decision
→ follow probabilistic Analysis
→ Uses Random no. generator

eg
① Choosing pivot in Quicksort
② Pick edge from graph randomly

Deterministic Algo	Random algo.
I/P → A1go → O/P	I/P → Alg → O/P, Random no.
O/P and time is same for same IP	O/P and exe. time varies for same input.

Types
(a) - las vegas.
→ Run within a specified amount of time
→ If solution is found within time, the solution will be exactly correct
→ If Algorithm run out of time it does not find any solution.

(b) Monte Carlo (probabilistic)
→ Depending on the I/P, there is slight probability of
(a) producing an incorrect result or
(b) failing to produce a result altogether

Adv. → Simple, fast, require less Resource

disadv. → not reach global optimum solution
→ Some time. Suboptimal is better than optimal solution

Approximation Algorithm.

→ Heuristic Algorithm

→ finding an optimal solution for NP-Complete.

Goal

→ To design an algorithm and come as close as possible to the optimum value in a reasonable amount of time

→ It find feasible solutions.

eg. TSP, optimization problem is find shortest cycle and approximation problem find short cycle.

Characteristic

(1) It gaurantee to run in polynomial time. But don't gaurantee for effective solution

(2) It used to get the answer near to optimal.

(3) With accuracy and top quality

Performance Ratio

Let C be the cost of solution by approximation and C* is cost of optimal solution

$$\max\left(\frac{C}{C^*}, \frac{C^*}{C}\right) \leq P(n)$$

$P(n) \geq 1$

High value of P(n) indicate bad solution.

Vertex cover problem

C ← {}

E' ← E

while |E'| ≠ 0 do

Select an edge (u, v) ∈ E'

Add u and v to C

E' ← E' − (Adjacent to u or v)

end

return C

Embedded System.

→ Capable of doing computation and handle the inputs.

→ But made for specific functionality

eg washing machine, AC, Bluetooth, etc

Characteristic

① Single function

② Tightly constraints

③ time bounded.

④ Microprocessor based.

* Algorithm implimented on the embedded system.

They have tight constraints

Problem for embedded Algorithm

① Limited Memory ② deadline based ③ low power consumption

Embedded System Scheduling

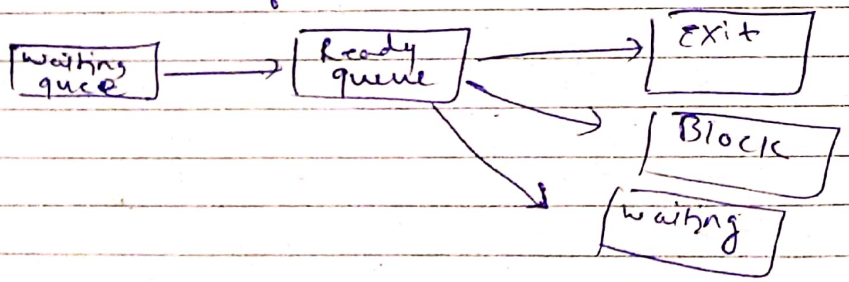
Scheduler - The ~~of~~ Software that decide which operation should be executed next.

The algorithm describe the logic and the mechanism of the Scheduler is called the "Scheduling Algorithm"

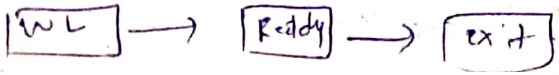
& each task has priority, based on that.

2 types

(1) fixed priority Algorithm.
priority assign at design time and not change during lifetime.



(2) Dynamic priority algorithm
→ priority change dynamically
→ check deadline of task in the highest priority job and schedule with the nearest deadline



Sorting Algorithm for Embedded System

It must have. (1) Sort in place — No dynamic Allocation
(2) Iterative
(3) Invariable running time
(4) I/p size dependence
(5) Reasonable code size
(6) clean implementation.

Insertion sort is best for this
→ Stable
→ In place

Algorithm

```
for j=2 to n-1
    key = A[j]
    i = j-1
    while i>0 and A[i] > key
        A[i+1] = A[i]
        i = i-1
    A[i+1] = key
```

A[i+1] = key.