

Amortized analysis

Tuesday, December 13, 2022

4:51 PM

Amortized Analysis -

- If there are some operations happening over a data structure and one operation has very high cost or time complexity than other operations then worst case time complexity using asymptotic analysis will not give accurate solution.
- For the cases where some operation costs very high then Amortized analysis is used to get accurate results.
- Asymptotic analysis multiplies number of operation with the worst case complexity and gives this as final solution
- For example a store has 501 items in which 500 items has cost Rs.1 and 1 item having cost Rs.500
- We wanted to find how much money we should take with us to buy all the items.
- Using asymptotic analysis - Highest cost/worst case = 500 and number of items = 501
Hence solution is $500 \times 501 = 250,500$
- Using amortized analysis - 1 item with cost 500 and 500 items with cost 1
Hence solution is $1 \times 500 + 500 \times 1 = 1,000$
- Hence Amortized analysis has given better solution than asymptotic analysis

Analysis of Augmented Stack -

Augmented stack

→ push() → $O(1)$
→ pop() → $O(1)$
→ multipop(k)
 k is no. of elements
Now we have to perform
multipop(N)
multipop(N)
→ Before this we have to push N elements again back to stack

Time complex = $O(1) \times N \times O(1) \times N$
 $= O(N^2)$

Using Amortized analysis

Time complexity
 $= O(1) \times N + O(1) \times N + O(1) \times N$
multipop(N) push(N elements) multipop(N)
 $= O(N) + O(N) + O(N)$
 $= 3O(N)$
 $= O(N) \rightarrow$ For N operation
For single op. = $\frac{O(N)}{N} = O(1)$

Aggregate method -

Amortized cost per op. = $\frac{\text{Total cost for all ops}}{\text{No. of operations}}$

consider a dynamic table

Insert $\begin{array}{|c|} \hline 1 \\ \hline \end{array} \rightarrow \begin{array}{|c|} \hline 1 \\ \hline \end{array} \rightarrow \begin{array}{|c|} \hline 1 \\ \hline \end{array}$
 Insert $\xrightarrow{\text{overflow}} \begin{array}{|c|} \hline 2 \\ \hline \end{array} \rightarrow \begin{array}{|c|} \hline 2 \\ \hline \end{array}$
 Insert $\xrightarrow{\text{overflow}} \begin{array}{|c|} \hline 3 \\ \hline \end{array}$
 Insert $\rightarrow \begin{array}{|c|} \hline 1 \\ 2 \\ 3 \\ \hline \end{array}$
 Insert $\rightarrow \begin{array}{|c|} \hline 1 \\ 2 \\ 3 \\ 4 \\ \hline \end{array}$

Asymptotic cost = cost req to copy prev. elements \times no. of operations
 $= O(n) \times n$
 $= O(n^2)$

Amortized cost = $\frac{O(n)}{\text{copy prev}} \times \frac{O(1)}{\text{adding new element}}$
 $= O(n)$

Amortized cost per operation = $\frac{O(n)}{n} = O(1)$

Accounting method -

- Assign a random cost as ammotized cost to the 1st operation
- If actual cost is less than ammotized cost then use the remaining cost for next operaitons

Accounting method

actual cost is 1

let, ammotized cost for insertion be $C_i^A = 3$

$\sum C_i \leq \sum C_i^A$
 $[1 \leq 3]$
 $= 2$ can be used for next op.

Upto 4 each req 0 cost but from 5 we have to copy prev 4 & then insert. For this we can use remaining cost 3

For this cost upto 8 is 8 from prev. & this can be used to add next elements

$C_i^A = 16$

$C_i = 3$
 $C_i = 8$

Potential method -

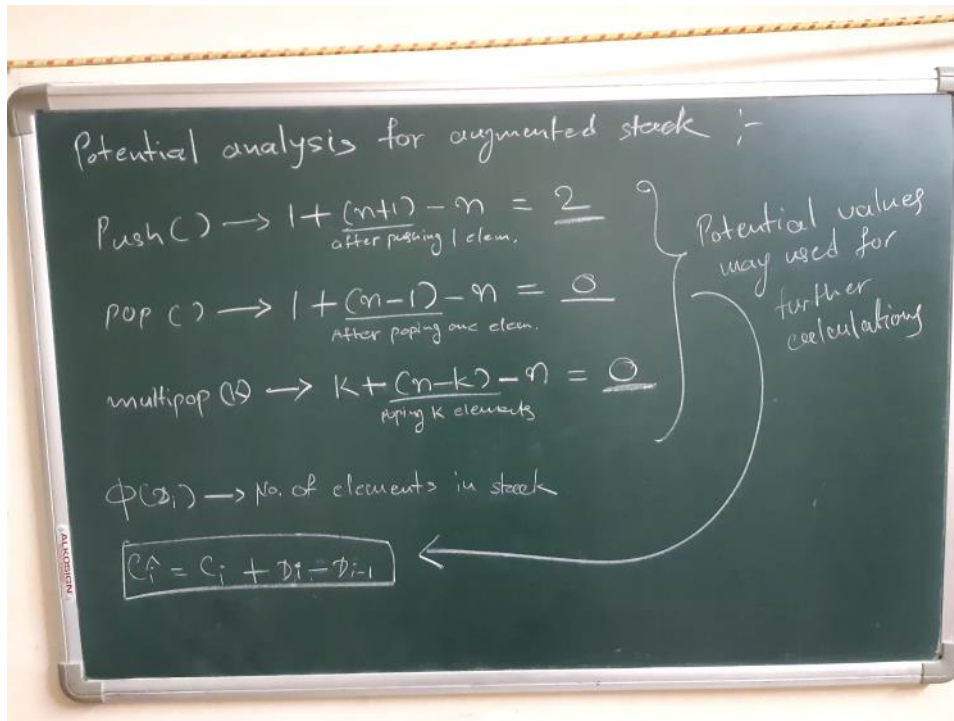
- At any given time there is potential in a data structure
- Initial potential = D_0
- Operation 'i' will change potential from $D(i-1)$ to D_i
- Actual cost = C_i and Ammotized cost = C_i^A
- Potential function are pre-defined function calculated after analysis

- Let it be $\phi(D_0)$
- Potential method says that,

Amortized cost = Actual cost + Change in potential

$$C_i^A = C_i + (D(i) - D(i-1))$$

$$C_i^A = C_i + \Delta(D_i)$$



Tractable problems -

A problem that is solvable by a polynomial-time algorithm.

The upper bound is polynomial.

Here are examples of tractable problems (ones with known polynomial-time algorithms):

- Searching an unordered list
- Searching an ordered list
- Sorting a list
- Multiplication of integers (even though there's a gap)
- Finding a minimum spanning tree in a graph (even though there's a gap)

Non tractable problems -

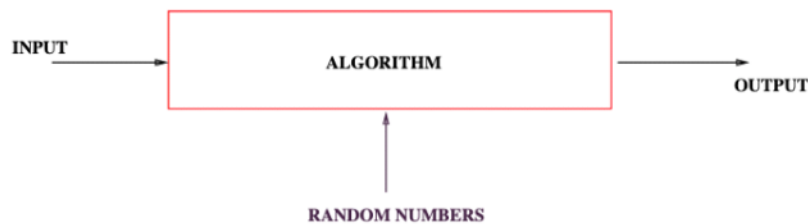
- A problem that cannot be solved by a polynomial-time algorithm.
- The lower bound is exponential.
- From a computational complexity stance, intractable problems are problems for which there exist no efficient algorithms to solve them.
- Most intractable problems have an algorithm that provides a solution, and that algorithm is the brute-force search.
- This algorithm, however, does not provide an efficient solution and is, therefore, not feasible for computation with anything more than the smallest input.
- Towers of Hanoi: we can prove that any algorithm that solves this problem must have a worst-

case running time that is at least $2^n - 1$.

Randomized algorithm -

- An algorithm that uses random number to decide what to do next anywhere in its logic is known as Randomized algorithm
- Used to reduce space and time complexity
- Output may vary even the input is same
- Hence it is a non-deterministic algorithm
- It is also called as probabilistic algorithm

Randomized Algorithms

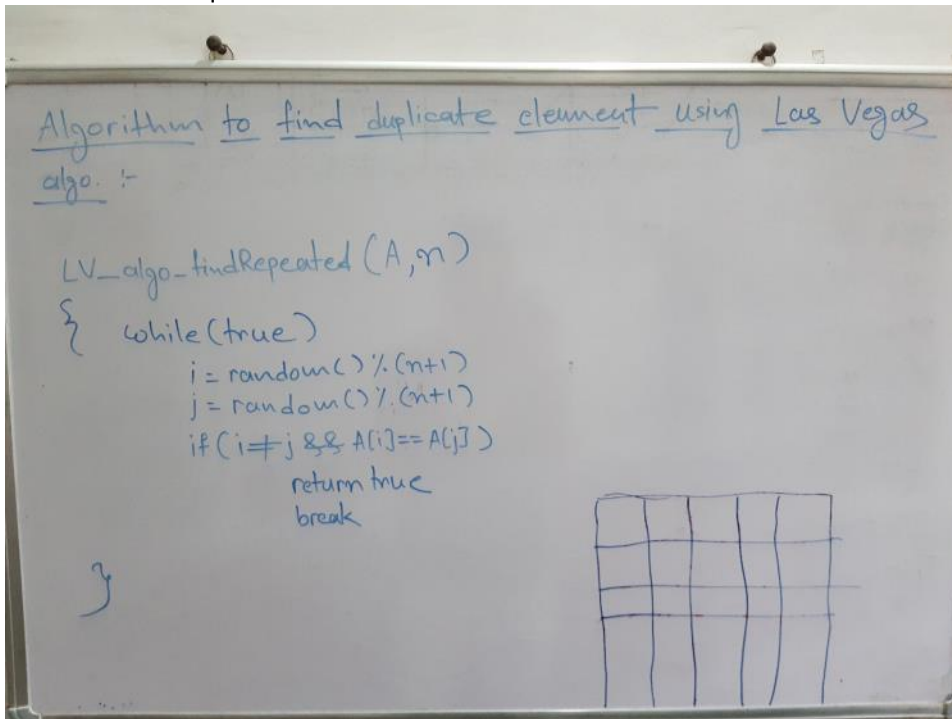


Las Vegas Algorithm -

Output is always correct.

Always gives same output for same input.

Ex. Randomized quick sort

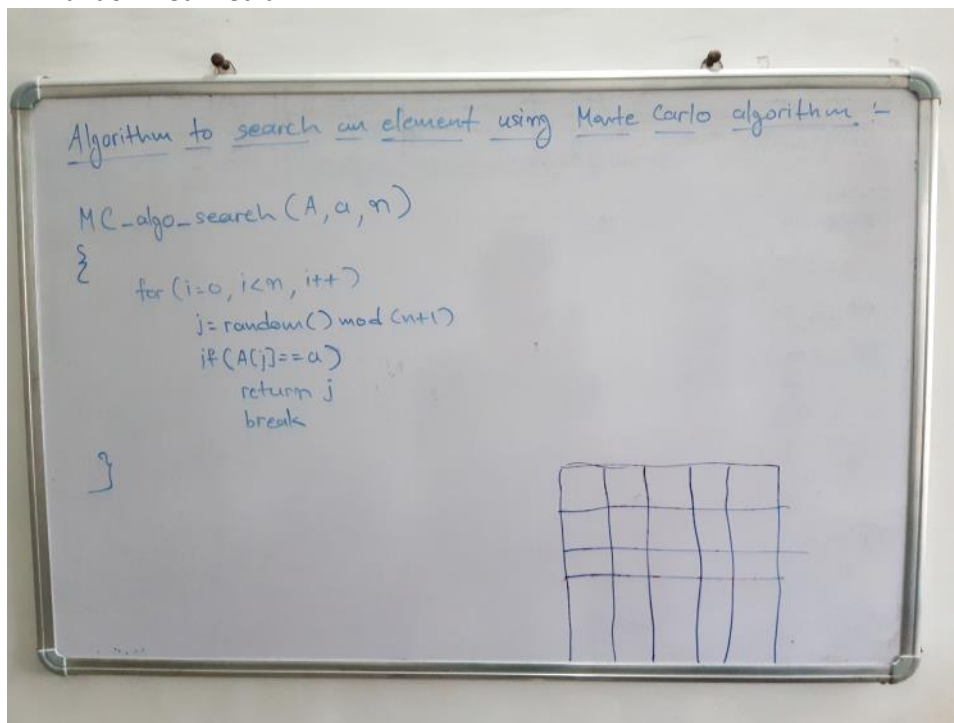


Monte Carlo algorithm -

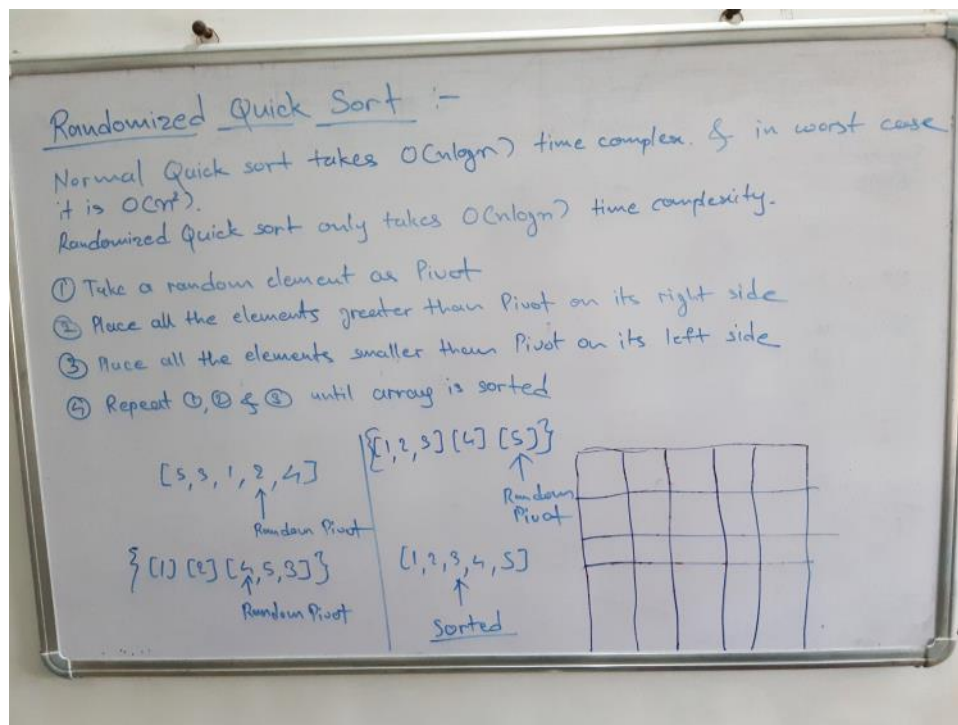
Output may be incorrect.

May generate different output for same input.

Ex. Randomized median



Randomized Quick Sort -



Approximation algorithm -

- It is the way of dealing with the NP Complete problems for optimization.
- The goal is to generate solution close to the optimal solution.
- It does the same in polynomial time.
- C = Cost of solution
 C^* = Cost of optimal solution
 $\text{Rho}(n)$ = Approximation ratio

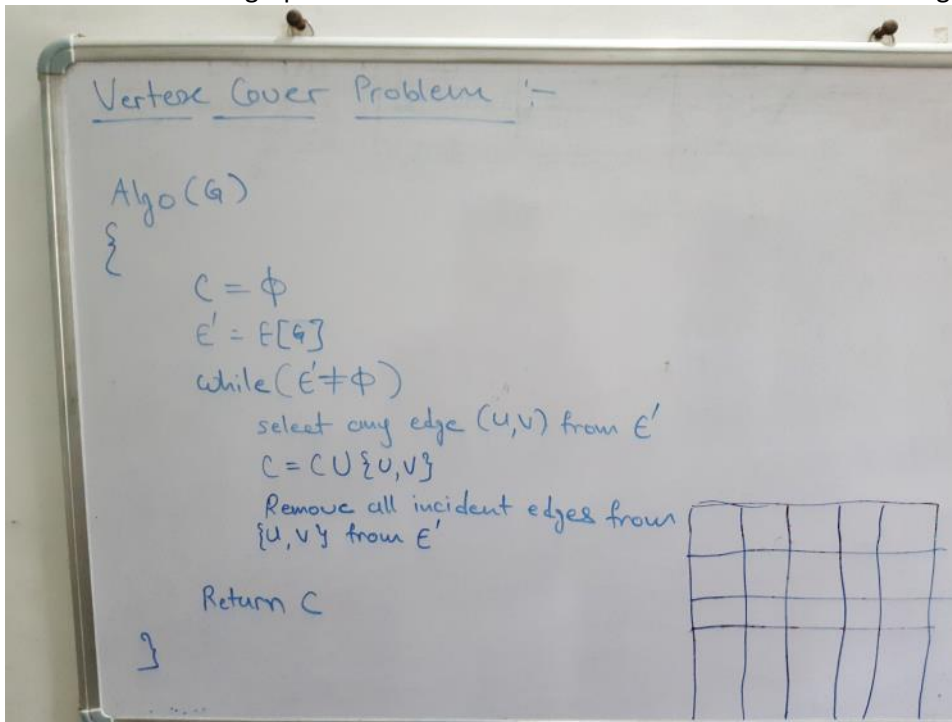
Maximization - $C^*/C \leq \text{Rho}(n)$

Minimization - $C/C^* \leq \text{Rho}(n)$

$\text{Rho}(n) > 1$

Vertex cover problem -

A vertex cover for a graph is the set of minimum vertices that covers all the edges in the graph.



Vertex Cover Problem

① Take edge (b,c)
 ② $U = \{b, c\}$
 ③ Remove incident edges on b or c
 ④ Take edge (e,f)
 ⑤ $U = \{b, c, e, f\}$
 ⑥ Remove incident edges on e or f

⑦ Take edge (d,g)
 ⑧ $U = \{b, c, e, f, d, g\}$
 $\therefore C = 6 \rightarrow$ No. of elements in U
 But we can get soln. by only selecting (b, e, d)
 $\therefore C^* = 3$
 \therefore As for Min. problem $= \frac{C}{C^*} \leq \sigma(n)$
 $\therefore \sigma(n) > 1$
Valid
 $= \frac{6}{3} \leq \sigma(n)$
 $= 2 \leq \sigma(n)$

denoted in graph

Embedded Systems -

- The computer hardware having software embedded in it is known as Embedded system
- It is a microcontroller or microprocessor based systems
- They are made for specific functionalities
- They are cheapest
- They have low power consumption
- Only one functionality
- They have very strict constraints to work
- Work in real time environment to generate results

Embedded algorithms -

The algos implemented on a microcontroller or microprocessor based systems are known as Embedded algos

They are similar to normal algos but they have to satisfy the stricter constraints of embedded systems.

Embedded Scheduling Algo -

- Scheduler is the software that decides which operation to do next
- Scheduling algo is the logic or implementation of the Scheduler
- For embedded system a scheduling algo should be microprocessor based and deadline based
- It has to operate in real time to generate the result
- Factors on which scheduling is done
 1. Interdependencies of tasks
 2. Resources
 3. CPU utilization
 4. Deadlines
 5. Cost of the operation

Embedded sorting algo -

It is similar to any normal sorting algo just it has to consider following factors,

- 1.sort inplace
- 2.Iterative
- 3.minimum code size
- 4.feasible time complexity

Ex. Insertion sort

Time complexity - $O(n)$ for best case, $O(n^2)$ for worst case