U-5
Amortized Analysis.

DOMS | Page No.
Date / /

- measure performance of algo in avg. case.
- measures time/resource complexity in context of computer program.
- some method/resource → cheaper
- says about avg cost of all operations in worst case.

Methods of amortized analysis.

1) Aggregate Method

- assign n → same cost all oper^n
- find upper bound cost $T(n)$ of oper^n & calculate amortized cost as avg of that $T(n)/n$.

↓ stack

- Total no. of pops/multiple pops ≤ no. of push (total)
- max no. of push → n
- time for entire sequence → O(n)
  amortized cost = $\frac{O(n)}{n} = O(1)$

Binary ↓

- n + n/2 + n/4 —— + n/2k + r = 2n (bit flips)
- Total cost of sequence = O(n)
- Amortized cost = $\frac{O(n)}{n} = O(1)$

2) Accounting Method

- diff cost for diff operation
- less or greater actual cost
- amortized cost > actual cost
  ↳ diff of two → credit
- credit → pay debt for oper^n

Stack

- Push - 2
- Pop - 0
- Multipop - 0
- push → cost 1
  1 as credit.
- enough credit → each oper^n
- amortized cost → O(1)
- cost of entire sequence → O(n)

Binary

- all changes form 1 to 0 are pair with already stored credit.

Amortized.

- cost per op = O(1)

3) Potential method.

- similar to accounting. not credit → potential energy
- special ds → store potential of object.
- potential diff > 0 → overcharge
  PD < 0 → undercharge.

Stack ↓

- push = 2
  pop = 0.
  multipop = 0.

Binary ↓

- Bi is no. of 1 after ith increment.
- $B_i \le B_{i-1} - t_i + 1$.
- diff in potential
  $= B_i - B_{i-1} \le B_{i-1} + t_i + 1 - B_{i-1}$
  $= t_i - 1$
- Amortized cost
  $c_i + B_i - B_{i-1} \le t_i + 1 - t_i$
  $= 2$.

DOMS | Page No.
Date / /

Time Space Trade Off -

- situation where either a space efficiency (memory utilization) can be acheived at cost of time or time efficiency / performance efficiency can be acheived at cost of memory.

Acheiving time efficiency in computation.

- Input enhancement
  - Counting methods for sorting
  - string matching algorithm
- Prestructuring
  - Hashing
  - B-trees
- Dynamic Programming

Issues → 1) time & space complete → efficient algo → space efficient
2) Data compression techniques -

DOMS | Page No.
Date / /

Tractable

- solved in polynomial time.
- verified in " "
- easy to solve
- principle of fucn known
- Independence to other systems
- eg → sorting list
  multiplication of integers
  search unordered list.
  search ordered list

Non-Tractable

- solved in exponential time.
- verified in polynomial time
- not easily to solve.
- principles of fucn are partly known.
- Interdependence.
- eg → Traveling Salesman Problem
  → Knapsack problem.

Randomized Algos-

- NP may be → randomized or approximity algo in polynomial time.
- random number generator or randomizer
- decision → based current output.
- goal → quickly give a soln nearest to optimal soln

Categories → Las Vegas Algo • (produce same outcome on each execution
• (Always right answers) for same input)

→ Monte Carlo Algo • (produce diff outcomes for same input on each execution)-
• (may produce wrong answers)

Adv → simple to implement
→ many times efficient than trad^n algo

Disadv → small degree of error (dangerous)
→ not possible to obtain better results.

Approximate Algo -

- Quick soln → no guarantee of optimal soln
- apply heuristics to cut computation
- feasible soln → near to optimal soln → called approximate soln

characteristics

I → Instance

P → problem.

FO(I) & FA(I) → value of optimal & feasible soln

based on their diff → categorize approximation.

1) $|FO(I) - FA(I)| \leq K$ [absolute Approx.]
2) $|FO(I) - FA(I)| \quad FO(I) \leq f(n)$ [f(n)-Approx]
3) $|FO(I) - FA(I)| / FO(I) \leq \epsilon$ [ϵ approx]

Performance ratio

C^* → cost of optimal soln

C → cost of soln returned.

ϵ → approx algo

Ratio P(n) for input size n →

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

DOMS | Page No.
Date / /

Embedded Algorithms.

↳ computer system with dedicated fun" with larger mechanical or electrical system.

Characteristics → • single fuctions.
• tightly ~~fixed~~ constrained
• part of large systems
• Reactive & real time
• programmable.

• embedded system scheduling → power optimized scheduling algo.
→ ~~fixed priority algo~~.

Algo → 1) Fixed priority algo
• priority task → assigned → design time
• Task → exist → blocked or waiting state.
• simple lifecycle is followed here.

2) Dynamic priority algo → • priority of task → changes dynamically
• rest is same as fixed priority.

• sorting algo for embedded system

• characteristics → should be in place
→ not be recursive
→ code size → as per need of problem.
→ running time → increase linearly or logarithmic with no. of elements sorted.

Insertion sort used here → Best time complex → O(n)
Avg → O(n²)
worst → O(n²)