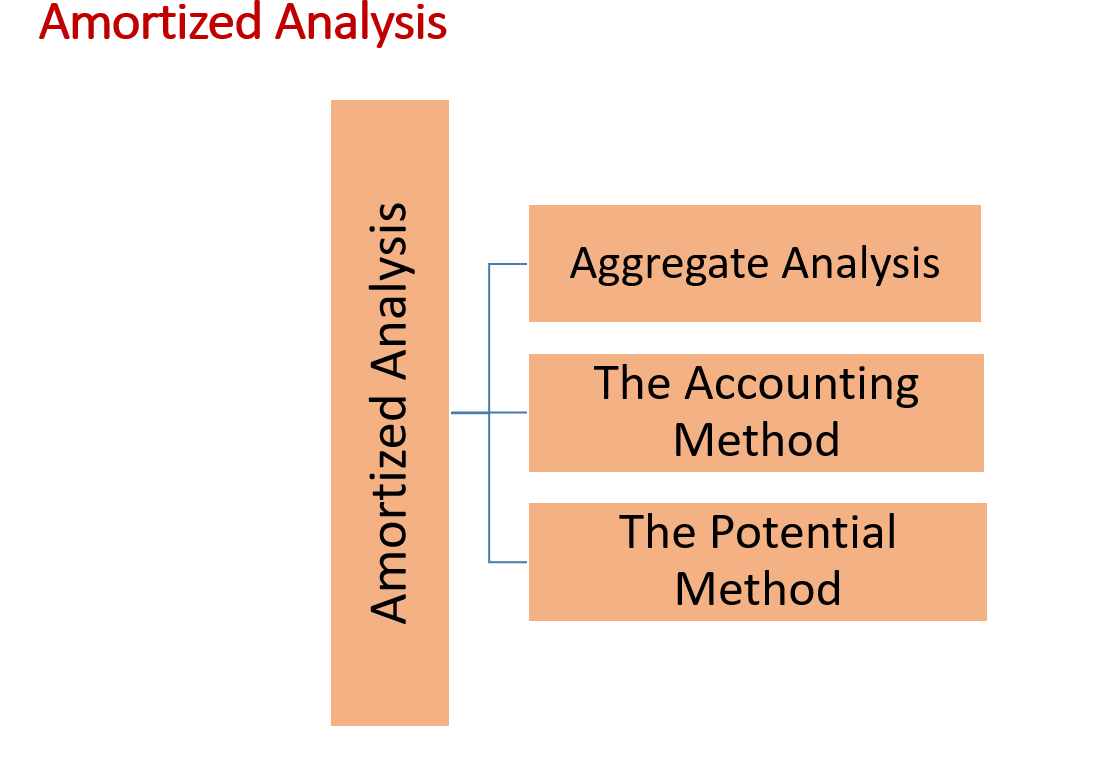
\*\*Amortized Analysis\*\*

- \*\*Definition:\*\* Amortized analysis is a technique used to analyze the average time or cost of a sequence of data structure operations, where we average the cost over all operations performed.

- \*\*Objective:\*\* The goal of amortized analysis is to demonstrate that even though some individual operations might be costly, the average cost per operation is relatively low.

- \*\*Average in Context:\*\* In this context, "average" doesn't involve a distribution of inputs or probabilities. It simply means calculating the cost per operation over a sequence.

- \*\*Average Cost in the Worst Case:\*\* Amortized analysis focuses on the worst-case scenario for the entire sequence of operations, not individual operations.



\*\*Two Problems:\*\*

1. \*\*Stack Operations, Including Multipop:\*\* Analyzing the cost of a sequence of stack operations, including multipop.

2. \*\*Binary Counter Using Increment Operation:\*\* Analyzing the cost of incrementing a binary counter.

\*\*Simple Analysis (Stack Operations):\*\*

- Starting with an empty stack, the stack's size can be at most O(n).

- A multipop operation might appear to have a complexity of O(n) in isolation.

- If there are n operations, the upper bound on complexity appears to be O(n^2).

- However, this upper bound is an overestimate and not a realistic reflection of the actual cost.

\*\*Aggregate Analysis (Stack Operations):\*\*



- Considering a sequence of n PUSH, POP, and MULTIPOP operations.

- The worst-case cost of MULTIPOP is O(n).

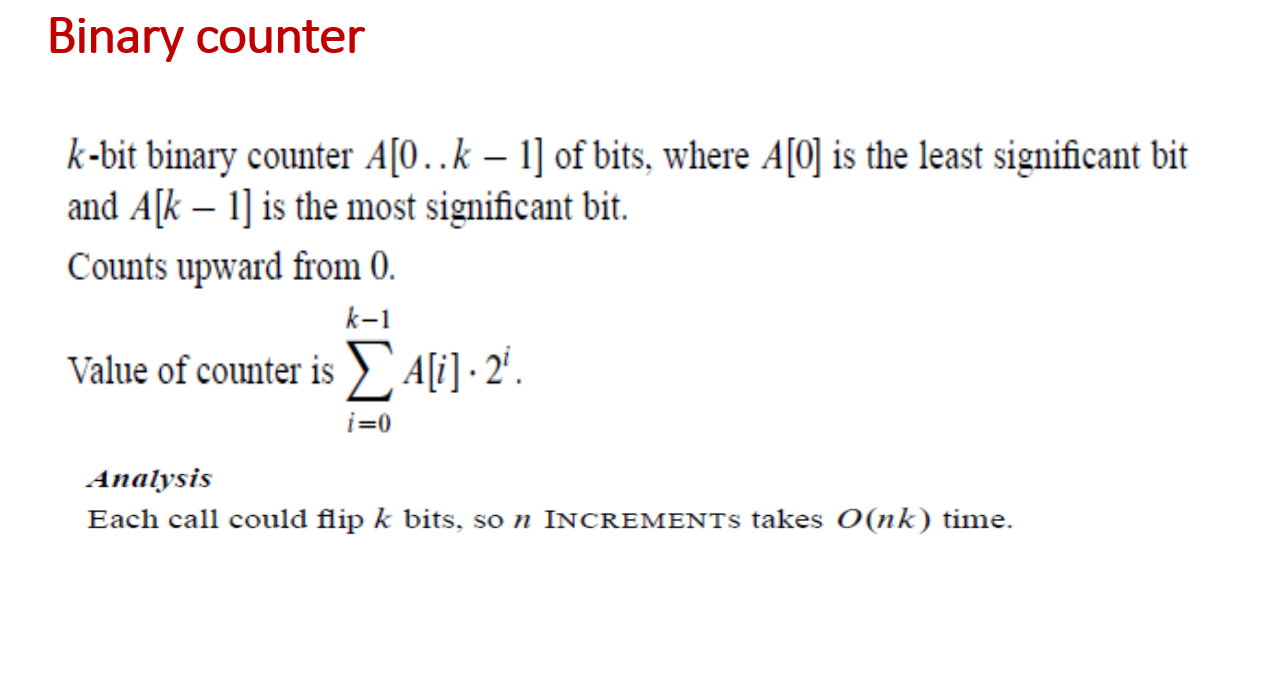
- Since there are n operations in total, the worst-case cost of the sequence seems to be O(n^2).

- Observation: Each object can only be popped once for every time it's pushed.

- As there are n PUSHes, there will be n POPs, including those in MULTIPOP.

- Therefore, the total cost is O(n).

- When averaged over the n operations, the cost per operation on average is O(1).

****

A screenshot of a computer

Description automatically generatedA math equations and numbers

Description automatically generated with medium confidenceA screenshot of a computer

Description automatically generated

\*\*Simple Analysis for Binary Counter:\*\*

In this analysis, we are measuring the cost of operations in terms of the number of bit flips in a binary counter. Some operations flip only one bit, while others can ripple through and flip multiple bits. The goal is to find the average cost per operation.

- \*\*Cursory Analysis:\*\*

- The worst-case cost for the increment operation is O(k), where k is the number of bits in the binary counter.

- If we have a sequence of n increment operations, it might seem that the worst-case behavior would be O(n \* k).

- However, this upper bound is not very tight and might overestimate the actual cost.

\*\*Amortized Analysis for Binary Counter:\*\*

In an amortized analysis, we consider that not all bits are flipped in each iteration of the increment operation. Some bits flip more frequently than others, following a pattern:

- A[0] is flipped every iteration.

- A[1] is flipped every other iteration.

- A[2] is flipped every fourth iteration, and so on.

- For i > ⌊log₂n⌋, the bit A[i] never flips.

By summing all the bit flips, we can determine the worst-case cost:

- The first ⌊log₂n⌋ bits flip in various patterns, but their total flips are bounded by a constant.

- For the remaining bits (i > ⌊log₂n⌋), they never flip.

- So, the worst-case cost is bounded by O(n).

Now, to find the average cost per operation, we divide this worst-case cost (O(n)) by the number of operations (n):

- O(n) / n = O(1).

Therefore, in the amortized analysis, the average cost per increment operation on the binary counter is O(1), which is much tighter and more accurate than the initial worst-case estimate of O(n \* k).