Amortized Analysis

Amortized analysis is used to show that the average cost of an operation is small even though a single operation may be expensive.

Amortized analysis differs from average case analysis in that probability is not involved. "Amortized analysis guarantees the average performance of each operation in the worst case."

Example. Consider the following stack operations:

- push(S,x);
- pop(S);
- multipop(S,k)

-- pop the k top objects of stack S, or the entire stack if S contains < k objects:

```
while not stack_empty(S) and k<>0 do
    begin
    pop(S);
    k := k-1;
    end;
```

Question: What is the running time of multi_pop(S,k)?

- # iterations of loop = min(s,k), where s = # items on stack S.
- In a sequence of n operations, multi_pop is O(n) since the stack size is at most n. => sequence of n operations costs $O(n^2)$.

This analysis is correct, but the bound is not tight. We can get a better bound using amortized analysis.

We will study an approach to amortized analysis called the *potential method.*

The potential method: Determine the amortized cost of each operation and possibly overcharge/prepay on operations performed early on to compensate for undercharges later.

The prepaid work is represented as the "potential" that can be released to pay for future operations.

Notation.

D_0 = initial data structure on which n operations are performed.

c_i = actual cost of i^{th} operation.

D_i = data structure resulting after applying the i^{th} operation to data structure D_{i-1}.

Φ = potential function which maps data structure D_i to real number $\Phi(D_i)$, which is the potential associated with data structure D_i.

Definitions.

Amortized cost $\hat{c}_i$ of i^{th} operation w.r.t. potential function Φ:

$\hat{c}_i = c_i + \Phi(D_i) - \Phi(D_{i-1})$ (1)

= (actual cost) + (increase in potential)

Total amortized cost of n operations:

sum from i=1 to n of ($\hat{c}_i$) = (2)

sum from i=1 to n of ($c_i+\Phi(D_i)-\Phi(D_{i-1})$) =

sum from i=1 to n of ($c_i+\Phi(D_n)-\Phi(D_0)$) (* $\Phi(D_i)$'s telescope! *)

Applying the potential method:

- Try to find a potential function Φ such that $\Phi(D_n) >= \Phi(D_0)$.
- Then, the total amortized cost: sum from i=1 to n of ($\hat{c}_i$)
  is an upper bound on the total actual cost.
- This is true since

sum from i=1 to n of (c_i) =

sum from i=1 to n of ($\hat{c}_i$ - ($\Phi(D_n)$ -$\Phi(D_0)$))

(So $c_i = \hat{c}_i$ - (x>0))

To guarantee $\Phi(D_n) >= \Phi(D_0)$, we often define $\Phi(D_0)=0$ and show that $\Phi(D_i) >= 0$ for all i.

Intuition.

- If the potential difference $\Phi(D_i) - \Phi(D_{i-1})$ of the i^{th} operation is positive, then $\hat{c}_i$ represents an overcharge to the i^{th} operation. ($\hat{c}_i = c_i + (x>0)$)
- If the potential difference $\Phi(D_i) - \Phi(D_{i-1})$ of the i^{th} operation is negative, then $\hat{c}_i$ represents an undercharge to the i^{th} operation ($\hat{c}_i = c_i + (x<0)$) and the actual cost of the operation is paid by the decrease in potential.

So overcharge, but not too much!

Example. For push, pop, and multi_pop define the potential function to be the number of objects on the stack.

- $\Phi(D_0) = 0$, where D_0 is the empty stack.
- $\Phi(D_i) >= 0$, since the number of objects on the stack is never negative.

Let the i^{th} operation be:

push: $\Phi(D_i) - \Phi(D_{i-1}) = (s+1) - s = 1$

$=> \hat{c}_i = c_i + \Phi(D_i) - \Phi(D_{i-1}) = 1 + 1 = 2.$

pop: $\Phi(D_i) - \Phi(D_{i-1}) = (s-1) - s = -1$

$=> \hat{c}_i = c_i + \Phi(D_i) - \Phi(D_{i-1}) = 1 - 1 = 0.$

multi_pop: Let k' = min(s,k) objects to be popped off the stack.

=> actual cost = k' and $\Phi(D_i) - \Phi(D_{i-1}) = -k'$.

Thus, $\hat{c}_i = c_i + \Phi(D_i) - \Phi(D_{i-1}) = k' - k' = 0.$

Conclusion:

- The amortized cost of each of the three operations is O(1). Thus, the total amortized cost of a sequence of n operations is O(n).
- Since $\Phi(D_n) >= \Phi(D_0)$, the total amortized cost of n operations is an upper bound on the total amortized cost.

=> Worst case cost of n operations is O(n).

Lessons learned:

1) Finding a potential function is tricky. Ensure that the function is initially 0 and always nonnegative, (e.g. the number of trees after an insert).
2) In general, T_{actual} + (change in potential) = $T_{amortized}$.
   If the final potential (over the entire sequence) is at least as large as the initial potential, then the amortized time is an upper bound on the actual time.
3) The potential function should cancel a term in the actual time, e.g., the actual time is c and the potential is 2-c. when these add, an amortized cost of 2 is obtained.
4) The potential function should capture the aspect of the time/cost that varies from operation to operation.