Amortized Analysis

Instructors: Subrahmanyam Kalyanasundaram
Karteek Sreenivasaiah

Week 10

Last Week

We saw:

- Union Find (Disjoint Sets) Data Structure
- Heuristics & Analysis
- Path Compression

Last Week

We saw:

- Union Find (Disjoint Sets) Data Structure
- Heuristics & Analysis
- Path Compression

Theorem

A sequence of m operations in total, n of which are MakeSet takes $O(m\alpha(n))$ time where $\alpha(n)$ is the inverse Ackermann.

Today

Amortized Analysis

Amortized Analysis

- Considering the application, we may not need each individual operation to be fast.
- What we may want is altogether, the operations to be fast.
- Trade-off between “per-operation" efficiency and “overall” efficiency.
- Would like to make statements: “The total runtime of the algorithm is $O(n \log n)$, even though there are $\Theta(n)$ steps, and each step has worst case run time of $\Theta(n)$.

Amortized Analysis

Usually, one of the following three techniques are used.

- Aggregate Method: Compute the total cost of a sequence of operations, and the average.
- Accounting Method: Assign charges to operations. Extra charges are redeemed for operations that cost more.
- Potential Method: A potential function is computed on the current state of the data structure. Costly operations are charged from the decrease of potential function.

All of these achieve the same objective, we are just counting in different ways.

Aggregate Method

- Compute the total cost of a sequence of operations.
- Divide it by number of operations.

Accounting Method

- Amortized cost is $\hat{c}_i$ and actual cost is c_i, such that for all n,

$$\sum_{i=1}^{n} \hat{c}_i \geq \sum_{i=1}^{n} c_i$$

- For individual i, we may have $\hat{c}_i \geq c_i$ or $\hat{c}_i < c_i$.

- For each i, $\hat{c}_i - c_i$ can be viewed as extra credits added by the operation.
- For each i, $c_i - \hat{c}_i$ can be viewed as the credits spent/redeemed by the operation.

Potential Method

- Let Φ be a potential function associated with the current state of the data structure
- Let D_i be the data structure's state at time i
- For each i, we have

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

- Amortized cost = Actual cost + Potential difference
- Need to ensure that $\Phi(D_i) \geq \Phi(D_0)$ always

Examples

- Two-stack queue
- 2-3-4 Trees

On other slides