Amortized Analysis

Applications: Data Structures
  Eg Binomial Heaps, Fibonacci Heaps
  Union-Find, Splay Trees

DS Applications: Dijkstra's Alg (shortest path)
  Kruskal's Alg (MST)

See CLRS Ch 17 for intro

Main Trick: Lazy DS Versus Eager DS
Timing Analysis Methods

Worst Case: eg Strassen
Average Case (average over inputs):
  eg Quick Sort pivoting on first element
Amortized Analysis (worst case input, averaged over time):
Randomized (worst case input, averaged over coin flips):
  eg (QS Randomized)

Amor Anal CHRS Chap 17

3 Methods
  Aggregation Method (won't use)
  Accounting Method (Kozen)
  Potential Method (will use)
Example: Dijkstra's Alg

Recall: Dijkstra computes single source shortest path
\[ \text{dist}(s, v) \forall v \in V \# \text{vertices} = n \quad \# \text{edges} = m \quad m > n \]

It uses priority queues:

**Input:** set of keys \( K_1, \ldots, K_n \)
with priorities \( \text{Prior}(K_1), \ldots, \text{Prior}(K_n) \)

| Dijkstra | \# ops | **Worst case for Heap** | Total \# inv
|-----------|--------|-------------------------|------------------|
| make heap \((S)\) | 1     | 1                       | 1
| find min \((S)\) | \(n\) | 1                       | \(n\)
| insert \((K, S)\) | \(n\) | \(\log n\)              | \(n\log n\)
| delete min \((S)\) | \(n\) | \(\log n\)              | \(n\log n\)
| decrease key | \(m\) | \(\log n\)              | \(m\log n\)     |
Consider some operation $O_p$

Suppose it is called $n$ times

Costing $O_p$, $-\rightarrow O_{pn}$

Goal: total cost $= \sum_{i=1}^{n} O_{pi} = TC$

Simplest idea: compute $WC = \max_{i} O_{pi}$

then $TC \leq n \cdot WC$

2 Problems:

1) $n(WC)$ too big (over estimate)

2) We won't even consider better alg.

Idea 2: Compute $Avg = \{O_{p1}, \ldots, O_{pn}\}$

then $TC \leq n \cdot Avg$

Prob: Hard to estimate $Avg$!
Trick: Add an artificial cost per operation!

\[ \Phi : \text{state} \to \mathbb{R} \, \text{a potential} \]

\[ \text{Def. unit-cost} = \Omega_i \]

\[ \text{Def. Amortized Cost} = \text{unit-cost} + \text{potential change} \]

\[ \Sigma \text{A}C_i = \Sigma [\Omega_i + (\Phi_i - \Phi_{i-1})] = \Sigma \Omega_i + \Phi_n - \Phi_0 \]

\[ = TC + \Delta \Phi \]

if \( \Delta \Phi \geq 0 \) then \( TC \leq \Sigma AC_i \)

\[ AC = \max_i AC_i \quad \text{then} \]

\[ TC \leq n \cdot AC \quad \text{if} \quad \Delta \Phi \geq 0 \]
# Priority Queues

Heaps (E.g., Binomial, Fibonacci)

CLRS Chaps 6, 19, 20  Kozan 8, 9

<table>
<thead>
<tr>
<th>Abstract Data Type</th>
<th>Binomial Eager/Lazy</th>
<th>Fibonacci Amort</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>make-heap(s)</code></td>
<td>$O(1)$</td>
<td>$O(1)$</td>
</tr>
<tr>
<td><code>find-min(s)</code></td>
<td>$O(1)$</td>
<td>$O(1)$</td>
</tr>
<tr>
<td><code>insert(k,s)</code></td>
<td>$O(\log n)$</td>
<td>$O(\log n)$ / $O(1)$</td>
</tr>
<tr>
<td><code>delete-min(s)</code></td>
<td>$O(\log n)$</td>
<td>$O(\log n)$</td>
</tr>
<tr>
<td><code>Union-Meld(A,B)</code></td>
<td>$O(n)$</td>
<td>$O(\log n)$ / $O(1)$</td>
</tr>
<tr>
<td><code>decrease-key(i,k,s)</code></td>
<td>$O(\log n)$</td>
<td>$O(\log n)$</td>
</tr>
</tbody>
</table>
Regular Heap (recall)

Heap: 1) Balanced Binary Tree, T (Heap)
2) \( \text{Priority}(p(k)) \leq \text{Priority}(k) \) (Heap property)
   \( p(k) = \text{Parent}(k) \)

\begin{align*}
\text{makeheap} & \equiv \text{make empty tree} \\
\text{findmin} & \equiv \text{return root} \\
\text{insert} & \equiv 1) \text{ add new leaf} \\
& \quad 2) \text{ let it "float up"} \\
\text{deletemin} & \equiv 1) \text{ replace root with a leaf} \\
& \quad 2) \text{ "float down" root} \\
\text{meld} & \equiv ?
\end{align*}
Binomial Heaps (cheap meld)

**Iden:**
1) balanced $\rightarrow$ almost balanced
2) binary $\rightarrow$ (log\_n)-ary
3) link

$$\begin{array}{c}
\text{Problem:} 1 \quad 1 \\
\quad T_1 \quad T_n \\
\quad T'_1 = \text{Meld}(T_1, T_2)
\end{array}$$

$$\text{Meld}(T_{n-1}, T_n)$$

we get

(not balanced?)
Idea: Meld (link) trees of same size

**Def** Binomial Tree

\[ B_0 = \cdot \quad B_1 = \cdot \quad B_2 \quad \cdot \quad B_{k+1} \]

**Note**

\[ B_{k+1} \]

**Claim:** \( \text{depth}(B_k) = k \) &

\[ |B_k| = 2^k \]

Pointers stored
Why Binomial?

Definition: \( B(k,i) = \# \text{ nodes at depth } i \)

Claim: \( B(k,i) = \binom{k}{i} \) 
Proof: \( \binom{n}{i} = \binom{n-1}{i} + \binom{n-1}{i-1} \)

Eager Binomial Heap

- Link\((A,B)\): 1) combine 2 trees of same rank
- 2) increment rank of root

Meld\((\overline{A},\overline{B})\): Link until one tree per rank

Example: \( \overline{A} = A_0 \ A_1 \ A_2 \ A_3 \ A_4 \)
\( \overline{B} = B_0 \ B_1 \ B_4 \)
\( \overline{C} = C_1 \ C_2 \ A_3 \ C_5 \)
\[ \text{insert}^+ (k, \bar{A}) \equiv \]
1) make heap \((k, B)\)
2) meld \((\bar{A}, \bar{B})\)

\[ \text{delete}^\text{min} (\bar{A}) \equiv \]
1) Suppose \(\bar{A} = (T_1, \ldots, T_K)\) (trees)
2) Find tree with min root, \(T_i\).
3) remove \(T_i\) from \(\bar{A}\)
4) remove root \((T_i)\) giving \(B H \bar{B}\)
5) return meld \((\bar{A}, \bar{B})\) (Eager)

Example:
\[ \bar{A} = (T_0, T_1, T_2, T_3, T_4) \]
where

\[
\begin{array}{c}
0 \quad 1 \quad 2 \quad 3 \quad 4 \\
\end{array}
\]

\[
\begin{array}{c}
\text{min key} \\
T'_3 \quad T'_2 \quad T'_1 \quad T'_0
\end{array}
\]
\text{delete}(A) \quad \bar{A} = (T_0, T_1, T_2, T_3)
\bar{B} = (T_0', T_1', T_2', T_3')

\text{meld}(\bar{A}, \bar{B}) = (T_1'', T_2'', T_3'', T_4'')

\text{Note: delete} \text{ in } O(\log n) \text{ time.}

---

Amortized analysis of insert

Use potential argument

\[ \Phi(\bar{A}) = \# \text{trees} \]

\[ \text{Def: unit-cost of insert = \#trees linked + 1} \]
\[ = \# \text{trees removed + 1} \]

\[ \text{Amort. Cost} = \text{unit-cost} + \Delta \Phi \]
\[ = 1 + \# \text{trees removed} - (\# \text{tree removed} + 1) \]
\[ = 2 \]
Note: Amort. cost of delete-emin still $O(\log n)$.

Lazy Melds

1) Trees are kept as a linked list.
2) Lazy-Meld = list concatenation. (Destructive)
3) delete-emin =
   a) link till at most one per rank.
   b) do old-delete-emin.

Amort. Analysis of Lazy Meld.

$$\Phi(\bar{A}_1, \bar{A}_2, \ldots, \bar{A}_x) = 2 \# \text{trees}$$

unit cost of delete-emin in $\bar{A} = m+k$

$m = \# \text{trees in } \bar{A}$

$k = \# \text{links} = \# \text{trees deleted}$

Amort. Cost = $(m+k) + \Delta E = m+k - 2k = m-k = \text{remaining trees} \leq \log n$