Binomial Heaps

CLRS, Chapters 6, 19

Thanks to Kevin Wayne for the slides.

Priority Queues

(heaps)

Supports the following operations.

- Insert element x.
- Return min element.
- Return and delete minimum element.
- Decrease key of element x to k.

Applications.

- Dijkstra's shortest path algorithm.
- Prim's MST algorithm.
- Event-driven simulation.
- Huffman encoding.
- Heapsort.
- ...

Priority Queues in Action

HeapInit() /*Initialized Priority Queue */

for each v ∈ V

d[v] ← ∞;

HeapInsert(v)

while (!HeapIsEmpty())

u = Heap_Extract_Min()

for each v ∈ V(G) s.t (u,v) ∈ E(G)

if d[v] > d[u] + w(u,v)

Heap_Decrease(v, d[u] + w(u,v))

Binomial Tree

- Recursive definition:
  - $B_i$ is created from two $B_{i-1}$ by
    - connect the root of the larger key is a child of the second.
    - So the degree could be rather large.

Binomial Tree - definition.

Useful properties of order k binomial tree $B_k$.

- Number of nodes = $2^k$.
- Height = k.
- Degree of root = k.
- Deleting root yields binomial trees $B_{k-1}, ..., B_0$.

Proof.

- By induction on k.
Binomial Tree

A property useful for naming the data structure.
- \( B_i \) has \( \binom{k}{i} \) nodes at depth \( i \).

Binomial Heap

- Sequence of binomial trees that satisfy binomial heap property.
  - each tree is min-heap ordered (parent \( \leq \) each child)
  - 0 or 1 binomial tree of order \( k \)

Binomial Heap: Implementation

- Represent trees using pointers to
  - left-sibling, right-sibling, parent, a child.
- (need a pointer only to one child)
- All siblings are in a doubly connected list, called sibling list (so it is enough to point to only one of them)
- Roots of trees connected with linked list.
  - degrees of trees strictly decreasing from left to right

Binomial Heap: Union

Create heap \( H \) that is union of heaps \( H' \) and \( H'' \).
- "Mergeable heaps."
- Easy if \( H' \) and \( H'' \) are each order \( k \) binomial trees.
  - connect roots of \( H' \) and \( H'' \)
  - choose smaller key to be root of \( H \)
Binomial Heap: Union

Create heap $H$ that is union of heaps $H'$ and $H''$.

- Analogous to binary addition.

Running time. $O(\log N)$

- Proportional to number of trees in root lists $\leq 2\lfloor \log_2 N \rfloor + 1$.

Binomial Heap: Delete Min

Delete node with minimum key in binomial heap $H$.

- Find root $x$ with min key in root list of $H$, and delete
- $H' \leftarrow$ broken binomial trees
- $H \leftarrow$ Union($H', H$)

Running time. $O(\log N)$

Binomial Heap: Decrease Key

Decrease key of node $x$ in binomial heap $H$.

- Suppose $x$ is in binomial tree $B_k$
- Bubble node $x$ up the tree if $x$ is too small.
- Running time: $O(\log N)$
- Proportional to depth of node $x \leq \lfloor \log_2 N \rfloor$.

Binomial Heap: Delete

Delete node $x$ in binomial heap $H$.

- Decrease key of $x$ to $\infty$.
- Delete min.

Running time. $O(\log N)$

Binomial Heap: Insert

Insert a new node $x$ into binomial heap $H$.

- $H' \leftarrow$ MakeHeap($x$)
- $H \leftarrow$ Union($H', H$)

Running time. $O(\log N)$
Binomial Heap: Sequence of Inserts

Insert a new node \( x \) into binomial heap \( H \).
- If \( N = \ldots 0 \), then only 1 steps.
- If \( N = \ldots 01 \), then only 3 steps.
- If \( N = \ldots 011 \), then only 5 steps.
- If \( N = \ldots 0111 \), then only 7 steps.

Inserting 1 item can take \( \Omega(\log N) \) time.
- If \( N = 11\ldots11 \), then \( \log_2 N \) steps.

But, inserting sequence of \( N \) items takes \( O(M) \) time!
- \( (N/2)(1) + (N/4)(2) + (N/8)(3) + \cdots \leq 2N \)

Amortized analysis.
- Basis for getting most operations down to constant time.

\[
\frac{\log_2 N}{2} = \frac{2 - \frac{N}{2^2}}{2} \leq 1
\]