Fundamental Algorithms

Lecture #9

Cluj-Napoca

November, 27, 2019

Agenda

- Binomial Heaps & Binomial Trees
  - Def
  - Basic operations
- Fibonacci Heaps
- MT analysis

Binomial Trees

- Degree-based augmented trees; denoted B_k
- Degree of the tree = number of descendants of the tree
- Properties of a B_k tree:
  - P1: Degree of the root (B_k) = k;
  - P2: Number of nodes (B_k) = 2^k;
  - P3: Height(B_k)=k;
  - P4: Number of nodes at level i in (B_k) is $C_k^i = \binom{k}{i}$
  - P5: If children of (B_k) are numbered from the left to the right as (k-1, k-2, …, ,0), then child i is a (B_i) tree
- Recursive definitions:
  - A B_k tree is 2 B_{k-1} - trees, with their roots linked (picture on the blackboard)
- Be aware of the implication the equivalence definitions following P5 and the recursive definition!

Binomial Trees

- **Recursive definitions:**
  - **A B_k tree is 2 B_{k-1} - trees, with their roots linked (picture on the blackboard)**
  - **A B_k tree is collection of k trees: B_{k-1}, B_{k-2}, …, B_1, B_0 – trees (picture on the blackboard)**
- **Proof of P4: Number of nodes at level i of (B_k) is C_k^i**
  - **Goes by induction**
  - **On level i we have nodes from 2 B_{k-1} trees**
    - **From the first tree (containing the root of B_k) #nodes at level i = C_{k-1}^i**
    - **From the second one #nodes at level i-1 (one level less; level measured from the root) = C_{k-1}^{i-1}**
    - **So there are $C_{k-1}^i + C_{k-1}^{i-1} = C_k^i$ nodes at level i in B_k**

Binomial Heaps

- **Binomial Heap (H) = A set of Binomial trees with the following properties:**
  - **P1: each node has a key;**
  - **P2: each binomial tree in H is heap-ordered (min on top);**
  - **P3: for any k, there is at most one B_k tree in H.**

- **Consequence: if H has n nodes, it has at most $\lfloor lgn \rfloor +1$ binomial trees.**
  - **Justification:**
    - **Max number of trees a H may have = one of each type**
    - **If each type of tree is present, the number of nodes for B_{k-1}, B_{k-2}, ..., B_1, B_0 is 2^{k-1}, 2^{k-2}, ..., 2^0 respectively**
    - **Nb of nodes of H is their sum = $2^{k-1}+ 2^{k-2}+\ldots+2^0 = 2^k -1$**
    - **Denote 2^k = n. H has n nodes, and k tees ($\lfloor lgn \rfloor +1$**

Binomial Heaps – Operations
(|H|=n) (for all, examples on the blackboard)

- **Make-Heap** **O(1)**
  - **Builds an empty Binomial Heap**
- **Binomial-Heap-findMinimum(H)** **O(lgn)**
  - **Returns the pointer to the root of the B with the min key (NOT removed);**
- **Binomial-Heap-Unite(H1, H2) =**
  merge + links O(lgn)
  - **merge – merges 2 rooted lists (H1, H2) into a single one sorted by degree (increasing order)** **O(lgn)**
  - **link – changes a pair of B_{k-1} trees into a B_k tree** **O(1)**
  - **Unite = merge + links from left to right** **O(lgn)+lgnO(1)**

Binomial Heaps – Operations
(|H|=n) (for all, examples on the blackboard)

- **Binomial-Heap-extractMinimum(H) O(lgn)**
  - **Binomial-Heap-findMinimum(H) O(lgn)**
  - **removes that tree from H O(1)**
  - **Make a heap out of the binomial tree containing the min key =>H1in O(lgn)**
  - **Binomial-Heap-Unite(H, H1) O(lgn)**

- **Binomial-Heap-keyDelete(H1, H2) O(lgn)**
  - **As if we were to extract min.**
  - **How?**
    - **Decrease the key to delete to -∞+ O(1)**
    - **restore the heap property of the Binomial tree + O(lgn)**
    - **Extractmin O(lgn)**

Fibonacci Heaps

- Collection of ordered trees
- Properties: like Binomial Heaps with some constraints added/removed.
- Relaxed constraints:
  - May contain several trees of the same degree
  - Rooted, yet unordered
- Added constraints:
  - Children at a given level in a tree are linked to each other (left/right) in a circular, doubly linked list (child list)
  - Node augmentation: degree[x] = number of children in the child list of x
  - The heap maintain a pointer (min[H]) to the root of the tree containing the min key.
- Operations:
  - Like for Binomial Heaps (Hw)
  - Obs: due to relaxations, operations faster:
    - Insert node – a node which is a tree, at the top level
    - Find min – has a pointer
    - Union – simpler, since they are not ordered

Binomial/Fibonacci Heaps Comparative analysis

Operation	Binomial	Fibonacci
Make heap	O(1)	O(1)
Insert	O(lgn)	O(1)
Find min	O(lgn)	O(1)
Extract min	O(lgn)	O(lgn)
Union	O(lgn)	O(1)
Decrease key	O(lgn)	O(1)
delete	O(lgn)	O(lgn)

MT analysis – error types

- P1
- P2