Algorithm Theory

07 – Binomial Queues

Dr. Alexander Souza

Priority queues: operations

(Priority) queue *Q*

Data structure for maintaining a set of elements, each having an associated priority from a totally ordered universe. The following operations are supported.

key ≡ priority

Operations:

Q.initialize(): initializes an empty queue *Q*

Q.isEmpty(): returns true iff *Q* is empty

Q.insert(e): inserts element *e* into *Q* and returns a pointer to the node containing *e*

Q.deletemin(): returns the element of *Q* with minimum key and deletes it

Q.min(): returns the element of *Q* with minimum key

Q.decreasekey(v,k): decreases the value of *v*'s key to the new value *k*

Priority queues: operations

Additional operations:

Q.delete(v): deletes node *v* and its element from *Q*
(without searching for *v*) we have to store the pointer to the node which we get when we insert the element

Q.meld(Q´): unites *Q* and *Q´* (concatenable queue)

Q.search(k): searches for the element with key *k* in *Q* (searchable queue)

And many more, e.g. *predecessor, successor, max, deletemax*

Application: Shortest Path Problems

Priority queues: implementations

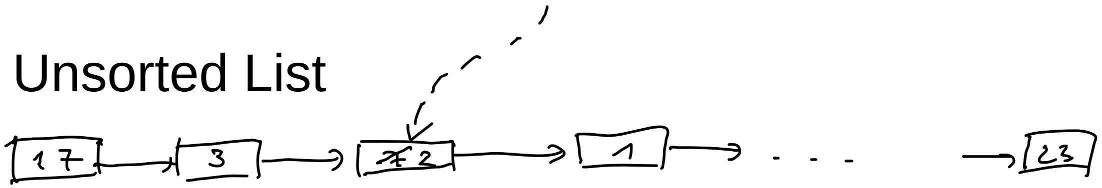
	List	Heap	Bin. – Q.	Fib.-Hp.
insert	O(1)	O(log n)	O(log n)	O(1)
min	O(n)	O(1)	O(log n)	O(1)
delete-min	O(n)	O(log n)	O(log n)	O(log n)*
meld (m≤n)	O(1)	O(n) or O(m log n)	O(log n)	O(1)
decr.-key	O(1)	O(log n)	O(log n)	O(1)*

*= amortized cost

Q.delete(e) = Q.decreasekey(e, -∞) + Q.deletemin( )

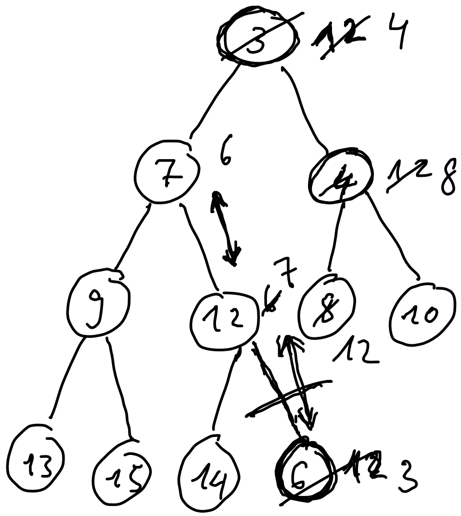
Unsorted List & Heap

Unsorted List



Insert $O(1)$ insert new element at the head
Min : $O(n)$ traverse the list
Delete Min : $O(n)$ Min + delete element
Meld : $O(1)$ append head to tail
Decrease Key : $O(1)$ change the key value. pointer to list-node is given.

Heap (Min-Heap)



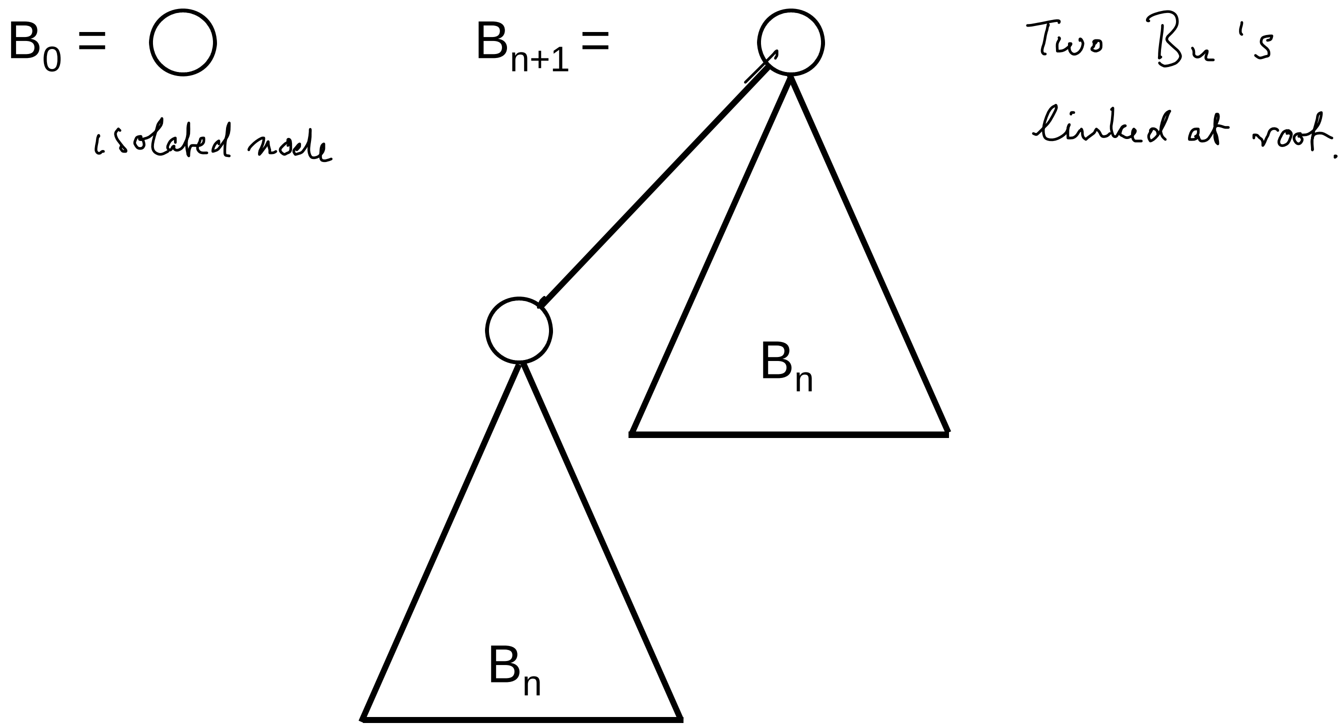
0	1	2								
3	7	4	9	12	8	10	13	15	14	6

$i \mapsto 2\cdot i+1,\ 2\cdot i+2$

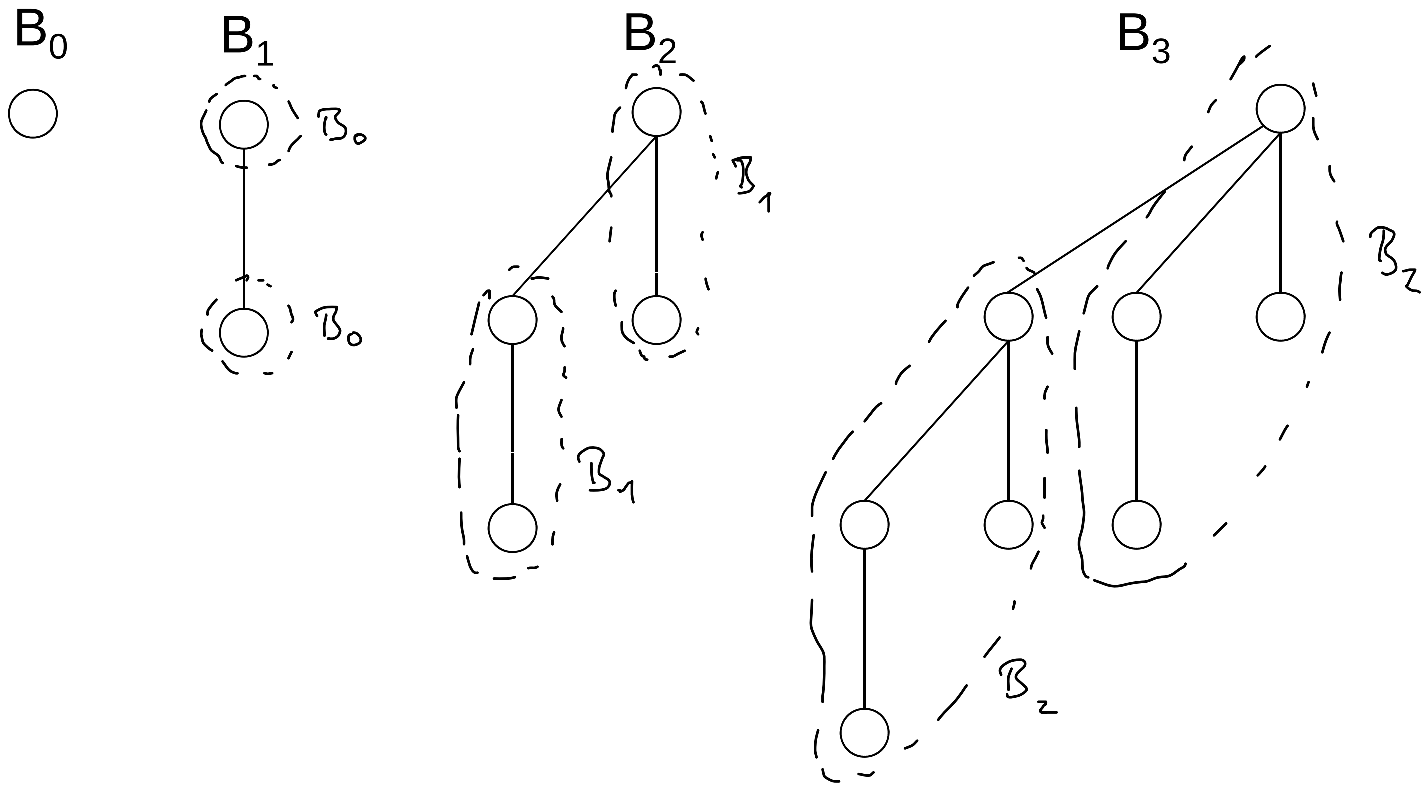
Insert: $O(\log n)$ insert as a leaf and restore heap property
Min: $O(1)$ return root value
Delete Min: $O(\log n)$ place value of last leaf at root, delete last leaf, restore heap property.
Meld: $O(n)$, $O(m \log n)$
Decrease Key: $O(\log n)$ change value and restore heap property

Definition

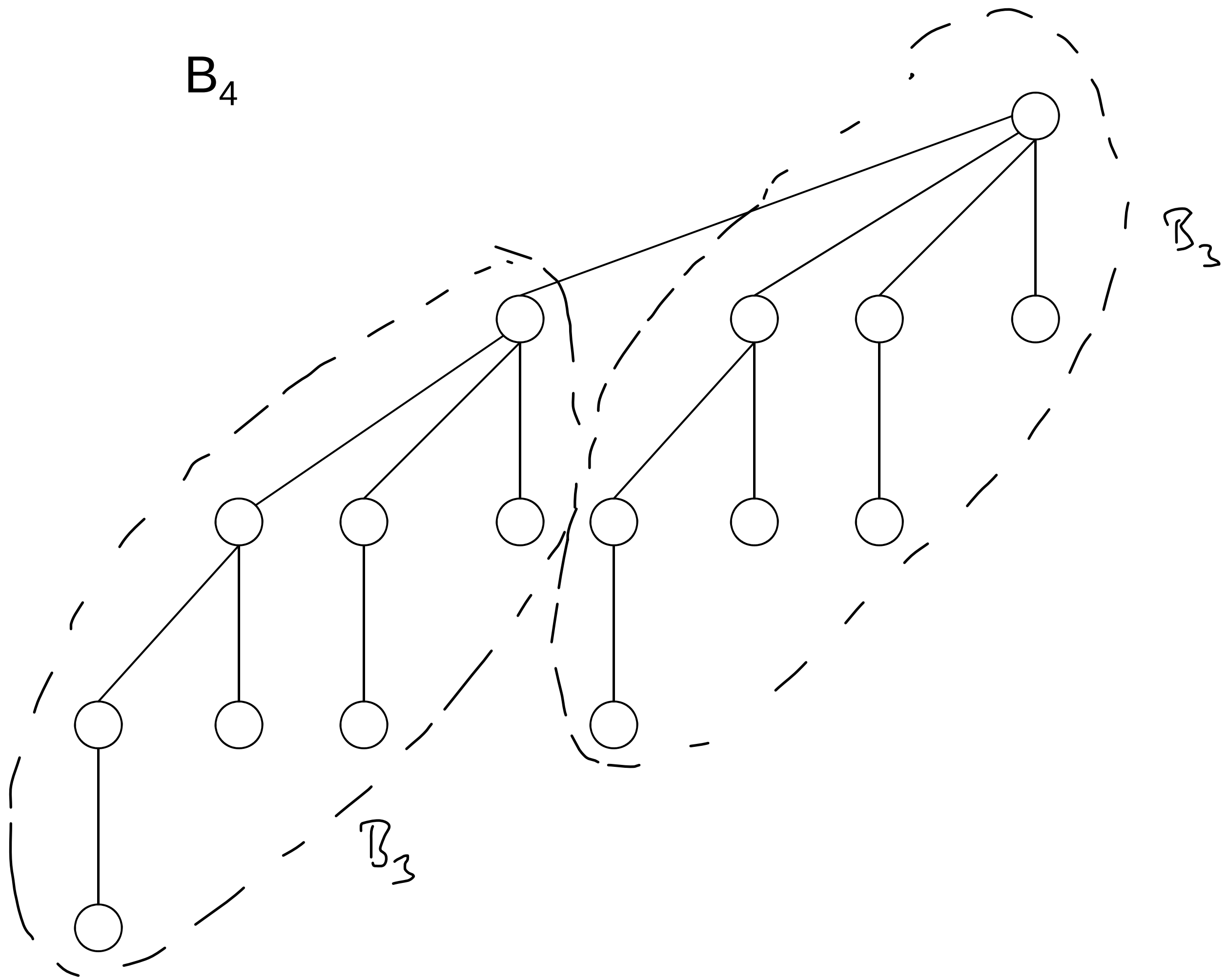
Binomial tree B_n of order n $(n \geq 0)$



Binomial trees

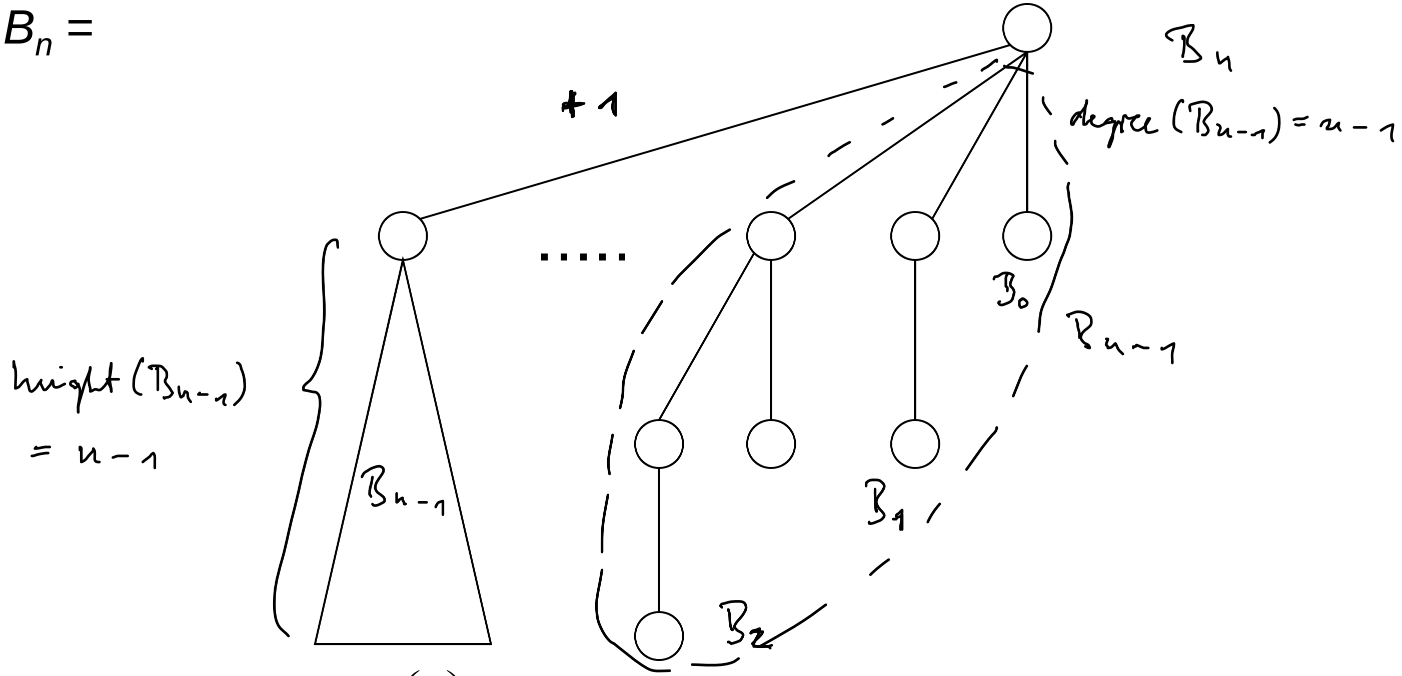


Binomial trees



Properties

1. B_n contains 2^n nodes. B_0 has $1 = 2^0$ nodes. We double the nodes whenever we increment the order of the tree.
2. The height of B_n is n.
3. The root of B_n has degree n.
4. B_n =



5. There are exactly $\binom{n}{i}$ nodes at depth i in B_n .

Binomial coefficients

$$\binom{n}{i} = \#\ i\text{-element subsets that can be chosen from an } n\text{-element set}$$

Pascal's triangle:

$$\begin{array}{ccccccccc}
 & & & & 1 & & & & \\
 & & & 1 & & 1 & & & \\
 & & 1 & & 2 & & 1 & & \\
 & 1 & & 3 & & 3 & & 1 & \\
1 & & 4 & & 6 & & 4 & & 1
\end{array}$$

$$\begin{array}{ccccccc}
 & & & \binom{0}{0} & & & \\
 & & \binom{1}{0} & & \binom{1}{1} & & \\
 & \binom{2}{0} & & \binom{2}{1} & & \binom{2}{2} & \\
\binom{3}{0} & & \binom{3}{1} & & \binom{3}{2} & & \binom{3}{3}
\end{array}$$

$$\frac{n!}{i!\,(n-i)!} = \binom{n}{i} = \binom{n-1}{i} + \binom{n-1}{i-1}$$

$$\parallel$$

$$\frac{(n-1)!}{i!\,(n-1-i)!} + \frac{(n-1)!}{(i-1)!\,(n-i)!}$$

$$\parallel$$

$$\frac{(n-1)!\,(n-i) + (n-1)!\,i}{i!\,(n-i)!}$$

$$\parallel$$

$$\frac{(n-1)!\,(n - \cancel{i} + \cancel{i})}{i!\,(n-i)!} = \frac{n!}{i!\,(n-i)!} \quad \checkmark$$

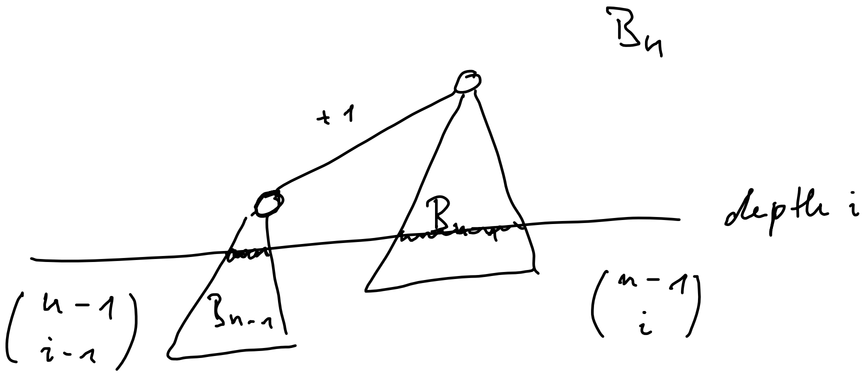
Number of nodes at depth i in B_n

There are exactly $\binom{n}{i}$ nodes at depth i in B_n.

Proof by induction

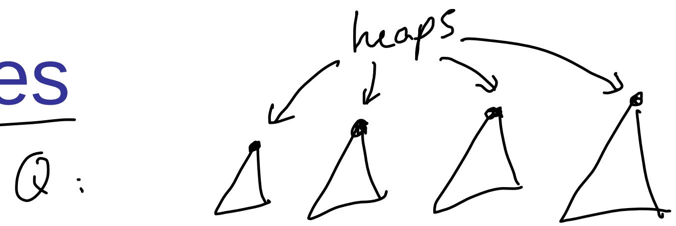
Base Case: $n = 0$ $\qquad \binom{0}{0} = \frac{0!}{0!\,0!} = 1$ ✓

Inductive Case: $n > 0$



$$\binom{n-1}{i} + \binom{n-1}{i-1} = \binom{n}{i}$$

Binomial queues



B_i 's are of different order

Binomial queue *Q*:

Set of heap ordered binomial trees of different order to store keys.

***n* keys:**

1 0 1 0 1 1 1 0

B_7 B_5 B_3 B_2 B_1

$B_i \in Q \iff$ i-th bit in $(n)_2 = 1$

Do we have sufficient space?

Yes: A B_i has 2^i many nodes

$n \mathrel{\hat{=}} b_k, b_{k-1}, \dots, b_0$

$n = \sum_{i=0}^{k} b_i \cdot 2^i$

9 keys:

{2, 4, 7, 9,12, 23, 58, 65, 85}

$9 = (1001)_2$

B_3 B_0