Chapter 4
Data Structures

Algorithm Theory
WS 2014/15

Fabian Kuhn

Priority Queue / Heap

- Stores (*key,data*) pairs (like dictionary)
- But, different set of operations:

- **Initialize-Heap**: creates new empty heap
- **Is-Empty**: returns true if heap is empty keys don't need to unique
- **Insert**(*key,data*): inserts (*key,data*)-pair, returns pointer to entry
- **Get-Min**: returns (*key,data*)-pair with minimum *key*
- **Delete-Min**: deletes minimum (*key,data*)-pair
- **Decrease-Key**(*entry,newkey*): decreases *key* of *entry* to *newkey*
- **Merge**: merges two heaps into one

Implementation of Dijkstra's Algorithm

Store nodes in a priority queue, use $d(s,v)$ as keys:

1. Initialize $\underline{d(s,s) = 0}$ and $\underline{d(s,v) = \infty}$ for all $v \neq s$
2. All nodes $v \neq s$ are unmarked

   n insert op.

3. Get unmarked node u which minimizes $\underline{d(s,u)}$:

   delete-min op

4. mark node u

5. For all $e = \{u,v\} \in E$, $\underline{d(s,v)} = \min\{\underline{d(s,v), d(s,u) + w(e)}\}$

   decrease-key

6. Until all nodes are marked

Analysis

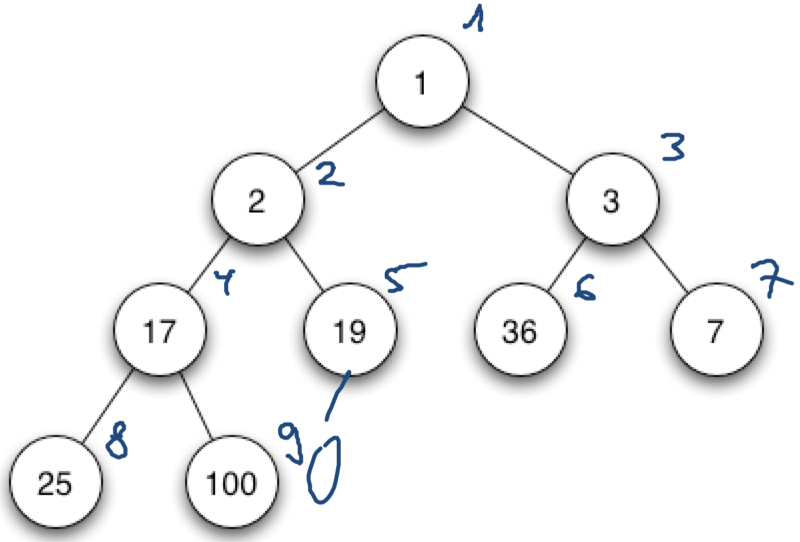
Number of priority queue operations for Dijkstra:

- **Initialize-Heap**: $\mathbf{1}$
- **Is-Empty**: $|\boldsymbol{V}|$
- **Insert**: $|\boldsymbol{V}|$
- **Get-Min**: $|\boldsymbol{V}|$
- **Delete-Min**: $|\boldsymbol{V}|$
- **Decrease-Key**: $|\boldsymbol{E}|$
- **Merge**: $\mathbf{0}$

Priority Queue Implementation

Implementation as min-heap:

→ complete binary tree,
e.g., stored in an array



- **Initialize-Heap**: $\boldsymbol{O(1)}$
- **Is-Empty**: $\boldsymbol{O(1)}$
- **Insert**: $\boldsymbol{O(\log n)}$
- **Get-Min**: $\boldsymbol{O(1)}$
- **Delete-Min**: $\boldsymbol{O(\log n)}$
- **Decrease-Key**: $\boldsymbol{O(\log n)}$
- **Merge** (heaps of size m and n, $m \leq n$): $\boldsymbol{O(m \log n)}$

Better Implementation

- Can we do better?

- Cost of Dijkstra with complete binary min-heap implementation:

$$O(|E|\log|V|) \qquad O(m \log n)$$

- Can be improved if we can make decrease-key cheaper…

  m decr.-key operations (in the worst case)

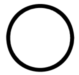
- Cost of merging two heaps is expensive

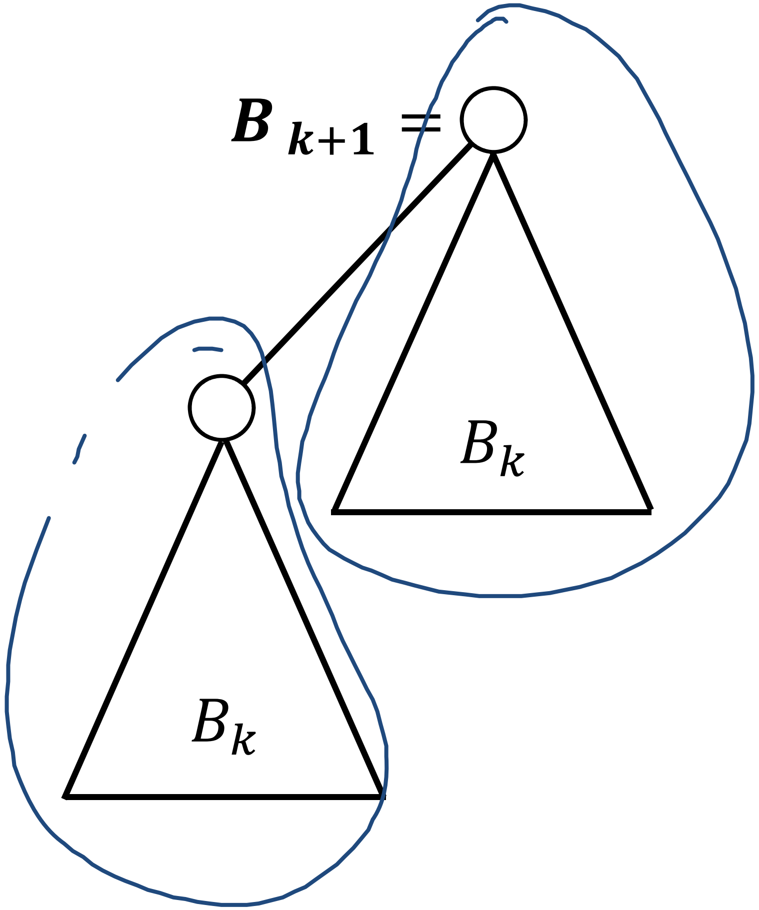
- We will get there in two steps:

  Binomial heap → Fibonacci heap

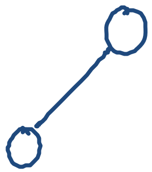
Definition: Binomial Tree

Binomial tree B_k of order k $(n \geq 0)$:

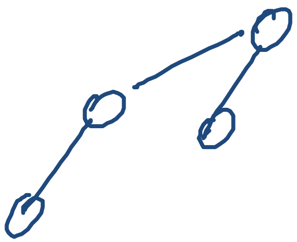
$B_0 =$ 

$B_{k+1} =$ 

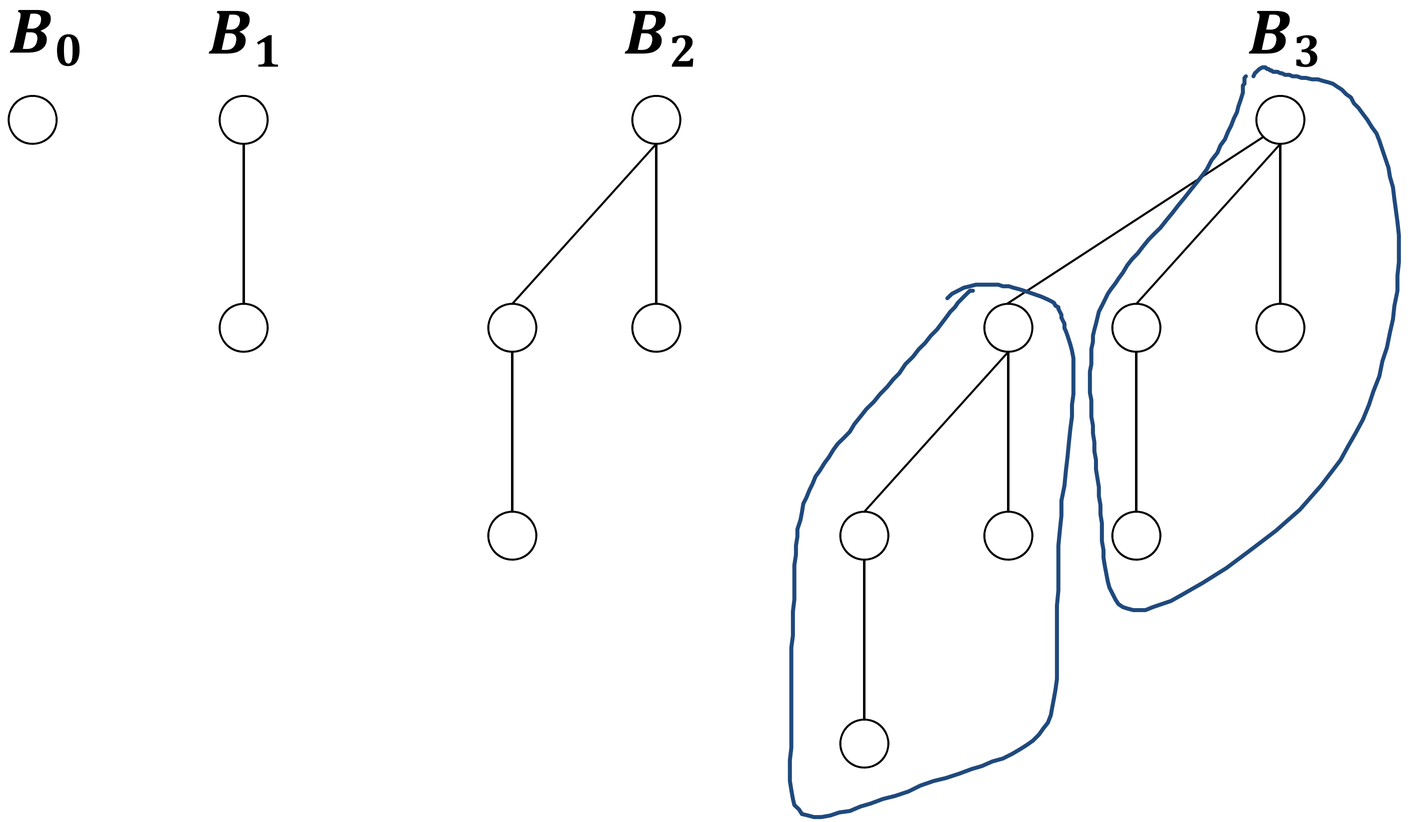
B_1



B_2



Binomial Trees

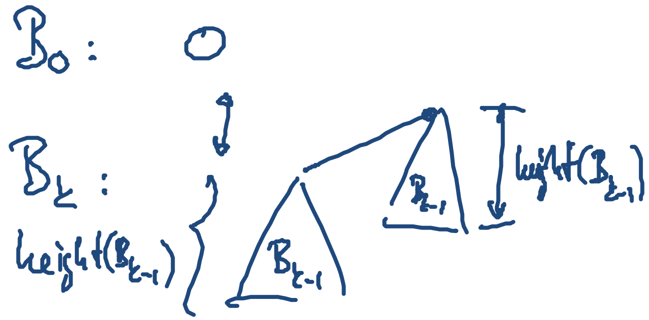


Properties

1. Tree B_k has 2^k nodes

$|B_0| = 1 = 2^0$

$|B_k| = 2 \cdot |B_{k-1}| = 2 \cdot 2^{k-1} = 2^k$



2. Height of tree B_k is k

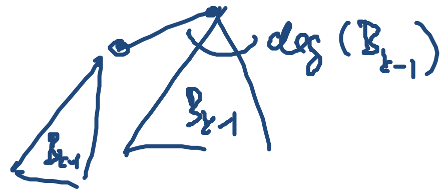
$\text{height}(B_0) = 0$

$\text{height}(B_k) = 1 + \text{height}(B_{k-1}) = 1 + (k-1) = k$

3. Root degree of B_k is k

$\deg(B_0) = 0$

$\deg(B_k) = \deg(B_{k-1}) + 1 = k$



4. In B_k, there are exactly $\binom{k}{i}$ nodes at depth i

Binomial Coefficients

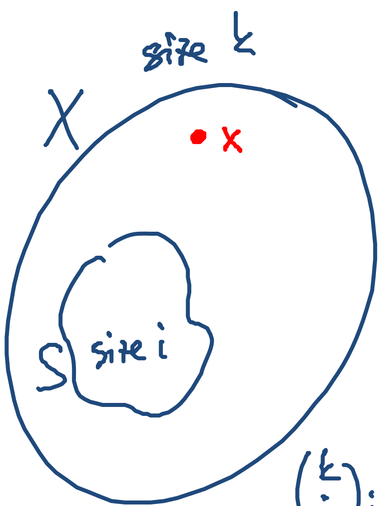
$\binom{k}{0} := 1$

- Binomial coefficient:

$$\binom{k}{i} = \frac{k!}{i!\,(k-i)!} : \text{\# of size } i \text{ subsets of a set of size } k$$

- Property: $\binom{k}{i} = \binom{k-1}{i-1} + \binom{k-1}{i}$

$k \geq 2,\ i \geq 1$



x : an elem. of X

#ways to choose S s.t.

a) S does not contain x: $\binom{k-1}{i}$
(i elem. from $X \setminus \{x\}$)

b) S contains x: $\binom{k-1}{i-1}$
($i-1$ elem. from $X \setminus \{x\}$)

$\binom{k}{i}$: # ways to choose S

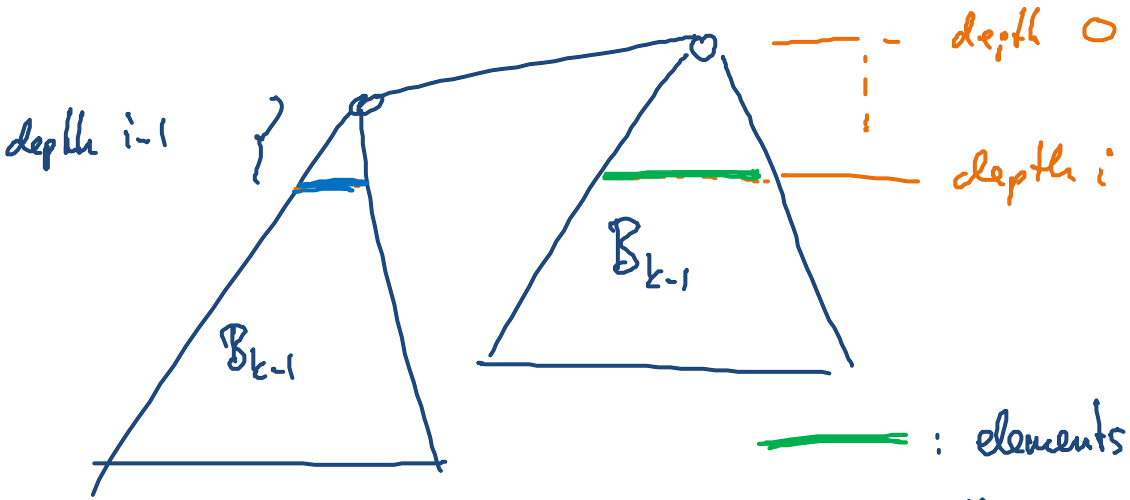
Pascal triangle:

$$\begin{array}{ccccccccccc}
 & & & & & 1 & & & & & \\
 & & & & 1 & & 1 & & & & \\
 & & & 1 & & 2 & & 1 & & & \\
 & & 1 & & 3 & & 3 & & 1 & & \\
 & 1 & & 4 & & 6 & & 4 & & 1 & \\
1 & & 5 & & 10 & & 10 & & 5 & & 1
\end{array}$$

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

Number of Nodes at Depth i in B_k

Claim: In B_k, there are exactly $\binom{k}{i}$ nodes at depth i



: elements at depth i in B_{k-1}
(by induction $\binom{k-1}{i}$)

: elements at depth $i-1$ in B_{k-1}
(by ind. $\binom{k-1}{i-1}$)

$\Longrightarrow$ # elem. at depth i in B_k:

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

Binomial Heap

$|B_k| = 2^k$

- Keys are stored in nodes of binomial trees of different order

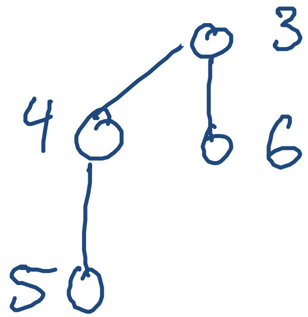
  n nodes: there is a binomial tree B_i of order i iff
  bit i of base-2 representation of n is 1.

  $n = 10 = (1010)_2$ → trees of order 1 and 3
  $\quad 3\,2\,1\,0$

  $|B_1| = 2^1 \qquad |B_3| = 2^3 = 8$

- **Min-Heap Property:**

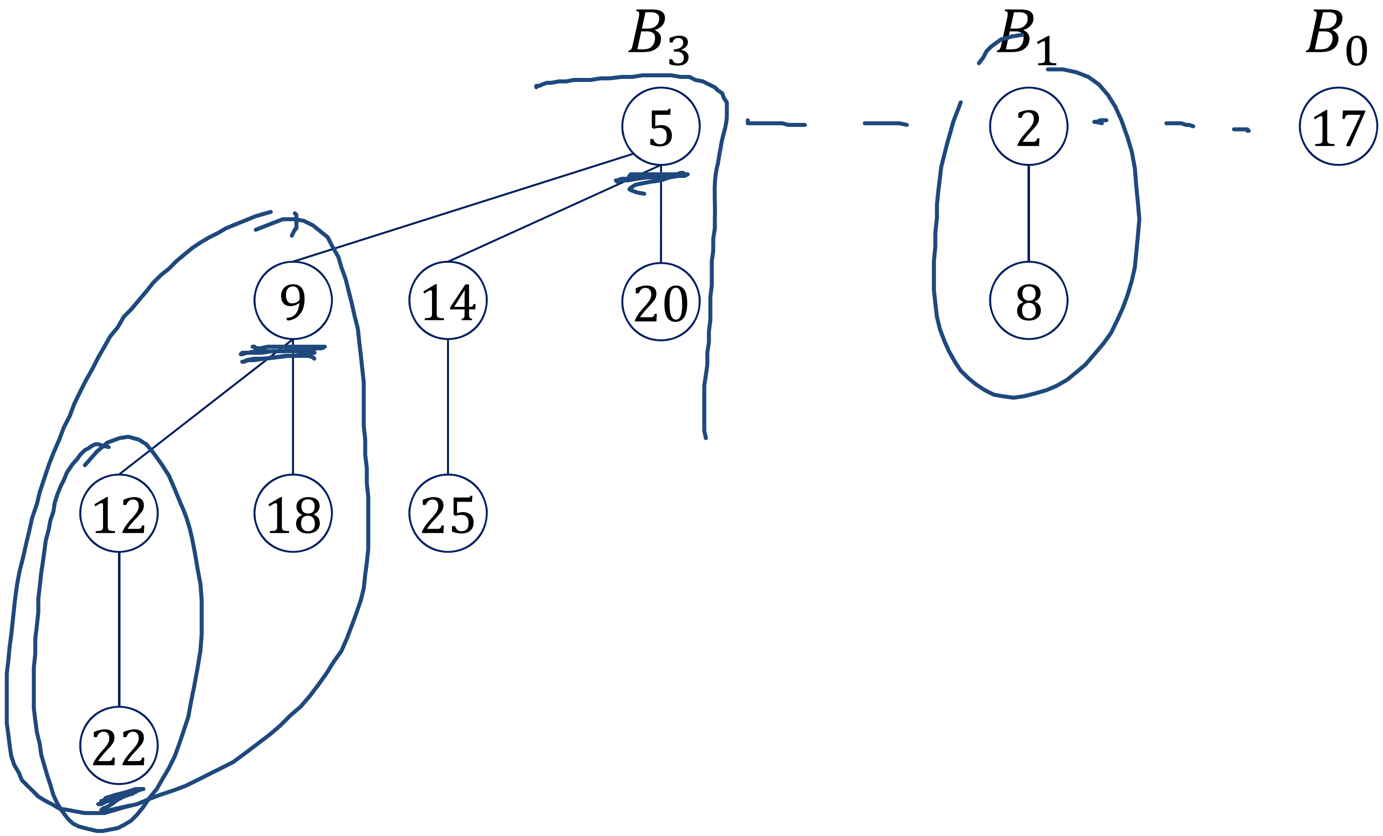
  Key of node $v \leq$ keys of all nodes in sub-tree of v



Example

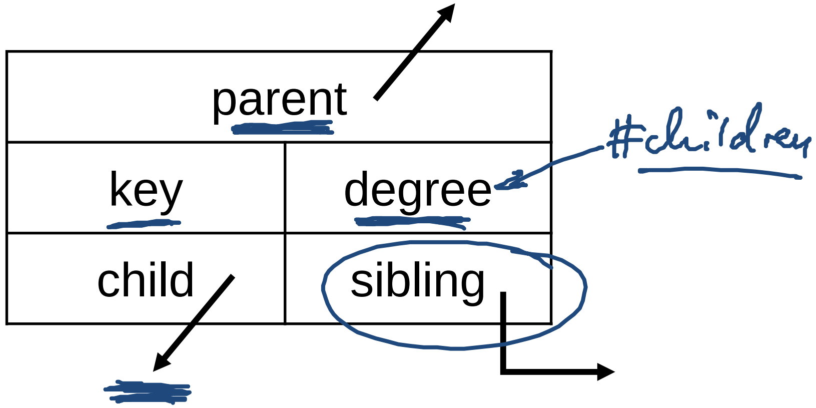
- 11 keys: $\{2, 5, 8, 9, 12, 14, 17, 18, 20, 22, 25\}$

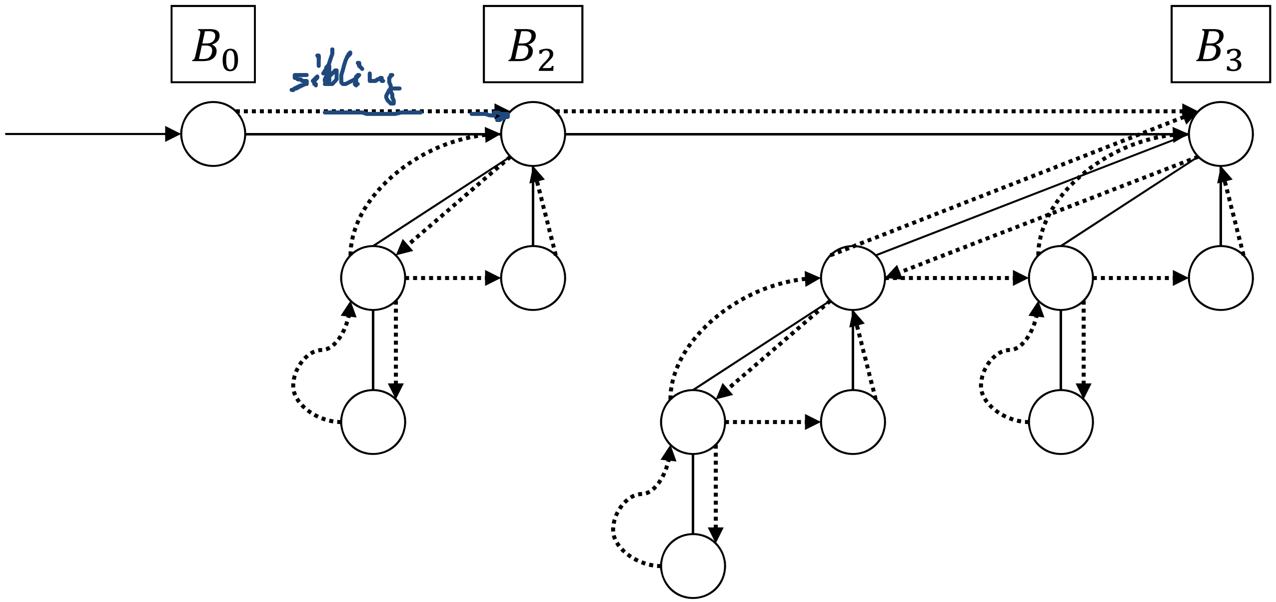
- Binary representation of n: $(11)_2 = \underset{3}{1}0\underset{1}{1}\underset{0}{1}$
  → trees B_0, B_1, and B_3 present



Child-Sibling Representation

Structure of a node:



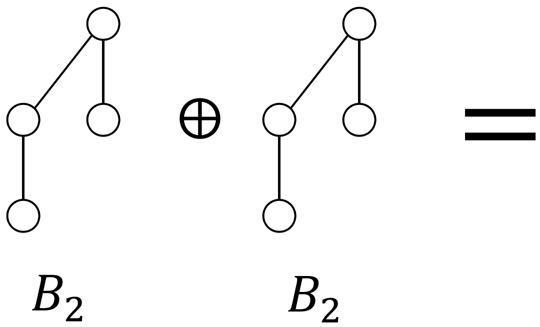


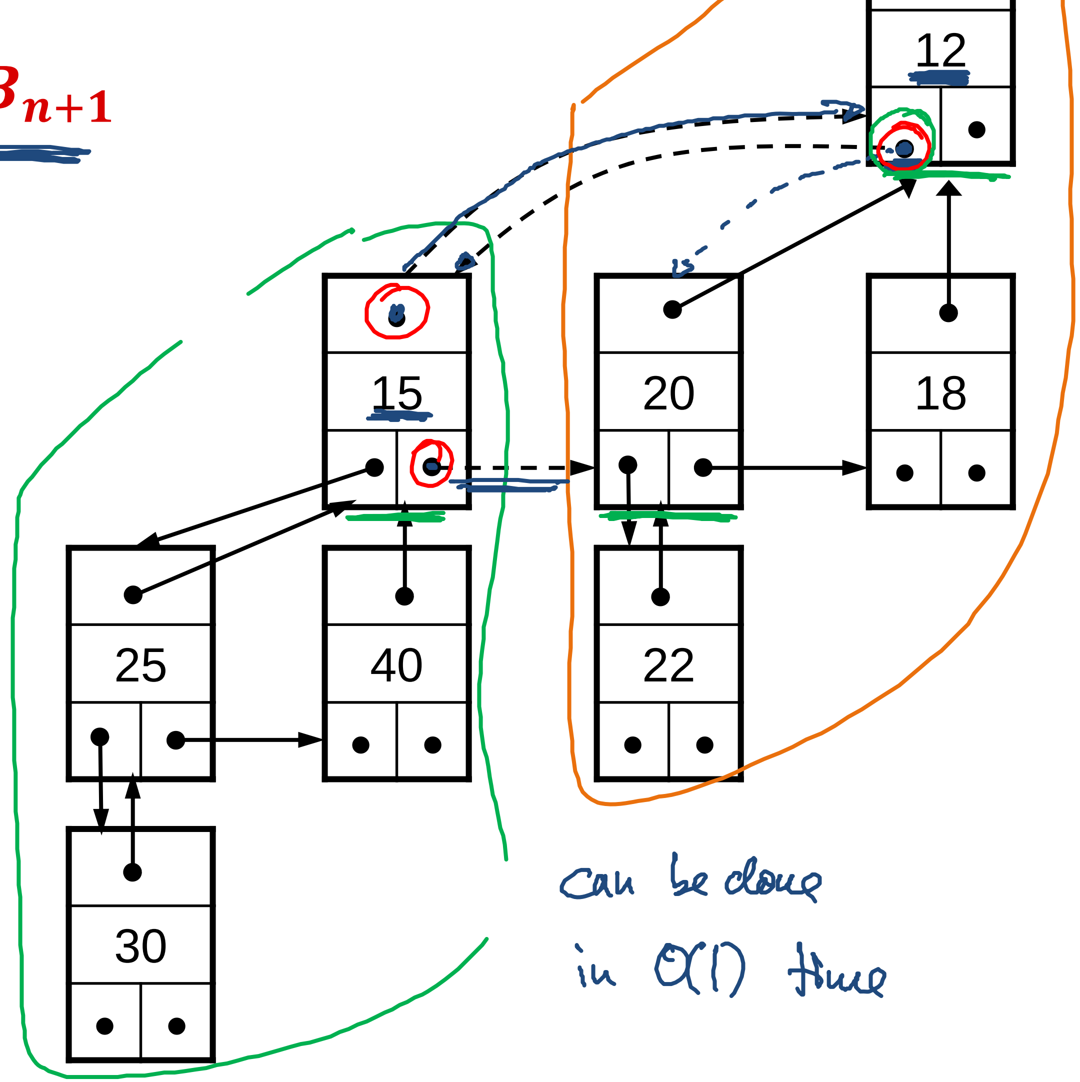
Link Operation

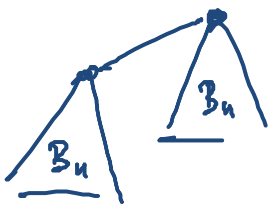
- Unite two binomial trees of the same order to one tree:

$$B_n \oplus B_n \Rightarrow B_{n+1}$$

- Time: $O(1)$





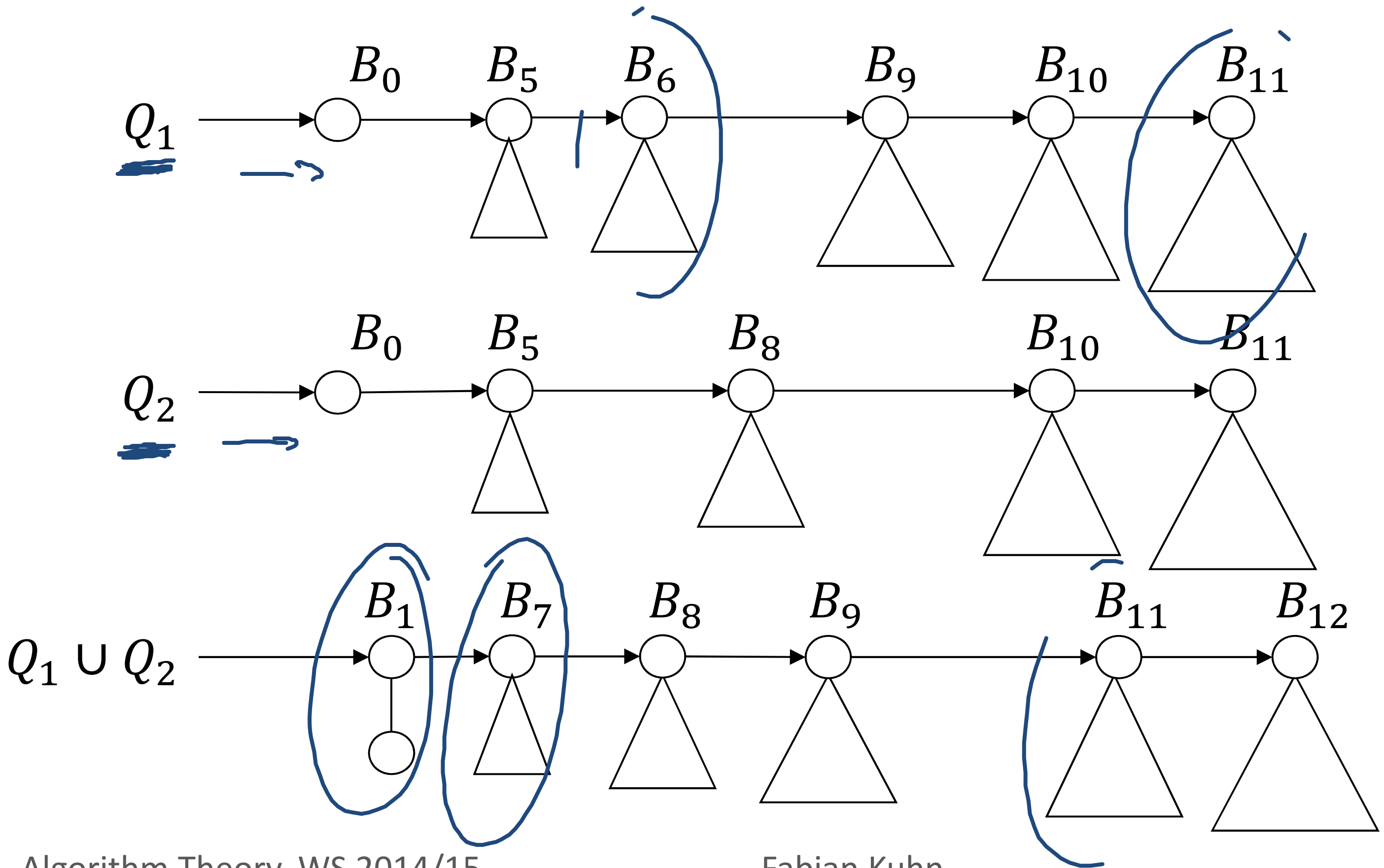


Merge Operation

111001100001
110100100001
1101110000010

Merging two binomial heaps:

- **For $\boldsymbol{i = 0, 1, \dots, \log n}$:**
  If there are 2 or 3 binomial trees B_i: apply link operation to merge 2 trees into one binomial tree B_{i+1}



Time: $\boldsymbol{O(\log n)}$