Datastructuren

Data Structures

Hendrik Jan Hoogeboom

Informatica – LIACS
Universiteit Leiden

najaar 2023

Table of Contents I

Contents

Contents

Set vs Dictionary

	{ key }	{ (key, value) }
	Set	Dictionary / Map
isEmpty	$A = \varnothing$?	
size	$\lvert A \rvert$	
isElement	$a \in A$?	$a \mapsto v(a)$
insert	$A \cup \{a\}$	(a, v)
delete	$A \setminus \{a\}$	a

ADT Set

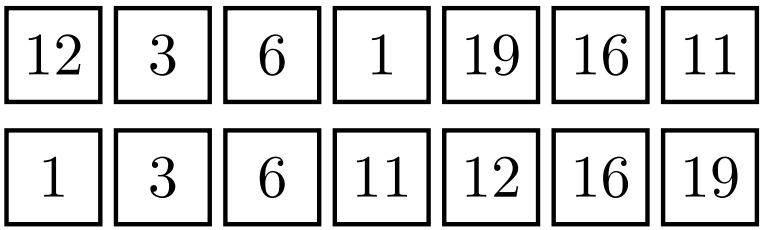
- INITIALIZE: construct an empty set, return $A = \varnothing$.
- ISEMPTY: check whether there the set is empty ($A \stackrel{?}{=} \varnothing$, contains no elements).
- SIZE: return the number of elements, the cardinality $|A|$.
- ISELEMENT(a): returns whether a given object from the domain belongs to the set, $a \stackrel{?}{\in} A$.
- INSERT(a): add an element to the set, return $A \cup \{a\}$ (assume if it is not present?)
- DELETE(a): removes an element from the set, return $A \setminus \{a\}$ (assume it is present?).

ADT Dictionary / Map / Associative array

stores (*key*,*value*) pairs retrieval based on *key*

- INITIALIZE: construct an empty Map
- ISEMPTY: check whether there the Map is empty
- SIZE: return the number of elements
- RETRIEVE(*key*): returns *value* for (*key*,*value*) in Map
  (and signals when no such element present)
- INSERT(*key*,*value*): inserts the pair
  (what if another value for *key* is present?)
- DELETE(*key*): removes element with given *key*

list representations (of Set)

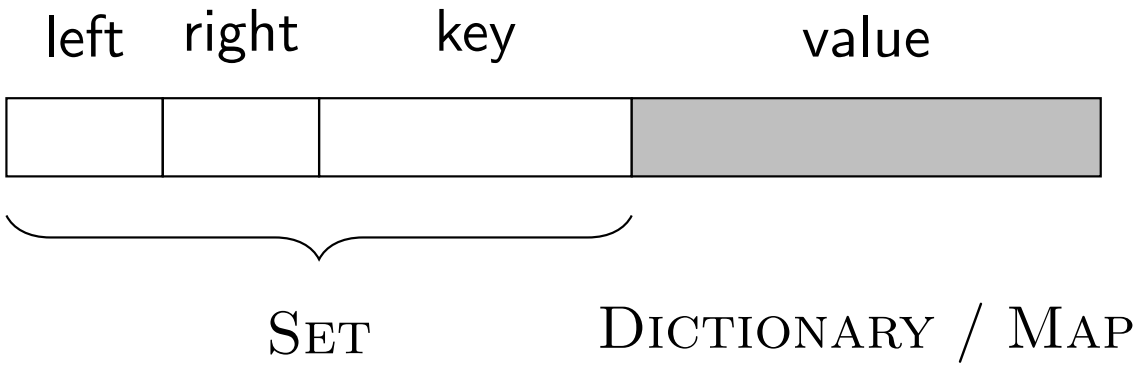


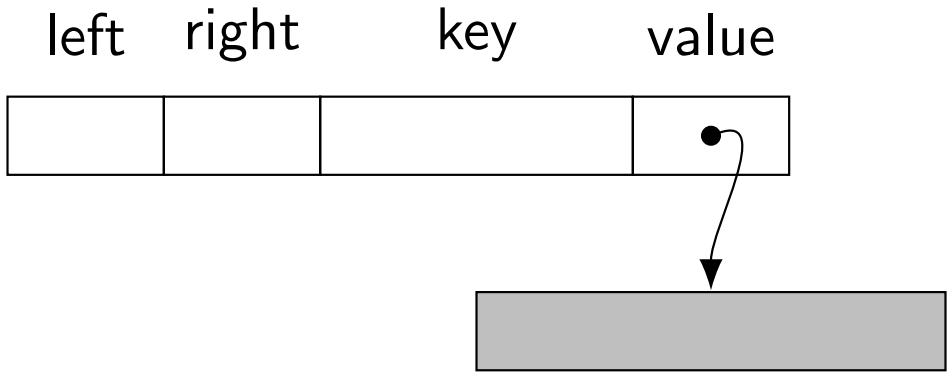
(unsorted) (array/linked)

(sorted)

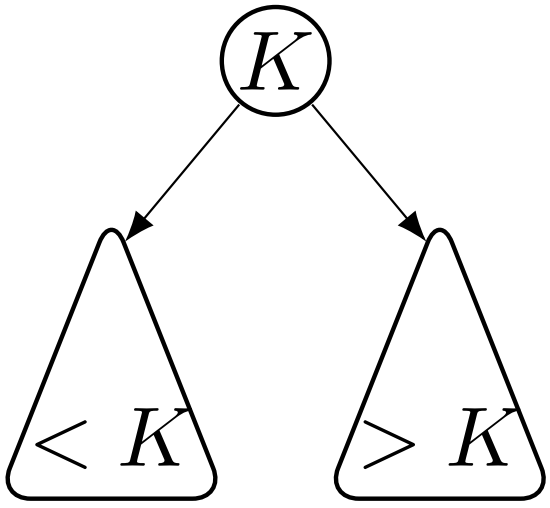
		$x \in S$	$S \cup \{x\}$	$S \setminus \{x\}$
array	unsorted	$\mathcal{O}(n)$	$\mathcal{O}(n) + \mathcal{O}(1)$	$\mathcal{O}(n) + \mathcal{O}(1)$
	sorted	$\mathcal{O}(\lg n)$	$\mathcal{O}(\lg n) + \mathcal{O}(n)$	$\mathcal{O}(n)$
linked	unsorted	$\mathcal{O}(n)$	$\mathcal{O}(n) + \mathcal{O}(1)$	$\mathcal{O}(n) + \mathcal{O}(1)$
	sorted	$\mathcal{O}(n)$	$\mathcal{O}(n) + \mathcal{O}(1)$	$\mathcal{O}(n) + \mathcal{O}(1)$
			locate+adapt	

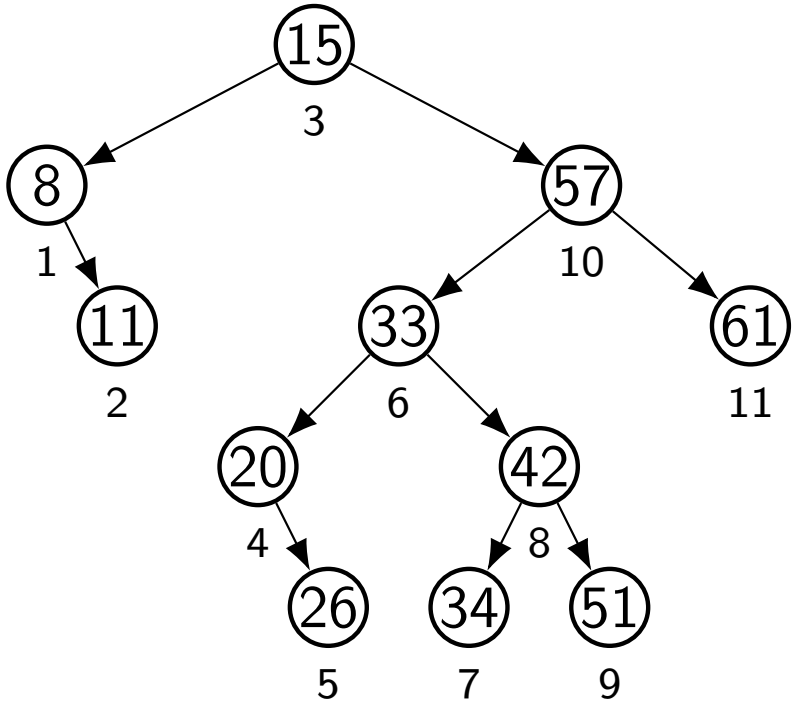
binary tree Node





binary search tree



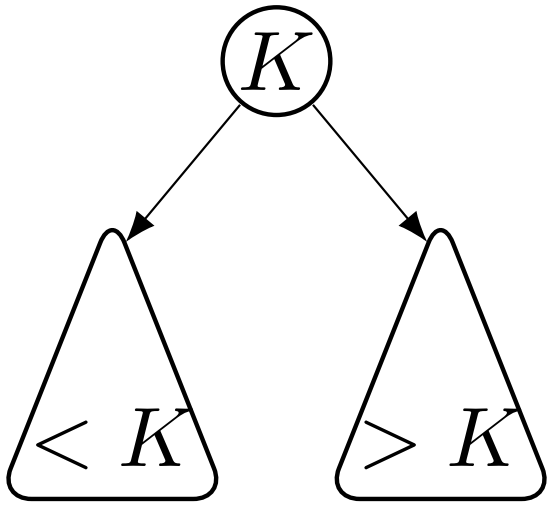


inorder: 8 11 15 29 26 33 34 42 51 57 61

BST and inorder

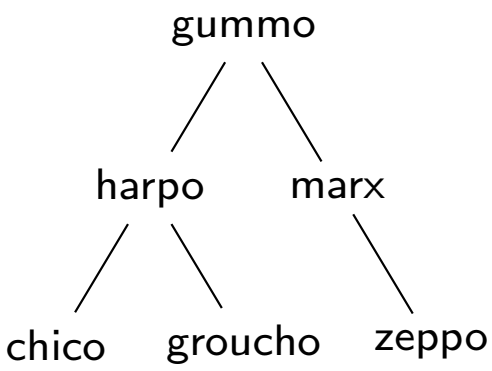
Lemma

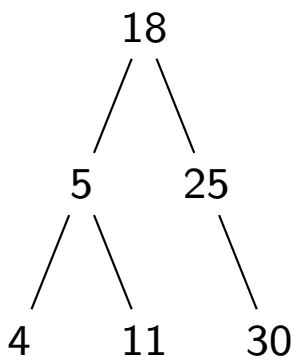
Let T be a binary search tree. Then the inorder traversal of T visits all the nodes in increasing order.

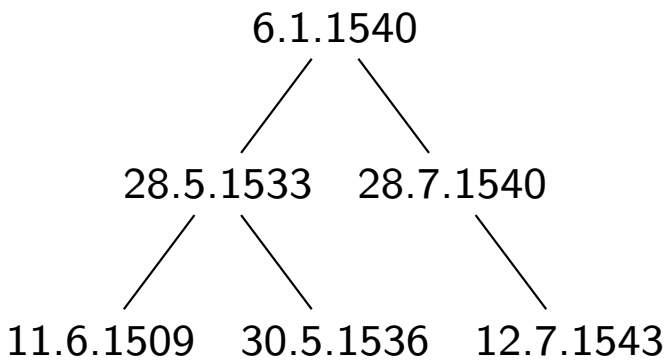


comparables

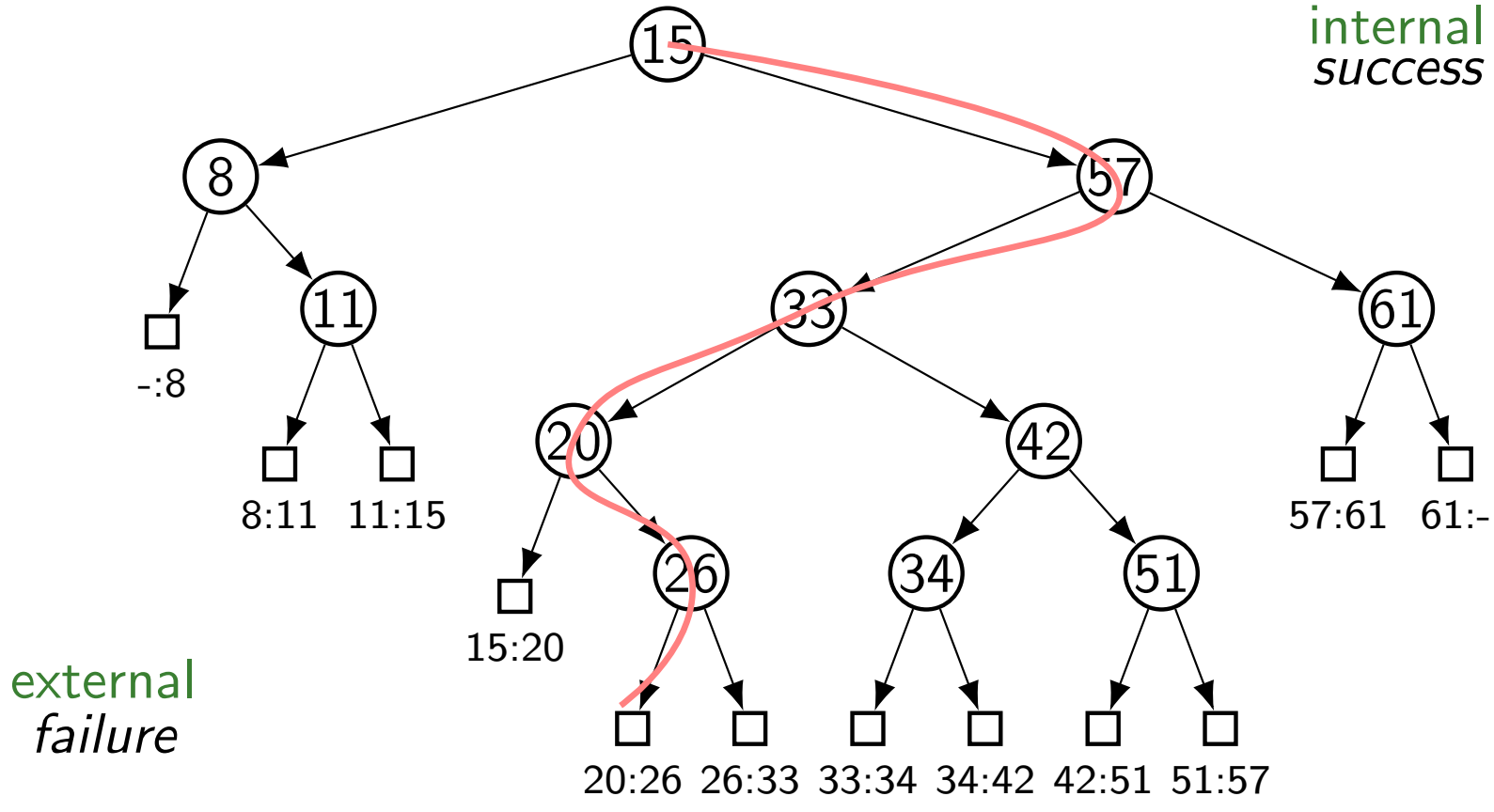
linear order on domain (keys) gummo $<$ zeppo $18 < 30$



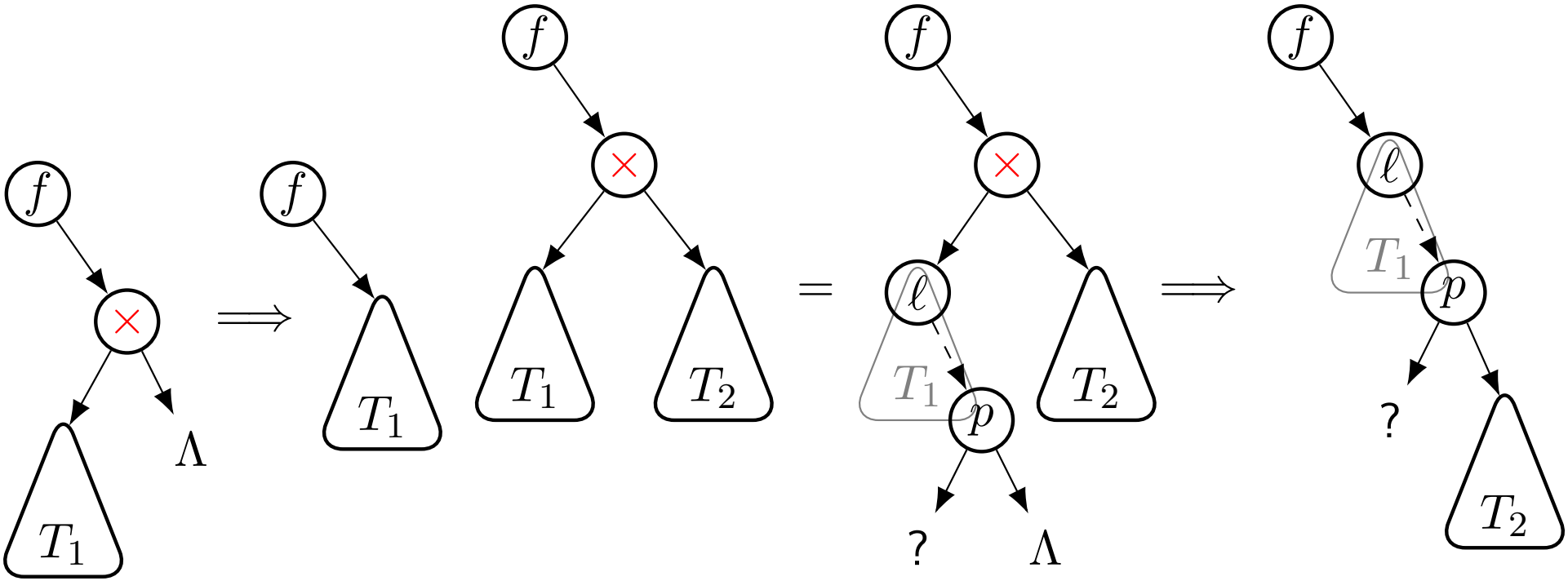




find, insert


15
8
57
-:8
11
33
61
8:11
11:15
20
42
57:61
61:-
15:20
26
34
51
20:26
26:33
33:34
34:42
42:51
51:57
internal
success
external
failure

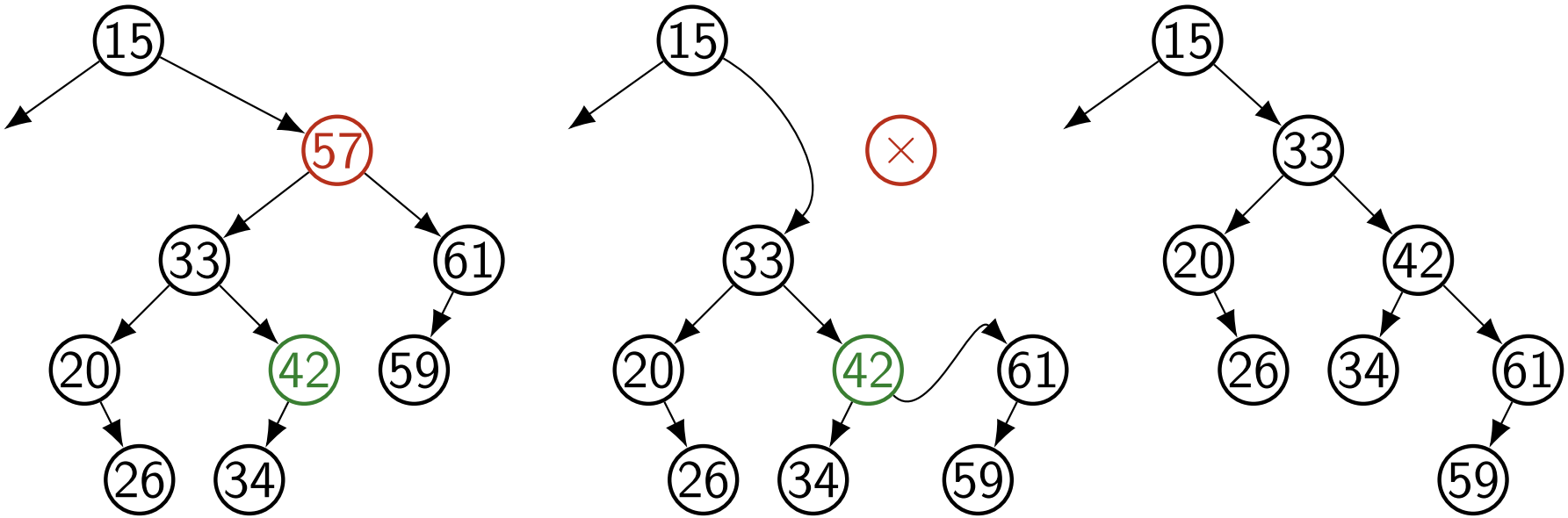
deletion 'by merging'



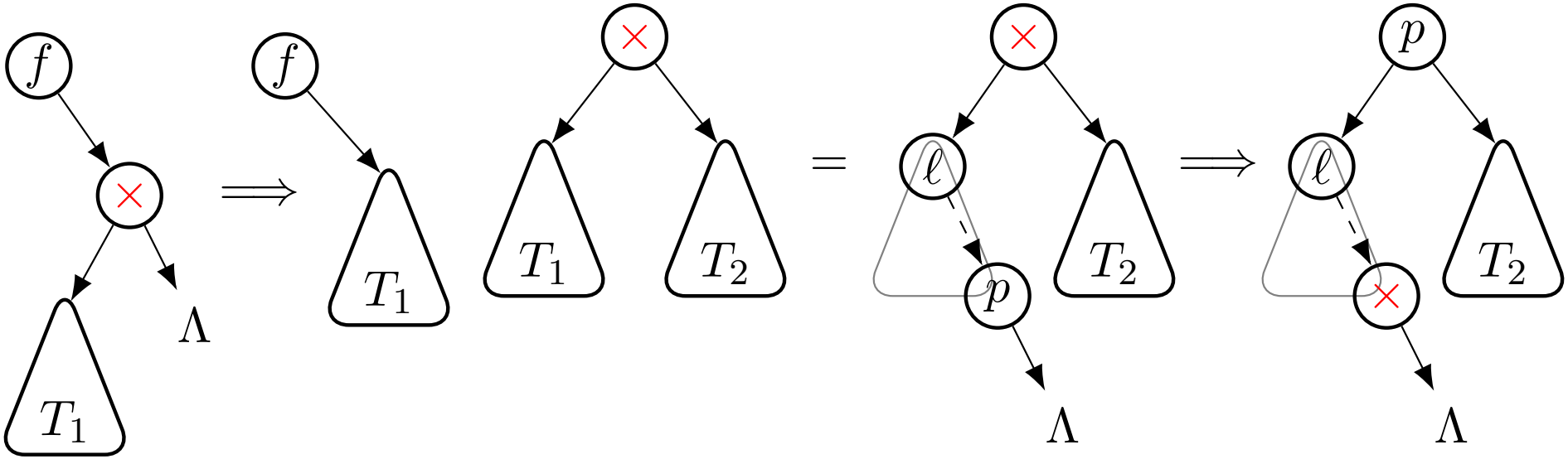
simple case: no [right] child otherwise link right child under predecessor

example: deletion by merging

deleting 57 predecessor 42



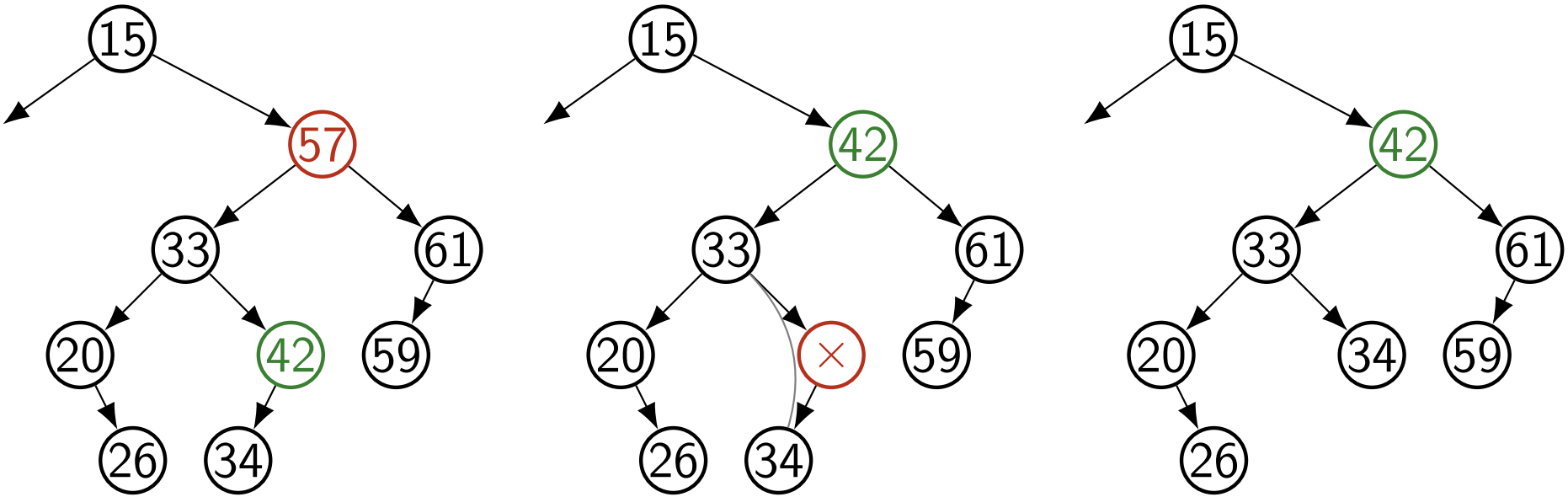
deletion 'by copying'



simple case: no [right] child otherwise copy predecessor

example: deletion by copying

deleting 57 predecessor 42



Contents

search value/key

```
bool contains( const Comparable & x, Node *t ) const {
    if( t == nullptr )
        return false;
    else if( x < t->element )
        return contains( x, t->left );
    else if( t->element < x )
        return contains( x, t->right );
    else
        return true; // found
}
```

call with: `contains(v,root);`

find min/max value

```
BinaryNode * findMin( BinaryNode *t ) const {
    if( t == nullptr )
        return nullptr;
    if( t->left == nullptr )
        return t;
    return findMin( t->left );
}
BinaryNode * findMax( BinaryNode *t ) const {
    if( t != nullptr )
        while( t->right != nullptr )
            t = t->right;
    return t;
}
```

call with: `findMin(root);` and `findMax(root);`

insertion (recursive)

```
template<class T>
void Node<T>::insert(const T& el, Node<T> * & p) {
    if( p == nullptr ) {
        p = new Node{el, nullptr, nullptr};
    } else if (el < p->data) {
        insert(el, p->left);
    } else if (el > p->data) {
        insert(el, p->right);
    } else {
        ; // Duplicate; do nothing
    }
}
```

call with: `insert(el,root);`

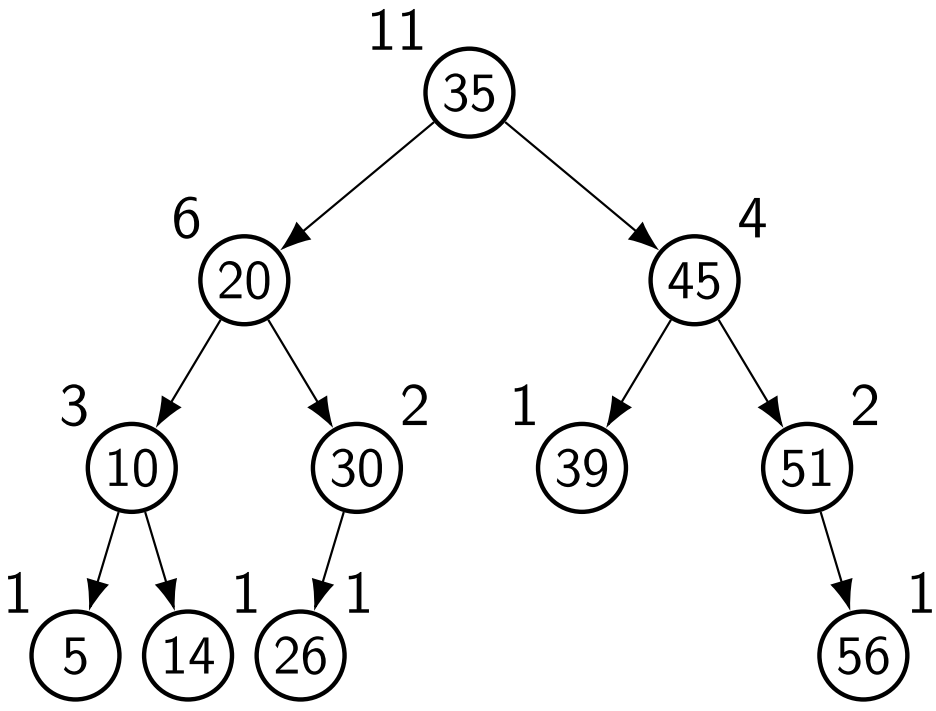
deletion (recursive)

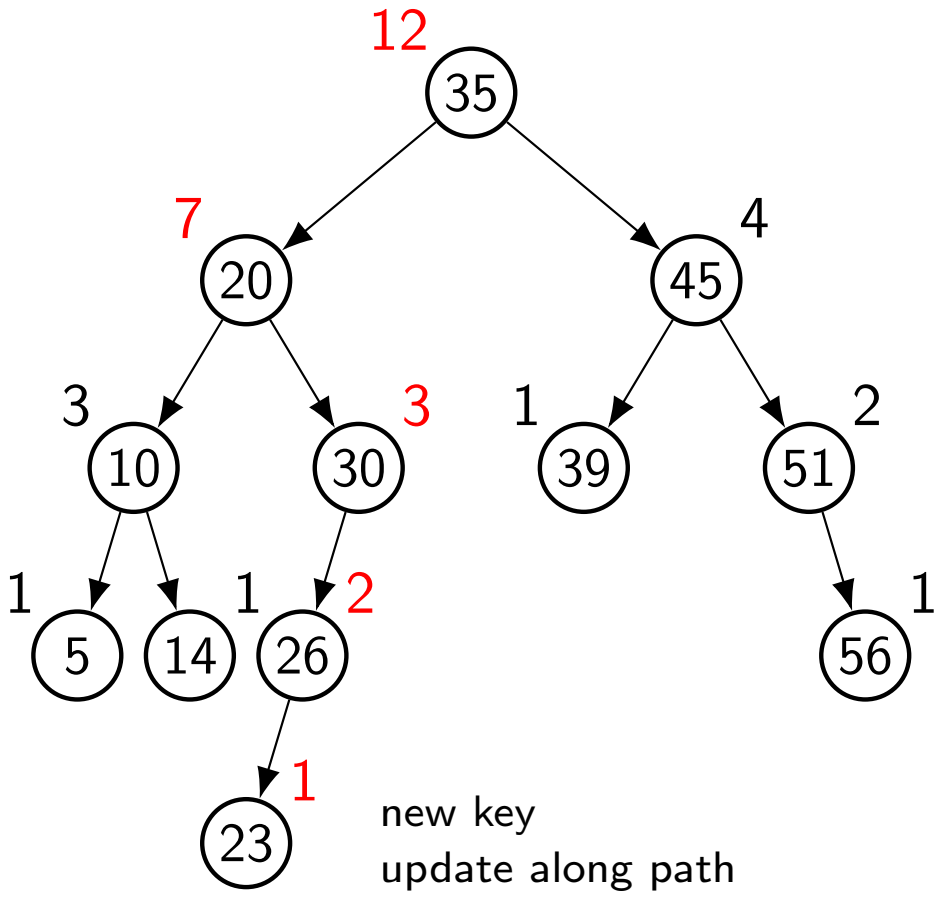
```
void remove( const Comparable & x, Node * & t ) {
    if( t == nullptr )     return;
    if( x < t->data )      remove( x, t->left );
    else if( x > t->data)  remove( x, t->right );
    else if( t->left != nullptr && t->right != nullptr ) {
        Node *pred = findMax( t->left );
        t->element = pred->element;   \\ copying
        remove( t->element, t->left );
    }
    else {       \\ no two children
        BinaryNode *oldNode = t;
        if(t->left != nullptr ) t = t->left
        else                    t = t->right;
        delete oldNode;
    }
}
```

aanroepen met: `remove(el,root);`

Contents

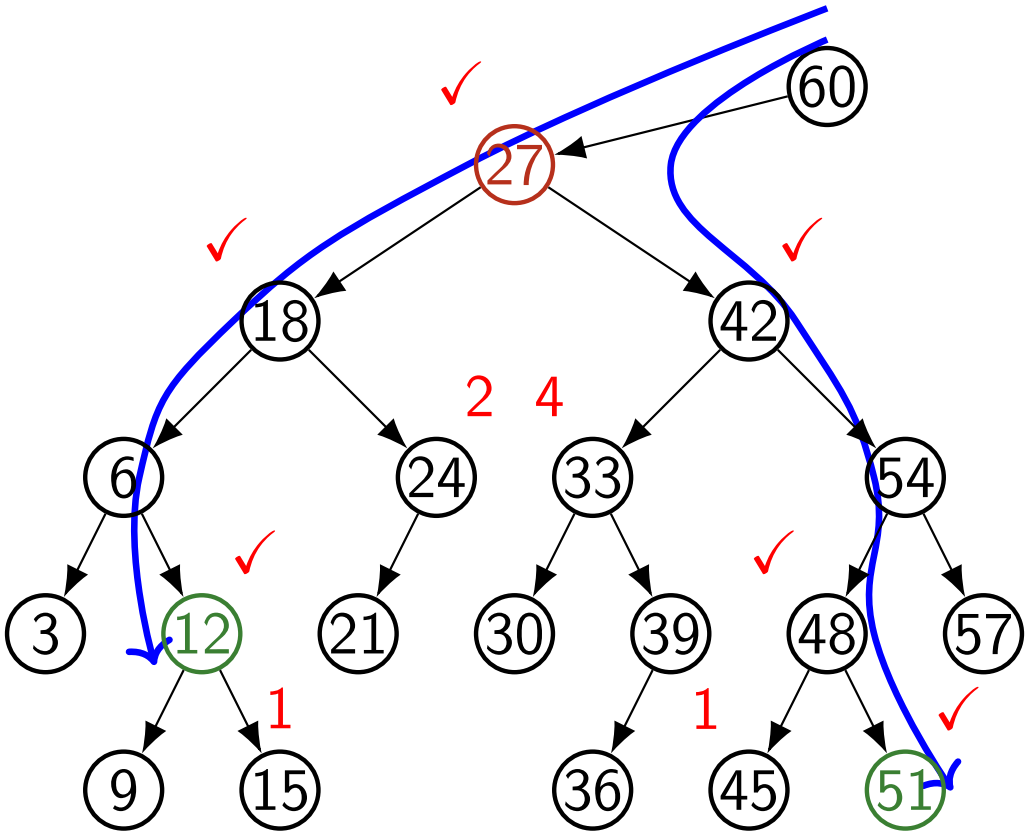
order statistics: find k-th element





augment each node with the size of its subtree

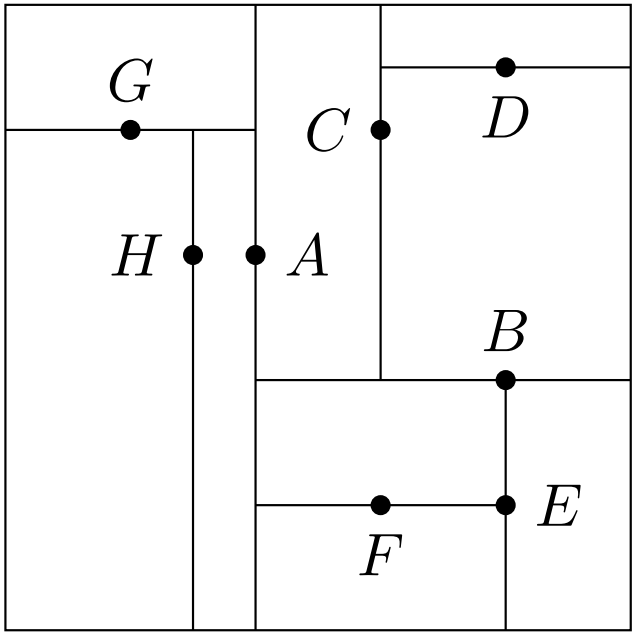
range search [12,52]

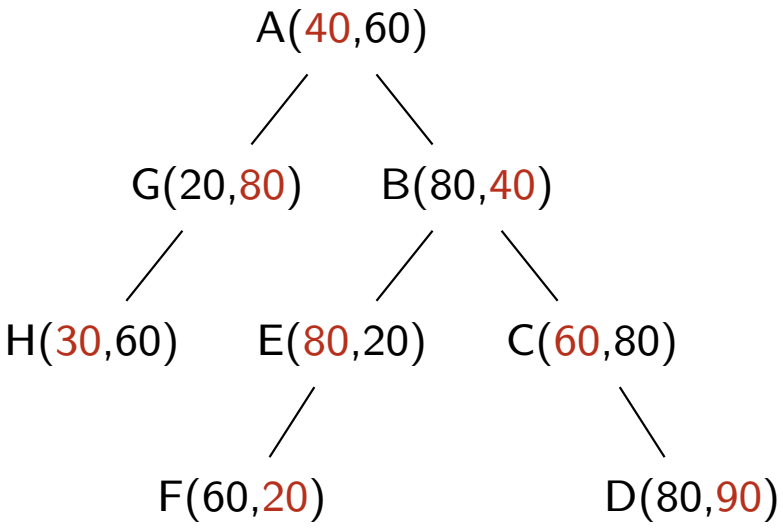


start counting where paths diverge

count trees ‘inside’ + nodes along path

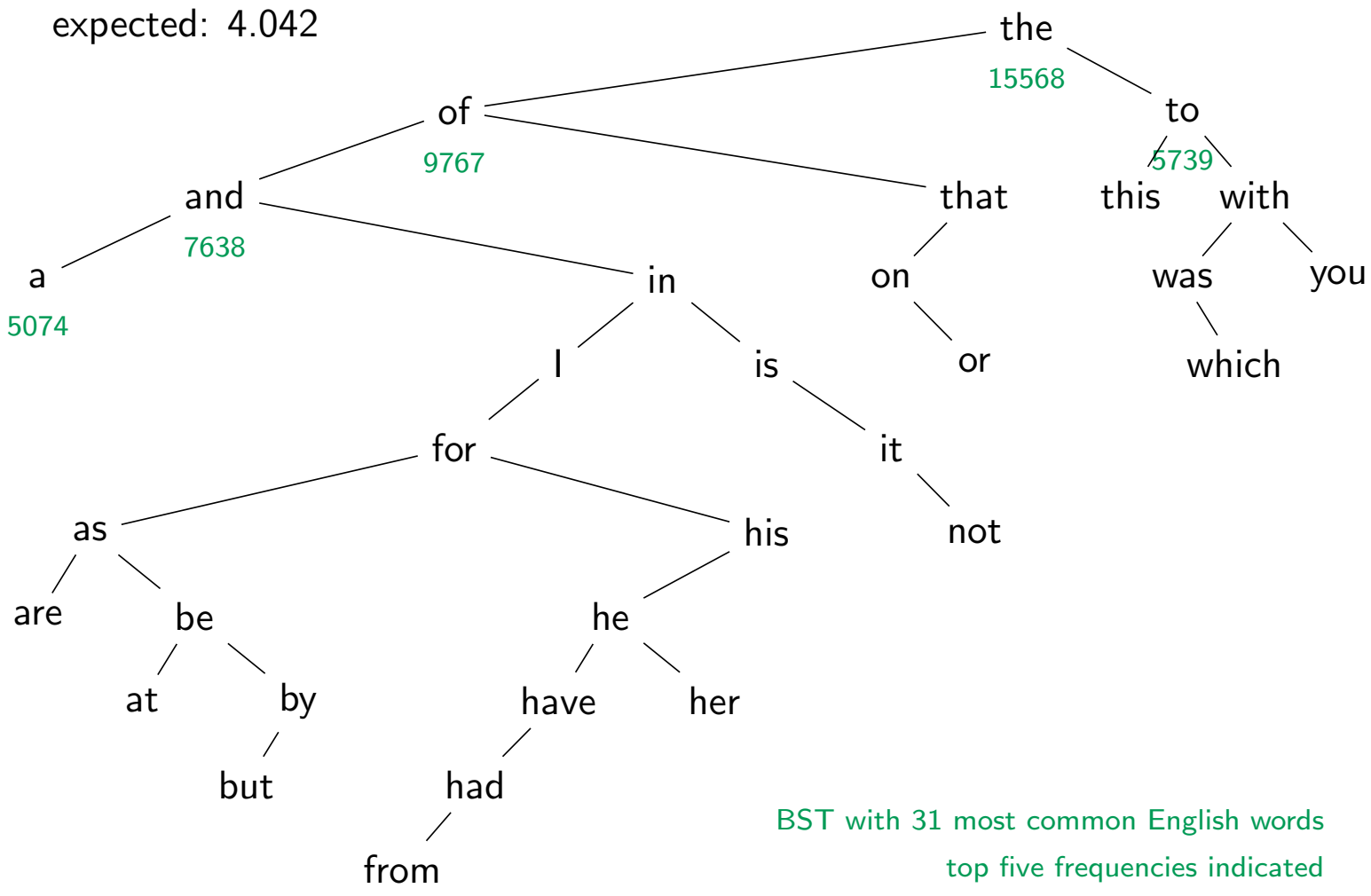
k-d-tree ☒





Contents

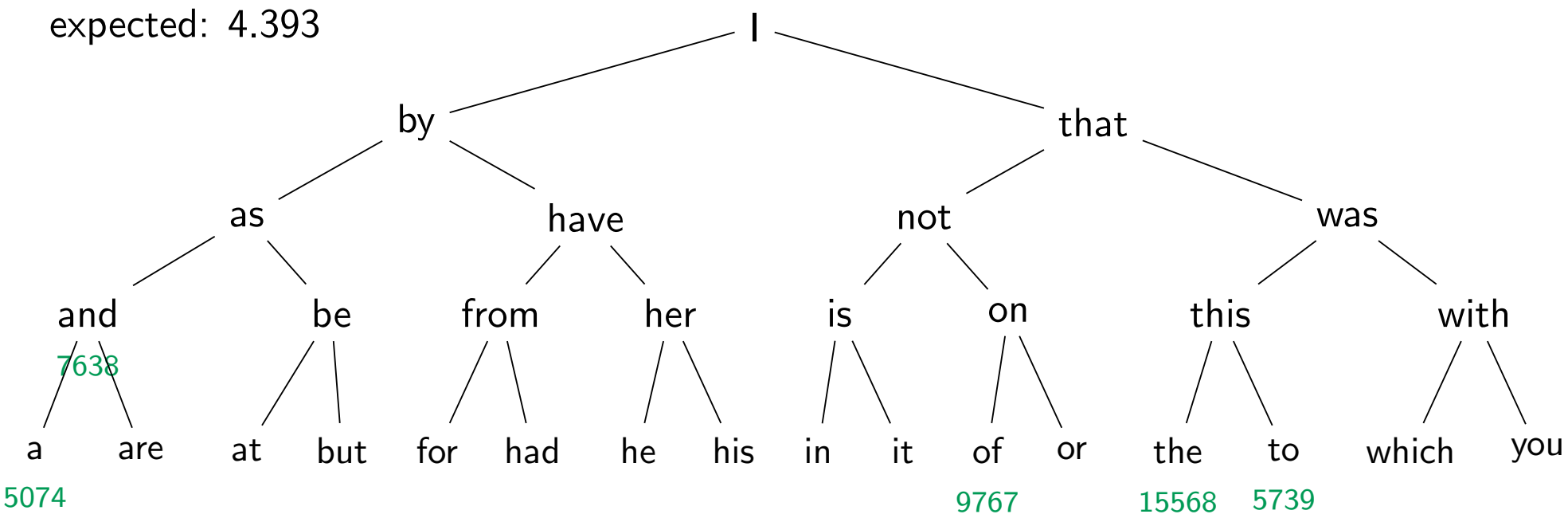
order of frequency



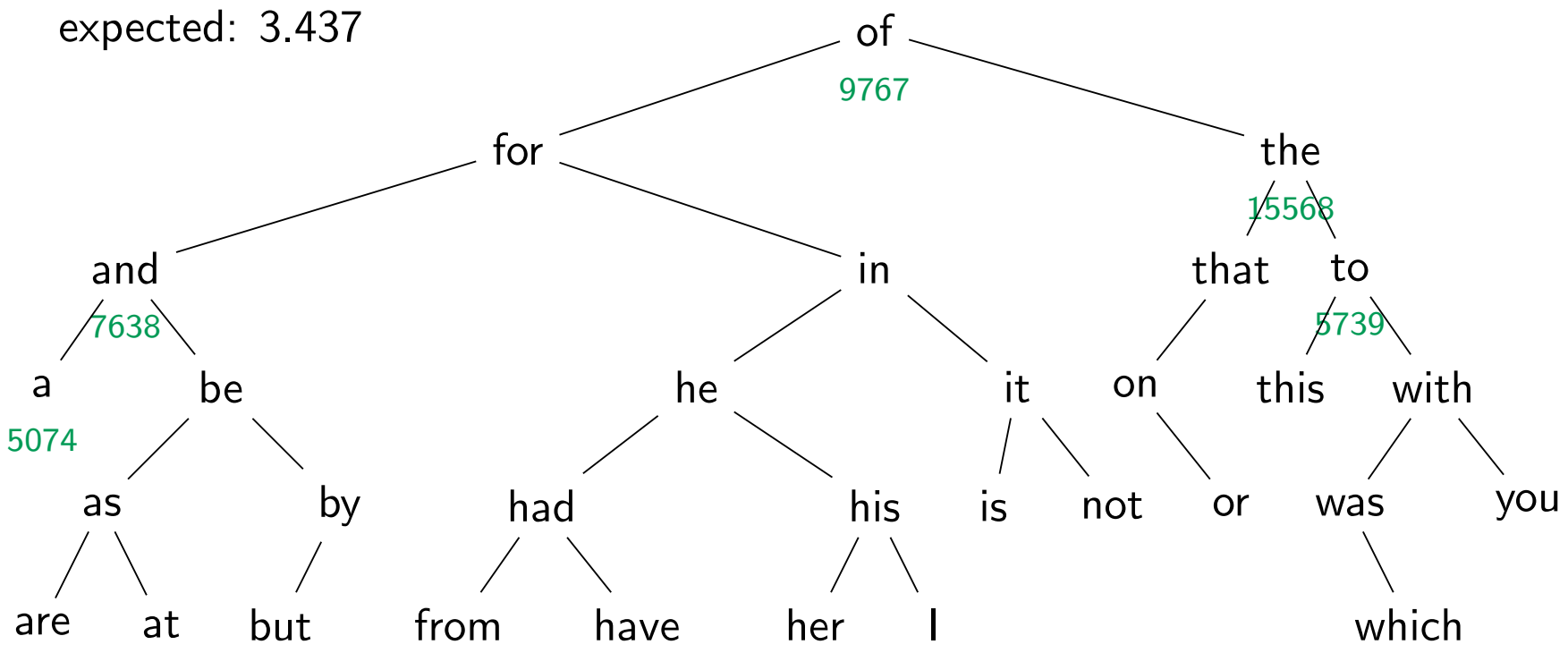
BST with 31 most common English words

top five frequencies indicated

perfectly balanced

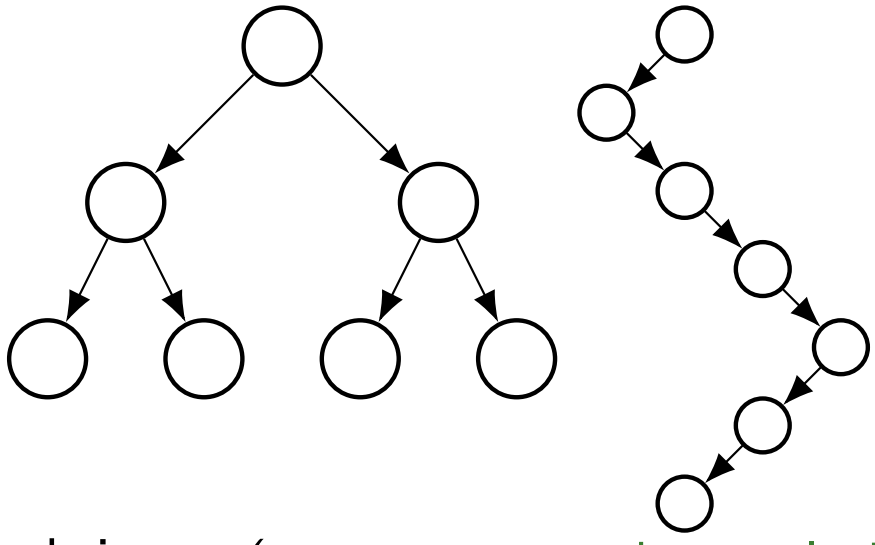


optimal



source: [Knuth] TAoCP Vol.3 (Sorting and Searching)

binary search tree BST



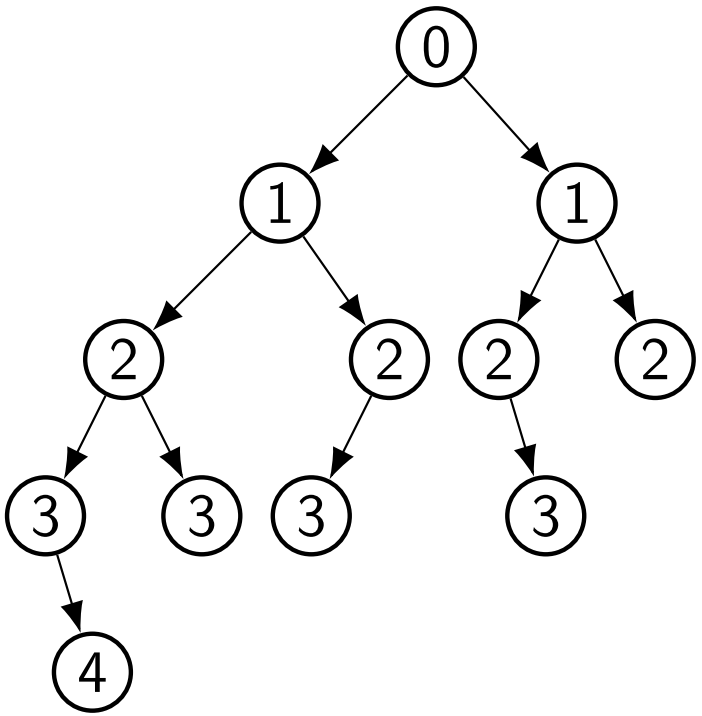
unsuccessful search in: (worst case search complexity)

- linear tree $O(n)$
- optimal tree $O(\lg(n))$ (complete tree)

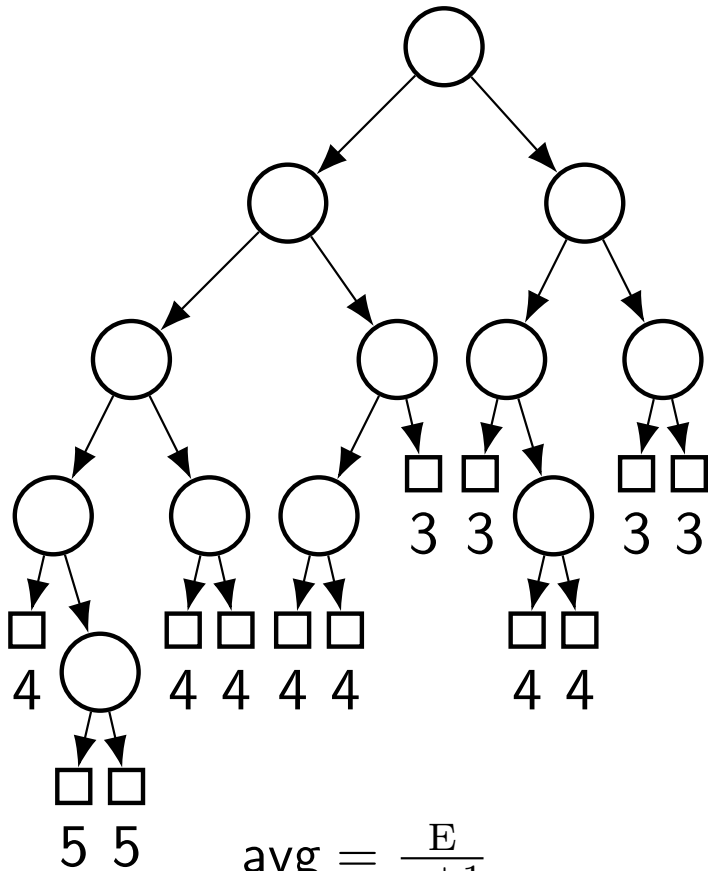
proof, and average case behaviour: see later

internal / external path length

successful search



failure



$$\text{avg} = \frac{\text{I}}{n} + 1 = \frac{\text{I}+n}{n}$$

$$\text{I} = 0 + 2{*}1 + 4{*}2 + 4{*}3 + 1{*}4 = 26$$

$$\text{avg} = \frac{\text{E}}{n+1}$$

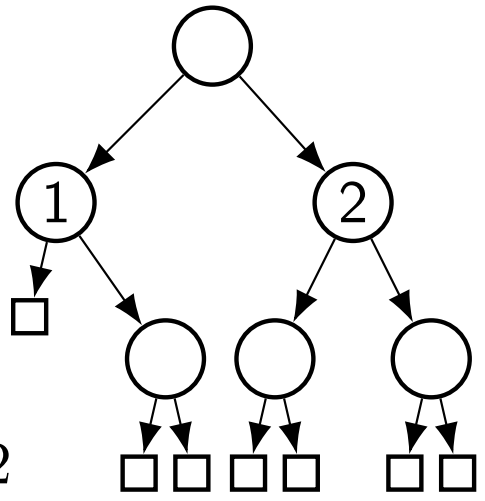
$$\text{E} = 4{*}3 + 7{*}4 + 2{*}5 = 50$$

relation I and E

Lemma

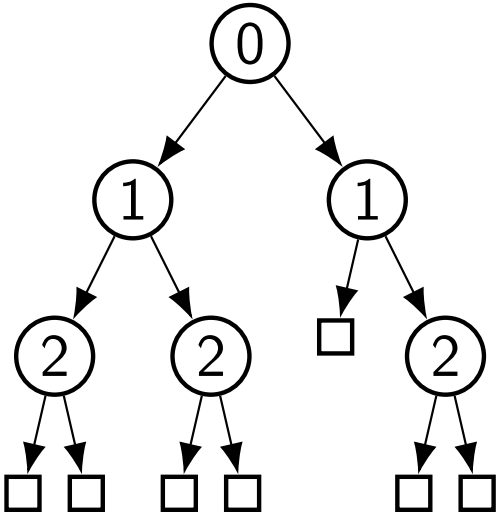
Let T be a binary tree with n nodes, with internal path length I and external path length E. Then $\mathrm{E} = \mathrm{I} + 2n$.

induction:



$$n = n_1 + n_2 + 1$$
$$I = I_1 + I_2 + n_1 + n_2$$
$$E = E_1 + E_2 + (n_1{+}1) + (n_2{+}1)$$
$$= (I_1 + 2n_1) + (I_2 + 2n_2) + n_1 + n_2 + 2$$
$$= (\underbrace{I_1 + I_2 + n_1 + n_2}_{I}) + (\underbrace{2n_1 + 2n_2 + 2}_{2n})$$

internal/external path length



path length: count edges

n (internal) nodes success

ipl $\mathrm{I} = 0+1+1+2+2+2 = 8$

$n+1$ (external) leaves failure

epl $\mathrm{E} = 3+3+3+3+2+3+3 = 20$

Definition (internal/external path length)

ipl I = sum of path lengths to all nodes

epl E = sum of path lengths to the 'extended' leaves

avg # comparisons *success*: $\frac{\mathrm{I}}{n}+1$ *failure*: $\frac{\mathrm{E}}{n+1}$

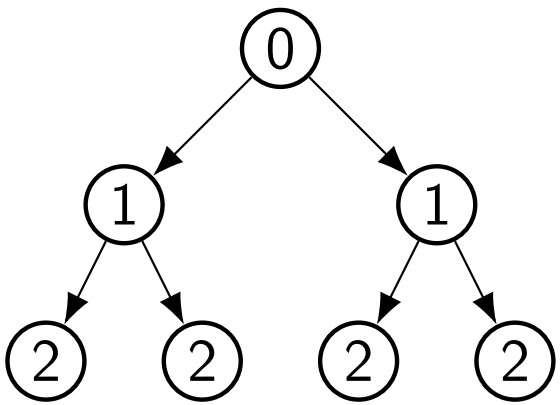
relation ipl/epl: $\mathrm{E} = \mathrm{I} + 2n$

extremal trees average per key

balanced (optimal)

h levels: $n = 2^h - 1$ nodes

$h = \lg(n+1)$

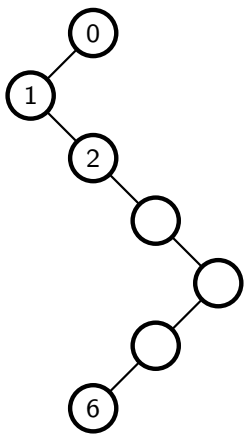


$\mathrm{I}_n = \sum_{i=0}^{h-1} i \cdot 2^i$, $\mathrm{E}_n = 2^h \cdot h$

$\Rightarrow \mathrm{I}_n = (n+1)\lg(n+1) - 2n$

avg $= \frac{n+1}{n}\lg(n+1) - 1 \sim \lg n$

linear (worst case)

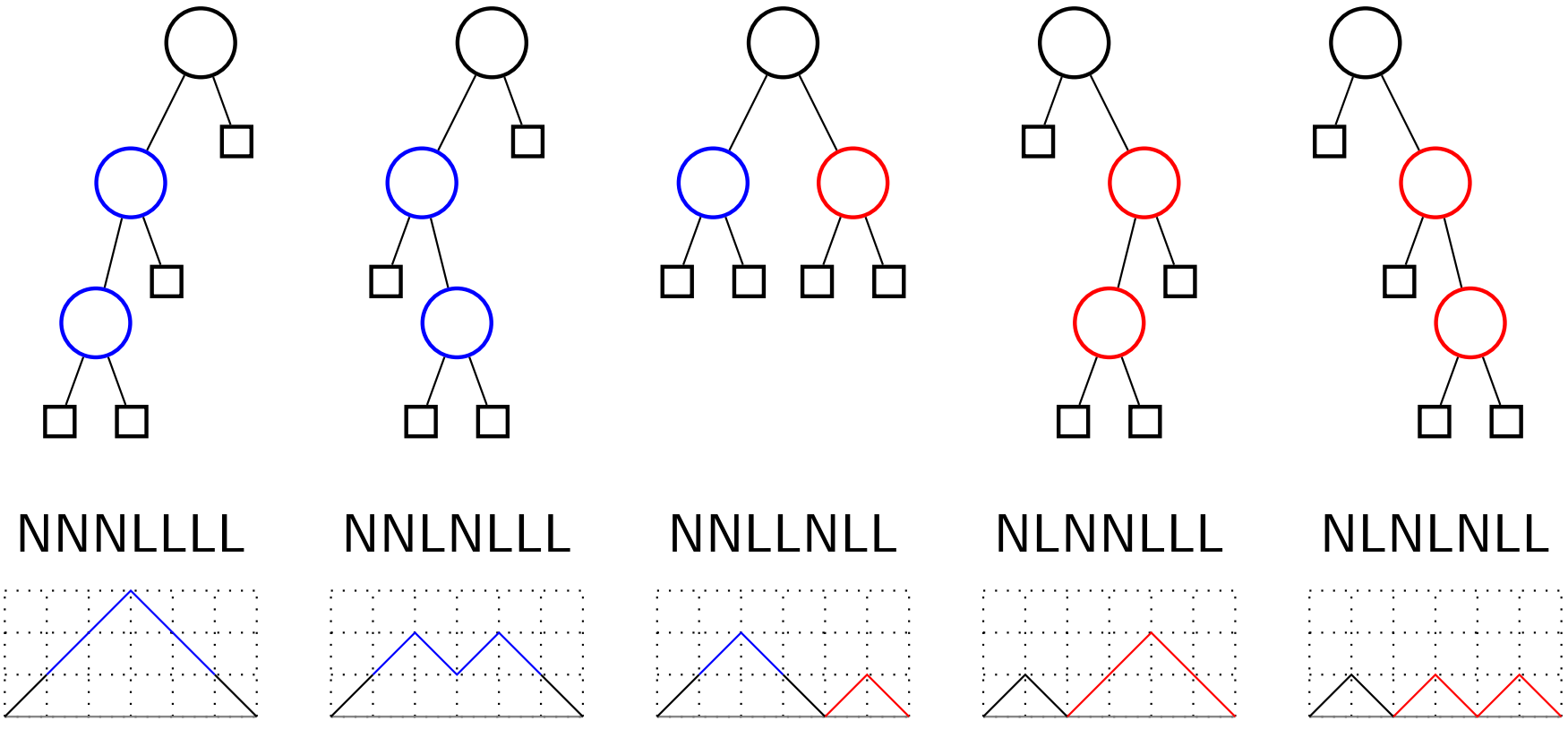


$\mathrm{I}_n = \sum_{i=0}^{n-1} i = \frac{n(n-1)}{2}$

$\mathrm{E}_n = \mathrm{I}_n + 2n = \frac{n(n+3)}{2}$

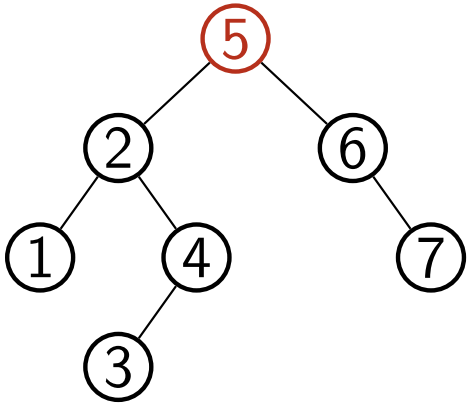
avg $= \frac{n+1}{2}$

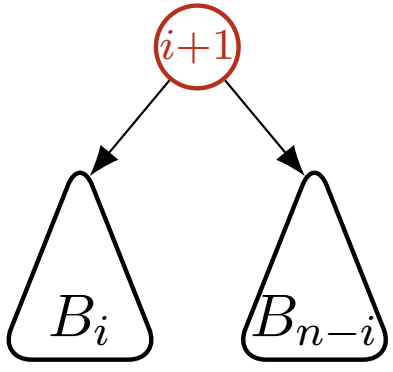
counting trees ☒



counting trees

$1, 1, 2, 5, 14, 42, 132, 429, 1430, 4862, 16796, 58786, \ldots$





1	1	2	5	14
14	5	2	1	1

(unlabeled) n-node binary trees

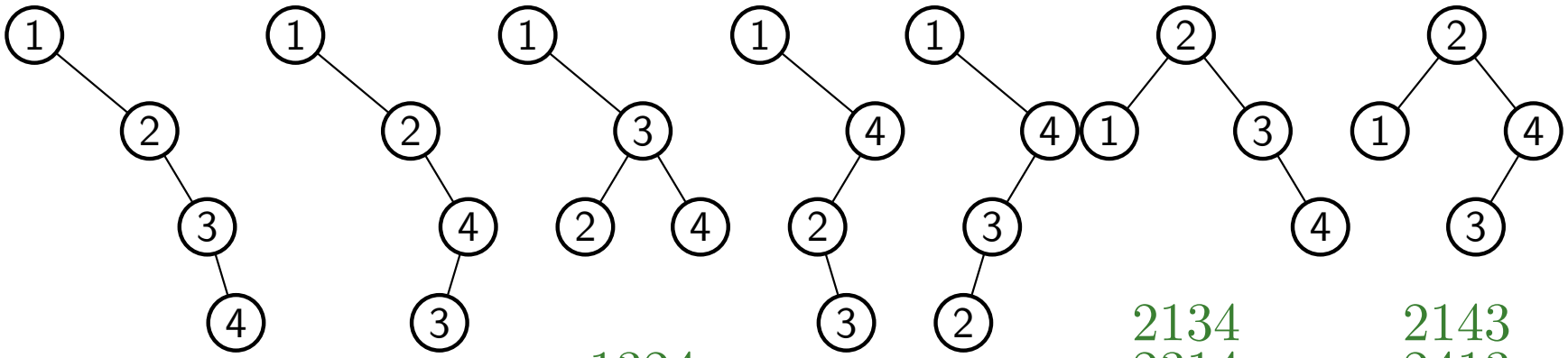
$B_{n+1} = \sum_{i=0}^{n}(B_i \cdot B_{n-i})$ with $B_0 = 1$

$B_n = \frac{1}{n+1}\binom{2n}{n} = \frac{(2n)!}{(n+1)!n!} \sim \frac{4^n}{n^{3/2}\sqrt{\pi}}$ Catalan numbers

also the number of BST with given values

trees and permutations

'*average tree*' adding (permutation) keys $\Rightarrow$ tree structure



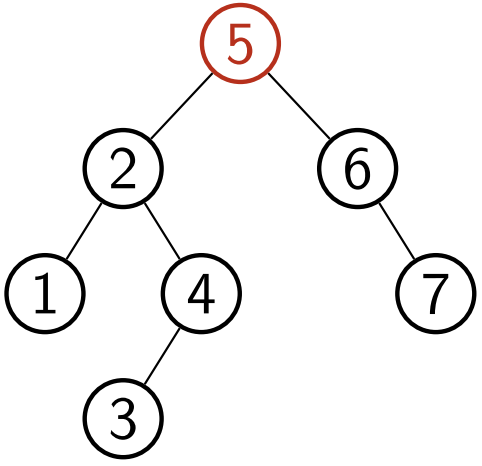
$4! = 24$ permutations but $B_4 = 14$ BST's

average ipl: $\frac{1}{24}(12 \times 4 + 4 \times 5 + 8 \times 6) = \frac{116}{24} = \frac{29}{6}$

average ipl BST ☒

I_n *average internal path length* over all BST with n nodes

average over permutations
permutation $1, \ldots, n$ into BST $\Rightarrow$ tree structure



permutation
determines left & right subtrees

$$5 \quad 2 \;\; 4 \qquad 1 \qquad 3 \quad \sim I_4$$
$$6 \qquad 7 \quad \sim I_2$$

any k root $=$ first element

$$\mathrm{I}_n = (n-1) + \tfrac{1}{n} \textstyle\sum_{k=1}^{n} (\mathrm{I}_{k-1} + \mathrm{I}_{n-k})$$

telescope! ☒

I_n average internal path length n nodes

so $$\mathrm{I}_n = (n-1) + 2(\mathrm{I}_0 + \mathrm{I}_1 + \cdots + \mathrm{I}_{n-1})/n$$

also $$\mathrm{I}_{n-1} = (n-2) + 2(\mathrm{I}_0 + \mathrm{I}_1 + \cdots + \mathrm{I}_{n-2})/(n-1)$$

subtract $$n\,\mathrm{I}_n - (n-1)\mathrm{I}_{n-1} = 2n - 2 + 2\mathrm{I}_{n-1}$$

thus $$n\,\mathrm{I}_n = (n+1)I_{n-1} + 2n - 2$$

$$\frac{\mathrm{I}_n}{n+1} = \frac{\mathrm{I}_{n-1}}{n} + \frac{2}{n+1} - \frac{2}{n(n+1)}$$

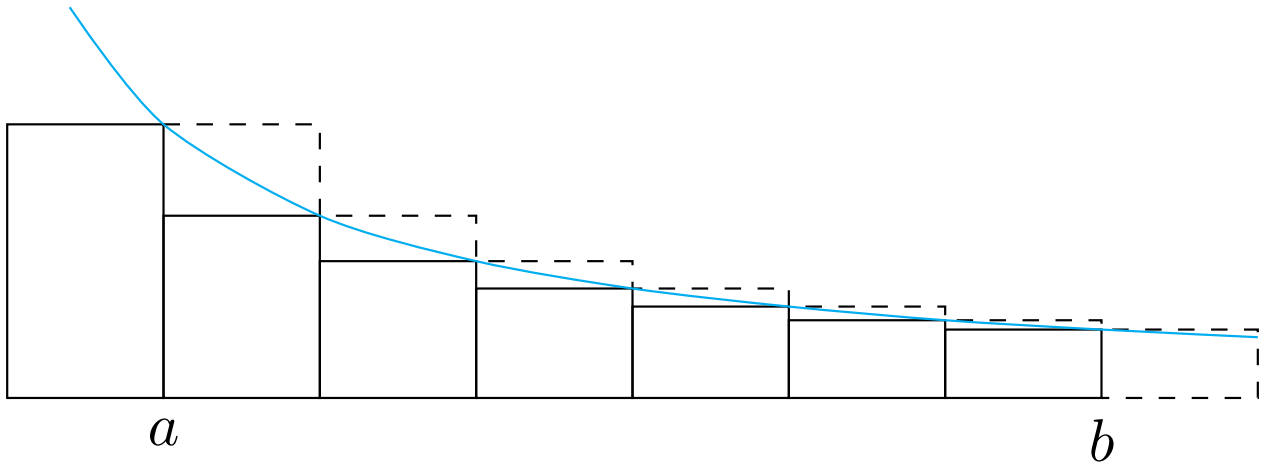
$$\frac{\mathrm{I}_{n-1}}{n} = \frac{\mathrm{I}_{n-2}}{n-1} + \frac{2}{n} - \frac{2}{(n-1)n}$$

$$\cdots$$

$$\frac{\mathrm{I}_1}{2} = \frac{\mathrm{I}_0}{1} + \frac{2}{2} - \frac{2}{1 \cdot 2}$$

$$\frac{\mathrm{I}_n}{n+1} = \frac{\mathrm{I}_0}{1} + O(\ln n) - \frac{2n}{n+1}$$

afschatten



$$\int_a^{b-1} f(x)\,dx \leq \sum_{k=a}^{b} f(k) \leq \int_{a-1}^{b} f(x)\,dx$$

$$\frac{1}{1} + \frac{1}{2} + \cdots + \frac{1}{n} = 1 + \sum_{k=2}^{n} \frac{1}{k} \leq 1 + \int_1^n \frac{1}{x}\,dx = 1 + \lg n$$

end.