Lösungen zu Aufgaben im Stoff

Foliensatz 3

Folie 29, 1

```
public interface Stack<T extends Object> {
	public void push(T obj) throws StackOverflowError;
	public T pop() throws Exception;
	public T peek();
	public boolean isEmpty();
	public void removeAll();
	public boolean isFull();
}
```

Folie 29, 2

```
public class MyStack<T extends Object> implements Stack {

	// Implementation mit einem Array
	T[] data;
	private int top;

	public MyStack(int capacity) {
		removeAll();
	}
	@Override
	public void push(Object obj) throws StackOverflowError {
		//Element auf den Stack legen
	}
	@Override
	public Object pop() throws Exception {
		//Letztes Element zurückgeben (wird entfernt)
		return null;
	}
	@Override
	public Object peek() {
		//Letztes Element zurückgehen (wird nicht entfernt)
		return null;
	}
	@Override
	public boolean isEmpty() {
		//True wenn leer
		return false;
	}
	@Override
	public void removeAll() {
		//Stack leeren
	}
	@Override
	public boolean isFull() {
		//True wenn voll
		return false;
	}

}
```

Folie 30, 1

```
public void push(Object obj) throws StackOverflowError {
	data.addFirst((T) obj);
}
```

Folie 30, 2

```
private ListStack<Integer> stack;
```

Folie 43, 1

```
public static <A extends Comparable<A>> A max (Collection<A> values) {
    Iterator<A> iterValues = values.iterator();
    A value = iterValues.next();
    while (iterValues.hasNext()) {
      A nextValue = iterValues.next();
      if (value.compareTo(nextValue) < 0) value = nextValue;
    }
    return value;
  }
```

Foliensatz 4

Folie 13

```
private void printList(List.Node n) {
	ausgabe += n.getElement().toString();
	if (n.getNext() != null) {
		printList(n.getNext());
	}
}

private void printListReverse(List.Node n) {
	if (n.getNext() != null) {
		printListReverse(n.getNext());
	}
	ausgabe += n.getElement().toString();
}
```

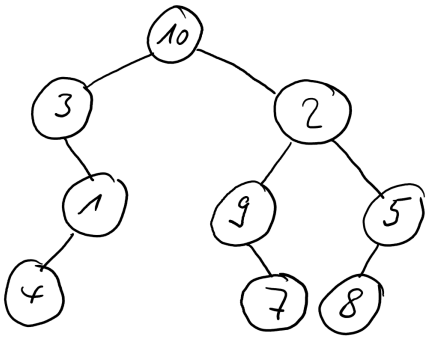
Folie 36

- $O(2^n)$

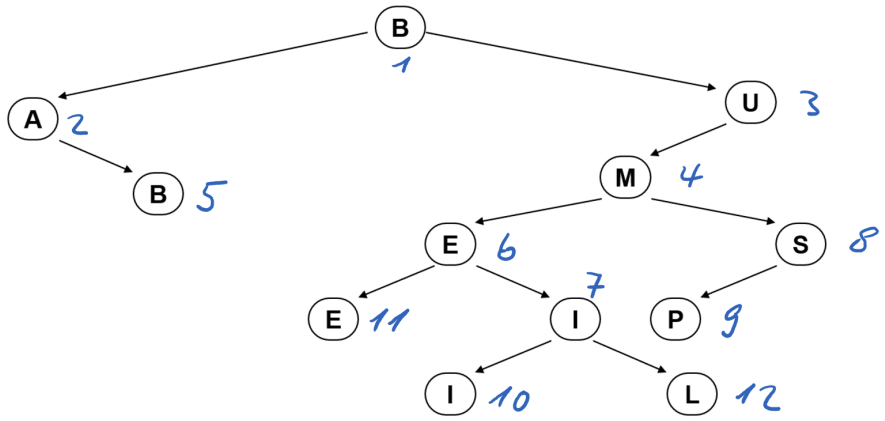
```
public int fib(int n) {
	int res = 0;
	int[] zwRes = new int[2];
	zwRes[0] = 1;
	zwRes[1] = 1;
	if (n <= 2) return 1;
	for (int i = 2; i<n;i++) {
		res = zwRes[0] + zwRes[1];
		zwRes[0] = zwRes[1];
		zwRes[1] = res;
	}
	return res;
}
```

Foliensatz 5

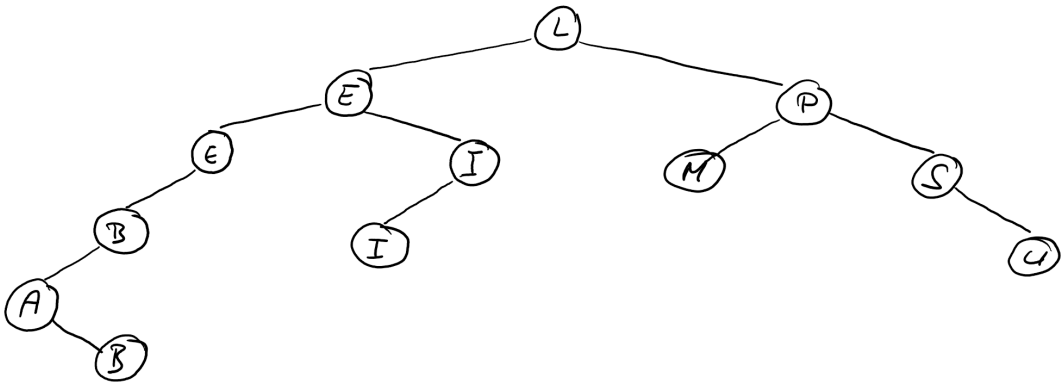
Folie 27

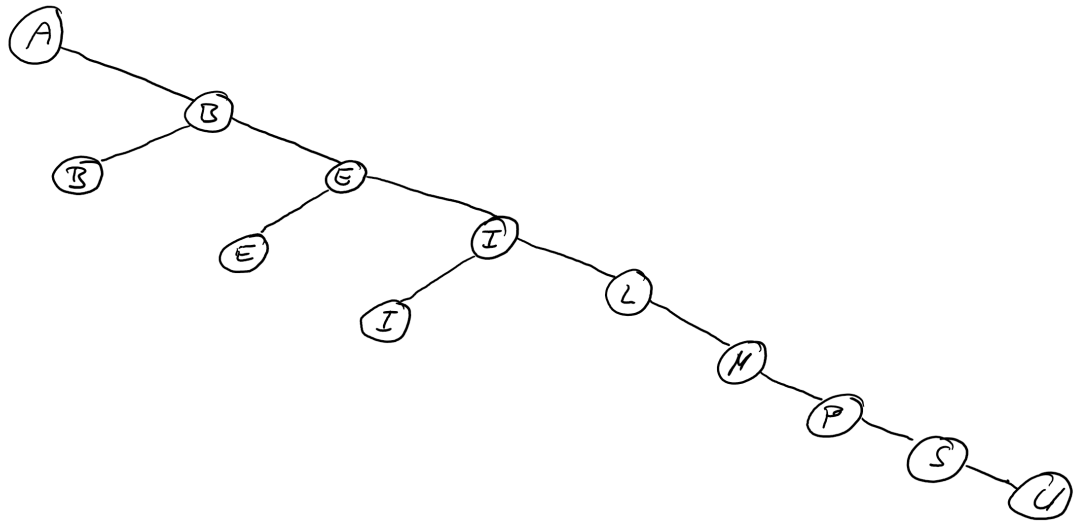


Folie 33



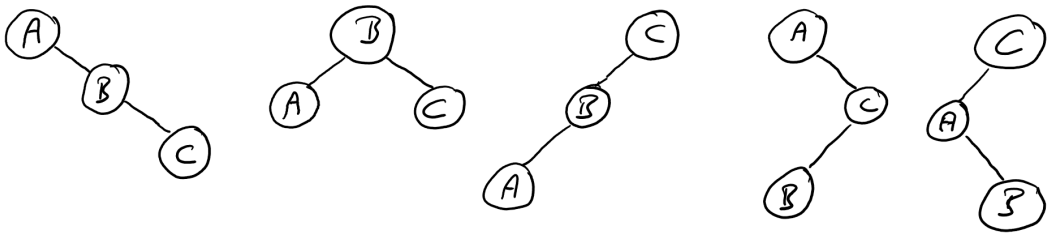
- ABBEEIILMPSU
- Inorder Traversierung

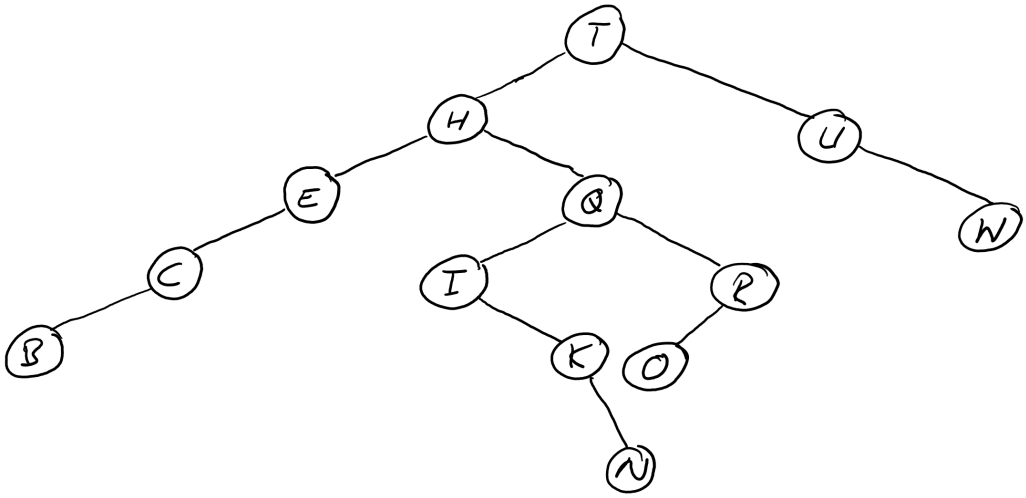


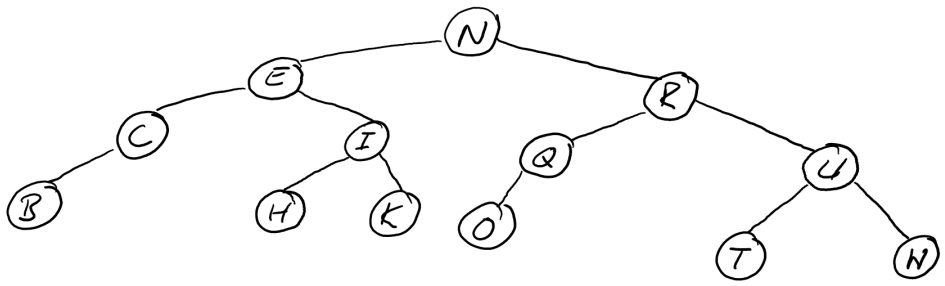


Foliensatz 6

Folie 7







Buchstaben sortieren und die Intervalle teilen

B	C	E	H	I	K	N	O	Q	R	T	U	W
1	2	3	4	5	6	7	8	9	10	11	12	13

Eingabe mit 7, 3, 5, 6, 2, 1, 10, 12, 11, 13, 9, 8

Foliensatz 7

Folie 11

(B, C, A, D, A) ist ein Pfad von B nach A. Er enthält einen Zyklus (A, D, A). (C, A, B, E) ist ein einfacher Pfad von C nach E. (F, F, F, G) ist ein Pfad mit einem Zyklus. (A, B, C, A) und (A, D, A) und (F, F) sind die einzigen geschlossenen Pfade (Zyklen). (A, B, E, A) ist kein Pfad und kein Zyklus.