

# ALGORITHMEN UND DATENSTRUKTUREN

## ÜBUNG 7: BÄUME

---

Eric Kunze

`eric.kunze@mailbox.tu-dresden.de`

TU Dresden, 11.12.2020

```
1 typedef struct element *list;  
2 struct element { int value; list next; };
```

```
1 typedef struct node *tree;  
2 struct node { int key; tree left, right; };
```

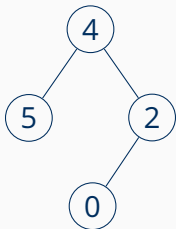
### Verknüpfen zweier Bäume mit einem neuen Wurzelnoten

- ▶ lege neuen Wurzelknoten an
- ▶ verknüpfe Wurzelknoten mit bestehenden Bäumen
- ▶ fülle neue Wurzel mit Inhalt

```
1  tree createNode(int n, tree l, tree r) {  
2      tree t    = malloc(sizeof(struct node));  
3      t->left   = l;  
4      t->right  = r;  
5      t->key    = n;  
6      return t;  
7  }
```

# TEIL (A) — ERSTELLEN EINES BAUMES

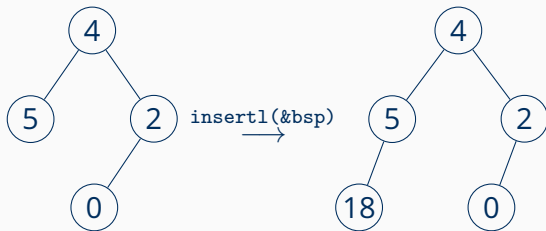
## Beispielbaum:



```
1 tree bsp =  
2   createNode(4,  
3     createNode(5, NULL, NULL),  
4     createNode(2,  
5       createNode(0, NULL, NULL),  
6       NULL));
```

## TEIL (B) — LINKS EINFÜGEN

**Beispiel:**

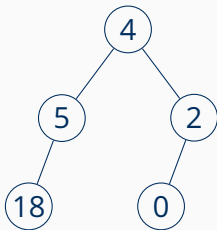


- ▶ solange kein Blatt erreicht: füge in den linken Teilbaum ein
- ▶ am Blatt: ein neuen Knoten einfügen (createNode)

```
1 void insertl(tree *tp, int n) {
2     if (*tp != NULL)
3         insertl(&((*tp)->left), n);
4     else
5         *tp = createNode(n, NULL, NULL);
6 }
```

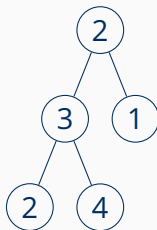
## Beispiele:

bsp



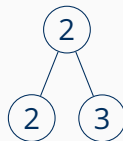
$$\text{leafprod}(\text{bsp}) = 0$$

s



$$\text{leafprod}(s) = 8$$

t

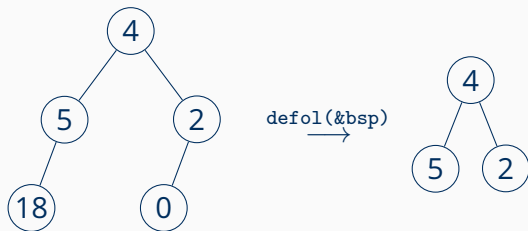


$$\text{leafprod}(t) = 6$$

- ▶ leerer Baum: leafprod = 0
- ▶ Rekursion: berechne leafprod in linkem und rechtem Teilbaum

```
1  int leafprod(tree t){  
2      if (t == NULL)  
3          return 1;  
4      if (t->left == NULL && t->right == NULL)  
5          return t->key;  
6      return leafprod(t->left) * leafprod(t->right);  
7  }
```

### Beispiel:



- ▶ Blatt: Knoten entfernen und Vorgänger auf NULL setzen
- ▶ Rekursion: Lösche in linkem und rechtem Teilbaum



## TEIL (D) — BLÄTTER ENTFERNEN

```
1 void defol(tree *tp){
2     if ( tp == NULL) return ;
3     if (*tp == NULL) return ;
4
5     if ((*tp)-> left == NULL && (*tp)->right == NULL){
6         free(*tp);
7         *tp = NULL;
8     } else {
9         defol( &((*tp)->left) );
10        defol( &((*tp)->right) );
11    }
12 }
```

## TEIL (E) — BAUM $\rightsquigarrow$ LISTE

Man übernehme nur gerade Elemente!

### Beispiel:



### Inorder – Durchlauf:

- ▶ traversiere durch linken Teilbaum
- ▶ Wurzel gerade?  $\rightarrow$  anhängen an Liste mit `append`
- ▶ traversiere durch rechten Teilbaum

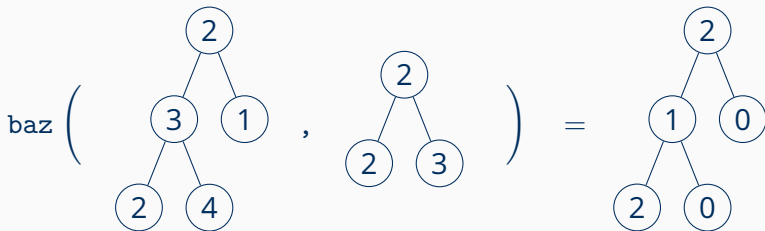
## TEIL (E) — BAUM $\rightsquigarrow$ LISTE

```
1 void append(list *lp, int n)
```

```
1 void treeToList_rec(tree t, list *lp){  
2     if (t == NULL) return ;  
3     treeToList_rec(t->left, lp);  
4     if (t->key % 2 == 0)  
5         append(lp, t->key);  
6     treeToList_rec(t->right, lp);  
7 }
```

```
1 list treeToList(tree t){  
2     list l = NULL;  
3     treeToList_rec(t,&l);  
4     return l;  
5 }
```

## Beispiel:



## TEIL (F) — ZÄHLE VORKOMMEN

```
1  int count(tree t, int k) {  
2      /* zaehlt die anzahl der vorkommen von k in t */  
3      if (t == NULL) return 0;  
4      return (t->key == k) + count(t->left, k) + count  
        (t->right, k);  
5  }
```

```
1  tree baz(tree s, tree t) {  
2      if (s == NULL) return NULL;  
3      tree result = malloc(sizeof(struct node));  
4      result->key   = count(t, s->key);  
5      result->left  = baz(s->left, t);  
6      result->right = baz(s->right, t);  
7      return result;  
8  }
```