Inhaltsverzeichnis

1 Probeklausur

1.1 Aufgabe 1

Zu zeigen:

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = c,\ 0 < c < \infty \Rightarrow f(n) \in \Theta(g(n))$$

$$\begin{aligned} \lim_{n\to\infty} \frac{f(n)}{g(n)} &= c \\ \Rightarrow \lim_{n\to\infty} f(n) &= c \cdot \lim_{n\to\infty} g(n) \\ \Rightarrow |\lim_{n\to\infty} f(n)| &= |c \cdot \lim_{n\to\infty} g(n)| \end{aligned}$$

Daraus folgt

$$\begin{array}{lcl} \exists M, n_0 \in \mathbb{N}, M \geq c \colon \forall n \leq n_0 \colon & \wedge & \exists M', n_0' \in \mathbb{N}, M' \leq \frac{1}{c} \colon \forall n \geq n_0' \colon \\ & & \begin{aligned} |f(n)| &\geq |\tfrac{1}{M'} \cdot g(n)| \\ \Rightarrow \quad |M' \cdot f(n)| &\geq |g(n)| \\ \Rightarrow \quad |g(n)| &\leq |M' \cdot f(n)| \\ \Rightarrow \quad g &\in \mathcal{O}(f) \\ \Rightarrow \quad f &\in \Omega(g) \end{aligned} \\ \begin{aligned} |f(n)| &\leq |M \cdot g(n)| \\ \Rightarrow \quad f &\in \mathcal{O}(g) \end{aligned} & \wedge & \end{array}$$

Da $f \in \Omega(g) \wedge f \in \mathcal{O}(g) \Rightarrow f \in \Theta(g)$

1.2 Aufgabe 2

1)

```
struct ListElem {
  int data;
  ListElem *next, *prev;
};
```

```
struct DynamicLinkedList {

  ListElem *head, *tail;
};
```

```
void pushfront(int x) {
  ListElem *e = new ListElem;
  e->data = x;
  if (head == NULL) {
    head = tail = e;
  } else {
    e->next = head;
    head->prev = e;
    head = e;
  }
}
```

```
int popfront() {
  ListElem *e = head;
  int r = e->data;
  head = e->next;
  head->prev = NULL;
  delete e;
  return r;
}
```

2)

```
int size() {
  int r = 0;
  ListElem *e = head;
  while (e != NULL) {
    r++;
    e = e->next;
  }
  return r;
}
```

```
DynamicLinkedList*[] splithalf() {
  if (head == NULL)
    return new DynamicLinkedList*[2];

  DynamicLinkedList *l1 = new DynamicLinkedList;
  DynamicLinkedList *l2 = new DynamicLinkedList;
  l1->head = head; l1->tail = head;
  l2->tail = tail; l2->head = tail;

  while (l1->tail != l2->head->prev &&
         l1->tail->next != l2->head->prev) {
    l1->tail = l1->tail->next;
    l2->head = l2->head->prev;
  }
  l1->tail->next = NULL;
  l2->head->prev = NULL;
  return new DynamicLinkedList*[] { l1, l2 };
}
```

3)

```
DynamicLinkedList *merge(DynamicLinkedList *l1, DynamicLinkedList *l2) {
  if (l1->head == NULL) return l2;
  if (l2->head == NULL) return l1;

  DynamicLinkedList *l = new DynamicLinkedList;
  ListElem *e1 = l1->head;
  ListElem *e2 = l2->head;
  l->head = new ListElem;
  if (e1->data <= e2->data) {
    l->head->data = e1->data;
    e1 = e1->next;
  } else {
    l->head->data = e2->data;
    e2 = e2->next;
  }
  l->tail = l->head;

  while (e1 != NULL && e2 != NULL) {
    ListElem *e = new ListElem;
    e->prev = l->tail;
    l->tail->next = e;
    l->tail = e;
    if (e1->data <= e2->data) {
      e->data = e1->data;
      e1 = e1->next;
    } else {
      e->data = e2->data;
      e2 = e2->next;
    }
  }

  ListElem *e0 = (e1 == NULL) ? e2 : e1;
  while (e0 != NULL) {
    ListElem *e = new ListElem;
    e->prev = l->tail;
    l->tail->next = e;
    l->tail = e;
    e->data = e0->data;
    e0 = e0->next;
  }

  return l;
}
```

4)

```
DynamicLinkedList *mergesort(DynamicLinkedList *l) {
  if (l->size() <= 1)
    return l;
  DynamicLinkedList*[] lists = l->splithalf();
  return merge(mergesort(lists[0]), mergesort(lists[1]));
}
```

1.3 Aufgabe 3

$p :=$ Pivot, $i :=$ Linker Zeiger, $j :=$ Rechter Zeiger

5	3	17	10	84	19	6	22	9
$\overset{\uparrow}{p}$	$\overset{\uparrow}{j}$	$\overset{\uparrow}{i}$						
3	5	17	10	84	19	6	22	9
$\surd$	$\overset{\uparrow}{p,j}$	$\overset{\uparrow}{i}$						
3	5	17	10	84	19	6	22	9
$\surd$	$\surd$	$\overset{\uparrow}{p}$		$\overset{\uparrow}{i}$				$\overset{\uparrow}{j}$
3	5	17	10	9	19	6	22	84
$\surd$	$\surd$	$\overset{\uparrow}{p}$			$\overset{\uparrow}{i}$	$\overset{\uparrow}{j}$		
3	5	17	10	9	6	19	22	84
$\surd$	$\surd$	$\overset{\uparrow}{p}$			$\overset{\uparrow}{j}$	$\overset{\uparrow}{i}$		
3	5	6	10	9	17	19	22	84
$\surd$	$\surd$	$\overset{\uparrow}{p_1,j_1}$	$\overset{\uparrow}{i_1}$		$\surd$	$\overset{\uparrow}{p_2,j_2}$	$\overset{\uparrow}{i_2}$	
3	5	6	10	9	17	19	22	84
$\surd$	$\surd$	$\surd$	$\overset{\uparrow}{p_1}$	$\overset{\uparrow}{i_1,j_1}$	$\surd$	$\surd$	$\overset{\uparrow}{p_2,j_2}$	$\overset{\uparrow}{i_2}$
3	5	6	9	10	17	19	22	84
$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$

1.4 Aufgabe 4

1.5 Aufgabe 5

```
TreeNode *createTree(int[] a, int n) {
  return createTree(a, 0, n);
}

TreeNode *createTree(int[] a, int l, int r) {
  if (l > r) return NULL;
  TreeNode *n = new TreeNode;
  int m = (l + r + 1) / 2;
  n->data = a[m];
  n->left = createTree(a, l, m - 1);
  n->right = createTree(a, m + 1, r);
  return n;
}
```

```
TreeNode *lookup(Tree *t, int x) {
  TreeNode *n = t->root;
  while (n != NULL && n->data != x) {
    if (n->data < x)   n = n->left;
    else               n = n->right;
```

```
  }
  return n;
}
```

1.6 Aufgabe 6

1.7 Aufgabe 7

$\rightarrow a$		$\texttt{dfsnum}[a] = 1$	$\texttt{compnum}[a] = 8$
$\quad \rightarrow b$		$\texttt{dfsnum}[b] = 2$	$\texttt{compnum}[b] = 7$
$\quad\quad \rightarrow c$		$\texttt{dfsnum}[c] = 3$	$\texttt{compnum}[c] = 5$
$\quad\quad\quad \rightarrow d$		$\texttt{dfsnum}[d] = 4$	$\texttt{compnum}[d] = 2$
$\quad\quad\quad\quad \rightarrow c$	bekannt		
$\quad\quad\quad\quad \rightarrow h$		$\texttt{dfsnum}[h] = 5$	$\texttt{compnum}[h] = 1$
$\quad\quad\quad \rightarrow g$		$\texttt{dfsnum}[g] = 6$	$\texttt{compnum}[g] = 4$
$\quad\quad\quad\quad \rightarrow f$		$\texttt{dfsnum}[f] = 7$	$\texttt{compnum}[f] = 3$
$\quad\quad\quad\quad\quad \rightarrow g$	bekannt		
$\quad\quad\quad\quad \rightarrow h$	bekannt		
$\quad\quad \rightarrow f$	bekannt		
$\quad\quad \rightarrow e$		$\texttt{dfsnum}[e] = 8$	$\texttt{compnum}[e] = 6$
$\quad\quad\quad \rightarrow f$	bekannt		
$\quad\quad\quad \rightarrow a$	bekannt		