C02 Binärer Suchbaum

panitz

Zusammenfassung

9 Aufgaben für binäre Suchbäume in der Programmiersprache C. Die Elemente sind int-Zahlen.

Frage: Elemente sortiert in Array schreiben

Gegeben sei folgende Struktur für Suchbäume:

```
typedef struct TreeNode {
  struct TreeNode* left;
  int element;
  struct TreeNode* right;}
 Tree;
```

Schreiben Sie die Funktion, die die Elemente aufsteigend sortiert in das per Referenz übergebene Array schreibt. Die Array in der Array Referenz ist dabei immer auf die nächste Position zu setzen.

C

```
      #include "Tree.h"
void intoArray(Tree* this, int* result){
  intoArrayWork(this, &result);
}

void intoArrayWork(Tree* this, int** result){
}
```

C

Lösung

C

```c
#include "Tree.h"

void intoArray(Tree* this, int* result){
  intoArrayWork(this, &result);
}

void intoArrayWork(Tree* this, int** result){
  if (this==NULL) return;
  intoArrayWork(this->left,result);
  **result = this-> element;
  ++*result;
  intoArrayWork(this->right,result);
}
```

C

Erläuterung

Erst ist der linke Teilbaum einzufügen, dann das Wurzelelement und dann schließlich der rechte Teilbaum einzufügen.Nach dem Einfügen des Wurzelelements ist jeweils der Zeiger eins weiter zu setzen.

Frage: Maximale Baumtiefe

Gegeben sei folgende Struktur für Suchbäume:

```
typedef struct TreeNode {
  struct TreeNode* left;
  int element;
  struct TreeNode* right;}
 Tree;
```

Schreiben Sie die Funktion, die die maximale Tiefe im Baum berechnet. Damit ist die größte Anzahl von Knoten auf einem Pfad von Wurzel bis zu einem Blatt gemeint.

C

```
      #include "Tree.h"
unsigned int maxDepth(Tree* this){
  return 0;
}
```

C

Lösung

C

```c
        #include "Tree.h"
unsigned int maxDepth(Tree* this){
  if (this==NULL) return 0;
  unsigned int lD = maxDepth(this->left);
  unsigned int rD = maxDepth(this->right);

  return 1 + (lD>rD?lD:rD);
}
```

C

Erläuterung

Der Nullpointer hat keinen Pfad un damit die maximale Tiefe 0. Ansonsten ist die maximale Tiefe der beiden Kinder zu wählen und eins für die Wurzel hinzu zu rechnen.

Frage: Summer der Elemente

Gegeben sei folgende Struktur für Suchbäume:

```
typedef struct TreeNode {
  struct TreeNode* left;
  int element;
  struct TreeNode* right;}
 Tree;
```

Schreiben Sie die Funktion, die die Summe der Elemente im den Baum berechnet. Ein Nullpointer habe die Summe 0.

C

```
      #include "Tree.h"
long int sum(Tree* this){
  return 0;
}
```

C

Lösung

C

```
        #include "Tree.h"
long int sum(Tree* this){
  if (this==NULL) return 0;
  return this->element + sum(this->left) + sum(this->right);
}
```

C

Erläuterung

Es sind aufzusummieren, die Summe aus den Kindbäumen plus das Wurzelelement.

Frage: Enthält mit Eigenschaft

Gegeben sei folgende Struktur für Suchbäume:

```
typedef struct TreeNode {
  struct TreeNode* left
  int element;
  struct TreeNode* right;
}Tree;
```

Schreiben Sie eine Funktion, die testet ob ein Element mit einer bestimmten Eigenschaft im Baum enthalten ist. Die Eigenschaft wird als Funktionspointer der Testfunktion übergeben.

C

```
      #include "Tree.h"

bool containsWithPred(Tree* this, bool test(int)){
  return false;
}
```

C

Lösung

C

```c
        #include "Tree.h"

bool containsWithPred(Tree* this, bool test(int)){
  return this!=NULL &&
      (test(this->element)
          || containsWithPred(this->left,test)
          ||containsWithPred(this->right,test)
          );
}
```

C

Erläuterung

Hier kann man nicht die Sucheigenschaft eines binären Suchbaumes nehmen, weil die Testeigenschaft ja beliebig sein kann. Alle Teile des Baumes müssen notfalls durchlaufen werden.

Frage: Aus Speicher Löschen

Gegeben sei folgende Struktur für Suchbäume:

```
typedef struct TreeNode {
  struct TreeNode* left
  int element;
  struct TreeNode* right;
}Tree;
```

Schreiben Sie eine Destruktor-Funktion, die den Baum komplett aus dem Speicher löscht.

C

```
      #include "Tree.h"

void deleteTree(Tree* this){
}
```

C

Lösung

C

```c
#include "Tree.h"
void deleteTree(Tree* this){
  if (this==NULL) return;
  deleteTree(this->left);
  deleteTree(this->right);
  free(this);
}
```

C

Erläuterung

Erst sind rekursiv die Kinder zu löschen, bevor ein `free` auf den eigentlichen Pointer gemacht werden kann.

Frage: Größtes Element

Gegeben sei folgende Struktur für Suchbäume:

```
typedef struct TreeNode {
  struct TreeNode* left
  int element;
  struct TreeNode* right;
}Tree;
```

Schreiben Sie eine Funktion, die das größte Element errechnet. Dabei ist davon auszugehen, dass der Parameter kein Nullpointer ist.

C

```
      #include "Tree.h"

int largestElement(Tree* this){
  return 0;
}
```

C

Lösung

C

```c
#include "Tree.h"

int largestElement(Tree* this){
  if (this->right==NULL) return this->element;
  return largestElement(this->right);
}
```

C

Erläuterung

Das größte Element eines binären Suchbaumes ist das Blatt, das am rechtesten Rand steht.

Frage: Enthält ein Element

Gegeben sei folgende Struktur für Suchbäume:

```
typedef struct TreeNode {
  struct TreeNode* left;
  int element;
  struct TreeNode* right;}
 Tree;
```

Schreiben Sie die Funktion, die testet ob ein Elemente im den Baum enthalten ist.

C

```
      #include "Tree.h"

bool contains(Tree* this, int element){
  return false;
}
```

C

Lösung

C

```c
#include "Tree.h"

bool contains(Tree* this, int element){
  if (this==NULL) return false;
  if (element==this->element) return true;
  if (element<this->element) return contains(this->left,element);
  return contains(this->right,element);
}
```

C

Erläuterung

Ein Nullpointer enthält kein Element. Ansonsten ist in einem binären Suchbaum das Wurzelelement mit dem gesuchten Element zu vergleichen. An diesem Vergleich entscheidet sich, ob das gesuchte Element im linken oder rechten Teilbaum zu finden ist.

Frage: Anzahl der Elemente

Gegeben sei folgende Struktur für Suchbäume:

```
typedef struct TreeNode {
  struct TreeNode* left;
  int element;
  struct TreeNode* right;}
 Tree;
```

Schreiben Sie die Funktion, die die Anzahl der Elemente im den Baum berechnet.

C

```
      #include "Tree.h"
unsigned int size(Tree* this){
  return 0;
}
```

C

Lösung

C

```c
        #include "Tree.h"
unsigned int size(Tree* this){
  if (this==NULL) return 0;
  return 1 + size(this->left) + size(this->right);
}
```

C

Erläuterung

der Nullpointer repräsentiert einen leeren Baum der Größe 0. Ansonsten ergibt sich die Größe aus der Summe der Größe der Teilbäume plus 1.

Frage: Einfügen

Gegeben sei folgende Struktur für Suchbäume:

```
typedef struct TreeNode {
  struct TreeNode* left;
  int element;
  struct TreeNode* right;}
 Tree;

Tree* newTree(int element);
```

Schreiben Sie die Funktion zum Einfügen in den Baum, so dass eine Menge realisiert wird.

C

```
      #include "Tree.h"

void insert(Tree* this, int element){
}
```

C

Lösung

C

```c
        #include "Tree.h"
void insert(Tree* this, int element){
  if (element < this->element){
    if(this->left==NULL) this-> left = newTree(element);
    else insert(this->left,element);
  }
  if (element > this->element){
    if(this->right==NULL) this-> right = newTree(element);
    else insert(this->right,element);
  }
}
```

C

Erläuterung

Es ist zu prüfen, ob das Element im linken oder rechten Teilbaum zu verorten ist. Dann ist zu prüfen, ob es dort bereits einen Baum gibt, oder einen Nullpointer. Je nachdem ist ein rekursiver Aufruf notwendig, oder aber ein neues Blatt zu erzeugen.