Theorie zum 1.SWP-OP Test

1. Unterschied zwischen linearer und binärer Suche?  
   A: Die lineare Suche ist der einfachste Suchalgorithmus, welcher existiert.  
   Die Aufgabe besteht darin, ein Element in einer Liste oder einem Array mit n Elementen zu finden. Man geht dazu die Liste Element für Element durch, bis man es gefunden hat. Der Suchaufwand wächst linear mit der Anzahl der Elemente in der Liste. Die Liste muss nicht sortiert werden.

Die binäre Suche ist ein Algorithmus, der aus einer Liste sehr effizient ein gesuchtes Element findet bzw. eine zuverlässige Aussage über das Fehlen dieses Elementes liefert. Voraussetzung ist, dass die Elemente in der Liste sortiert sind. Um in eine Liste mit n Einträgen auf An- oder Abwesenheit eines Elementes zu überprüfen, werden maximal log2(n+1) Vergleichsschritte benötigt. Damit ist sie deutlich schneller als die lineare Suche, welche allerdings den Vorteil hat, auch in unsortierten Listen zu funktionieren.

1. Wie ist der algorithmische Aufbau einer Binärsuche?

A: Zuerst wird das mittlere Element des Felds überprüft. Es kann kleiner, größer oder gleich dem gesuchten Element sein. Ist es kleiner als das gesuchte Element, muss das gesuchte Element in der hinteren Hälfte stecken, falls es sich dort überhaupt befindet. Ist es hingegen größer, muss nur in der vorderen Hälfte weitergesucht werden. Die jeweils andere Hälfte muss nicht mehr betrachtet werden. Ist es gleich dem gesuchten Element, ist die Suche beendet.

In der zu untersuchenden Hälfte (und erneut in den folgenden Hälften) wird genauso verfahren: Das mittlere Element liefert wieder die Entscheidung darüber, ob und wo weitergesucht werden muss. Die Länge des Suchbereiches wird so von Schritt zu Schritt halbiert. Spätestens wenn der Suchbereich auf ein einzelnes Element geschrumpft ist, ist die Suche beendet. Dieses eine Element ist entweder das gesuchte Element, oder das gesuchte Element kommt nicht vor.

Der Algorithmus zur binären Suche wird entweder als Iteration oder Rekursion implementiert. (Siehe HÜ!)

1. Was ist eine iterative Methode?

A: Unter Iteration versteht man ein mehrmaliges Ausführen einer Aktion. Eine iterative Methode ist eine Methode, welche Schleifen (for, while,..) nutzt. Die Schleifen besitzen eine Anfangs - und Abbruchbedingung. Die Schleifen werden so lange durchlaufen, bis die Abbruchbedingung die Schleife beendet.

1. Was ist eine rekursive Methode?  
   A: Unter Rekursion versteht man ebenfalls eine wiederholte Ausführung, doch diesmal gibt es keine Zählschleife, sondern die Methode ruft sich selbst mit anderen Parametern erneut auf. Dies geschieht so lange, bis eine Abbruchbedingung erfüllt wurde.
2. Vorteile und Nachteile von Iterationen?  
   A: Vorteile: geringerer Speicheraufwand  
    schnellere Ausführung   
    leichter verständlich  
     
    Nachteile: weniger übersichtlich  
    nicht so kompakt/kurz
3. Vorteile und Nachteile einer Rekursion?  
   A: Vorteile: übersichtlicher  
    kompakter/kürzer  
     
    Nachteile: höherer Speicheraufwand  
    langsamere Ausführung (durch aufrufen von   
    Unterprogrammen)  
    für Programmieranfänger nicht so leicht verständlich
4. Was ist eine Endrekursion?  
   A: Im Gegensatz zur normalen Rekursion wird im letztem Schritt keine Berechnung ausgeführt und zurückgegeben, sondern bei der Endrekursion wird im letztem Schritt die Funktion selbst aufgerufen.
5. Was ist der Stack?  
   A: Der Name deutet an, dass die Daten hier "aufeinander" liegen. Damit ist gemeint, dass neue Daten immer nur oben drauf gelegt werden können. Wenn die Daten wieder freigegeben werden, werden Sie von oben nach unten wieder entfernt. Dieses Prinzip nennt sich auch LIFO ("Last in, First out"). Der Stack kann, bedingt durch seine Struktur, sehr effizient verwaltet werden, weshalb Stack-Operationen sehr schnell sind.

Jeder Thread eines Programmes erhält für den Stack einen eigenen Speicherbereich mit fixer Größe zugewiesen. Darauf werden Informationen zum Programmablauf (z.B. Funktionsparameter) und lokale Variablen gespeichert. Beim Anlegen neuer lokaler Variablen wächst der Stack an und beim Verlassen des Sichtbarkeitsbereichs ("Scope") schrumpft er wieder und der Speicher wird automatisch aufgeräumt.

1. Eigenschaften eines Stacks?

A: Begrenzte Größe

a)LIFO Datenstruktur (die zuletzt angelegten Daten werden als erstes wieder freigegeben, deshalb auch "Stapel")

b)Wächst und schrumpft mit dem Programmverlauf

c)Wird verwendet für lokale Variablen und Funktionsparameter

d)Kein explizites Freigeben des Speichers nötig

e)Das Ablegen und Entfernen von Elementen ist sehr effizient

1. Was ist ein Heap?  
   A: Der Heap ist nicht so strukturiert wie der Stack. Du kannst ihn dir tatsächlich als Haufen vorstellen, auf dem jede Menge Platz ist. Während der Stack nämlich von der Größe her stark begrenzt ist, kann der Heap anwachsen bis die Speichergrenze auf Prozessebene erreicht ist. Dafür ist der Heap aber intern nicht so einfach zu verwalten, was ihn langsamer als den Stack macht. Auf dem Heap angelegter Speicher muss auch explizit wieder freigegeben werden (durch den Programmierer oder z.B. den Garbage Collector, je nach Programmiersprache).

Für den Zugriff auf den Heap werden Zeiger verwendet auch wenn das nicht immer direkt ersichtlich ist. Java und C# verwenden zum Beispiel Referenzen für Objekte die auf dem Heap liegen. Im Hintergrund muss aber natürlich trotzdem mit Zeigern gearbeitet werden, welche die Speicheradresse des Objektes beinhalten.  
  
Da auf dem Heap angelegte Objekte nicht auf den lokalen Sichtbarkeitsbereich beschränkt sind, kann global darauf zugegriffen werden (sofern ein Zeiger oder eine Referenz vorhanden ist).

1. Eigenschaften eines Heaps?  
   A: a)Der Heap kann innerhalb der Prozessgrenze beliebig groß werden  
   b)Anlegen und freigeben von Objekten ist vergleichsweise langsam  
   c)Auf dem Heap angelegte Objekte können global verfügbar gemacht werden  
   d)In Programmiersprachen ohne Garbage Collector muss der Speicher manuell freigegeben werden, wenn er nicht mehr benötigt wird
2. Verwendung von Stack und Heap!  
   A: Stack: Der Stack ist der Speicher zur Verwaltung von Methoden   
    und deren lokalen Variablen  
     
    Heap: Der Heap ist der Speicher zur Verwaltung von Objekten und   
    globalen Varibalen
3. Was ist ein FIFO?  
   A: First In – First Out , häufig abgekürzt mit FIFO, gleichbedeutend mit „First come, first served.“ Bezeichnet jegliche Verfahren der Speicherung, bei denen diejenigen Elemente, die zuerst gespeichert wurden, auch zuerst wieder aus dem Speicher entnommen werden. Eine solche Datenstruktur wird auch als (Warte-)Schlange bezeichnet.
4. Was ist ein FILO?  
   A: Last In – First Out“, häufig abgekürzt mit FILO, ist das „Highest In – First Out“-Verfahren (HIFO), bei dem das Element zuerst entnommen wird, welches den höchsten Wert besitzt und das „Lowest In – First Out“-Verfahren (LOFO), bei dem als erstes das niederstwertige Element entnommen wird.
5. Beispiele für FIFO bzw. FILO?  
   A: FIFO: Schieberegister, Kamera in Auto

FILO: Stack, Flugzeug Blackbox

1. Was sind mutable (veränderbare) Datenstrukturen?  
   A: Mutable Datenstrukturen wie eine ArrayList sind nach dem erstellen in ihrer Größe veränderbar und können jederzeit um ein Element erweitert werden ohne, dass eine neue ArrayList angelegt werden muss. Die Elemente werden einfach hinten an die ArrayList drangehängt.
2. Immutable (unveränderbare) Datenstrukturen?  
   A: Immutable Datenstrukturen wie ein Array sind nach dem erstellen in ihrer Größe nicht veränderbar. Jedes Mal, wenn die Größe des Arrays verändert wird, wird ein neuer Array erstellt, da beim Erstellen eines Arrays eine fixe Größe angegeben wird, welche nicht überschritten werden kann, ohne dass ein neuer Array angelegt wird.