Prüfungsprotokoll Praktische Informatik (B.Sc. Informatik)
Datum: 20.4.2006
Prüfer: Prof. Dr. Haake
Beisitzerin: Dr. Anja Haake
Note: 1,0

Datenstrukturen:

Was ist ein Algorithmus?
Verfahren zur Lösung eines Problems. Besteht aus Typen der Eingabe und Ausgabe, den Verarbeitungschritten und dazugehörigen Kontrollvorschriften.

Zur O-Notation: Wie sind die Abstraktionsschritte?
Siehe Skript.

Wenn man jetzt zwei verschiedene Komplexitätsklassen hat und in jeder Klasse hat man einen worst-case, wie bestimmt man denn dann den insgesamt schlechtesten Fall?
Na, schlechtester Fall aus allen Komplexitätsklassen ist halt der worst-case.

Was kann man denn mit der O-Notation noch darstellen?
Hmm… Ich bin jetzt nicht ganz sicher worum es Ihnen geht. Wollen Sie auf die Omega-Notation hinaus?

Nein. Was ist denn neben der Laufzeit noch ein Kriterium für die Qualität eines Algorithmus?
Der Speicherbedarf.

Genau und auch das Verhalten des Speicherbedarfs lässt sich mit der O-Notation darstellen. Wie ist die Laufzeit um eine n-elementige Menge zu sortieren?
Kommt auf den verwendeten Algorithmus an. Gute allgemeine Algorithmen erreichen O(n*log n), eingeschränkte Verfahren, also Verfahren die nicht nur auf einem Schlüsselvergleich beruhen, können auch bessere Laufzeiten erreichen.

Und ganz allgemein, also welche Laufzeit erreichen ganz einfache Verfahren?
O(n^2), da jeder Schlüssel mit allen verglichen wird

Was benutzt man denn um ein Element in einer Menge zu suchen?
Allgemein: Ein Dictionary, also z.B. eine Hashliste.

Wie ist die Laufzeit im Kollisionsfall (bei offenem hashing)?
O(n), also abhängig von der Anzahl der Elemente in der Liste (im Bucket).

Wie lässt sich das verbessern?
Es gar nicht erst zur Kollision kommen lassen, also die Anzahl der Buckets erhöhen. Und dann könnte man noch statt einer Liste im Bucket einen binären Baum verwenden.

Und welche Laufzeit hätte man damit erreicht?
O(log n) statt O(n).

Was passiert denn wenn ich aus Elementen in sortierter Reihenfolge einen Baum aufbaue?
Es entsteht ein degenerierter Baum, also eine Liste.

Was kann man da machen?
AVL-Baum, also durch Rebalancieren nach jeder Einfüge- und Löschoperation sicherstellen, dass der Baum nicht zu einer Liste entartet.

Wie viele Rotationen sind denn bei einem AVL-Baum nötig?
Beim Einfügen eine Einfach- oder Doppelrotation, beim Löschen muss man zusätzlich noch den Baum bis zur Wurzel hoch gehen und möglicherweise weitere Rotationen durchführen.

Betriebssysteme:

Was ist ein Programm und ein Prozess?
Ein Prozess ist ein Programm in Ausführung.

(Wie wir von Prozessen auf Threads kamen, weiss ich nicht mehr. Hier fehlen also ein paar Fragen.)

Was teilen Threads nicht?
Registerinhalte, Stacksegment

Was noch? Codesegment hatten wir ja schon erwähnt, Stacksegment auch.
Datensegment.

Kann es bereits zu Synchronisationsproblemen kommen wenn nur ein Thread schreibt und alle anderen nur Lesen?
Eigentlich nicht, wenn die Schreiboperation atomar ist…

Davon habe ich aber nichts gesagt. Ein Thread kann ja unterbrochen werden und wer weiß wie groß der zu schreibende Puffer ist.
Ok, dann kann es natürlich sein, dass ein lesender Thread Daten liest, die von einem Thread erst zum Teil geschrieben bzw. aktualisiert worden sind.

Was macht man um dieses Problem zu lösen?
Man verwendet Synchronisationsobjekte, also z.B. Semaphoren.

Welche gibt es noch?
Mutexe.

Welche noch?
Keine Ahnung.

Monitore. Macht nichts. Wie sind Semaphoren aufgebaut?
Warteschlange für die wartenden Prozesse, Zählvariable die die Anzahl der noch freien Betriebsmittel angibt.

Und wie sind die Operationen?
Up und down.

Und was ruft ein Prozess auf der ein Betriebsmittel belegen will und was passiert dann?
Er ruft down auf und wenn count>0 ist, wird count dekrementiert. Wenn nicht wird der Prozess in die Warteschlange aufgenommen und blockiert.

Und wenn der Prozess das Betriebsmittel wieder freigibt?
Ruft er up auf. Wenn es Prozesse in der Warteschlange gibt, wird ein Prozess daraus entnommen und rechnend, neee, bereit gesetzt, wenn nicht, dann wird count inkrementiert.

Rechnernetze:

Welche Beispiele für Protokolle auf der Transportschicht haben wir denn im Kurs näher kennen gelernt?
TCP und UDP.

Wie unterscheiden sich UDP und TCP?
TCP ist ein verbindungsorientiertes, zuverlässiges Protokoll, UDP ist ein verbindungsloses, unzuverlässiges Protokoll.

Sagen wir sie möchten ein Internet-Radio-Anwendung implementieren. Welches Protokoll wählen sie und warum?
UDP, denn es kommt in erster Linie auf Schnelligkeit an, nicht auf Zuverlässigkeit.

Warum ist UDP schneller?
Zum einen hat UDP einen kleineren Header, zum anderen ist es ein verbindungsloses Protokoll, d.h. im Gegensatz zu TCP gibt es keine initiale Verzögerung beim Verbindungsaufbau (3-Wege-Handshake).

Um jetzt die übertragenen Nachrichten wiederzugeben, was müssen sie dafür tun?
Reihenfolge wiederherstellen

Und was machen Sie mit Paketen die zu früh sind, von denen sie also noch nicht alle Vorgänger haben?
Natürlich speichern, schließlich werden diese Pakete bei UDP ja nicht noch mal versandt.

Das heißt was braucht man?
Einen Puffer.

Und wie arbeitet TCP?
Eigentlich ähnlich wie wir es gerade besprochen haben, d.h. es ist ein Fensterprotokoll. TCP arbeitet mit kumulativen Bestätigungen.

Was ist denn wenn eine Nachricht verfälscht ankommt? Welche beiden Möglichkeiten hat der Empfänger dann allgemein?
NACK-Nachricht senden oder einfach nichts tun und auf erneutes Senden warten, d.h. der Sender lässt einen Timer laufen und sendet Pakete für die er keine ACKs bekommen hat von sich aus noch mal.

Wenn der Sender automatisch nach einer gewissen Zeit Pakete die nicht bestätigt wurden neu sendet, wie schnell muss der Timer auslaufen?
Na, also er muss mindestens die doppelte Paketlaufzeit warten. Wenn er zu schnell ausläuft werden Pakete unnötig erneut verschickt, also obwohl eigentlich bereits eine ACK-Nachricht unterwegs ist. Und wenn er zu langsam ausläuft, dauert es halt sehr lange bis diese Pakete letztendlich korrekt beim Empfänger verfügbar sind.

Und worin liegt jetzt das Problem bei der Festlegung des Timeouts? Wenn man mal bedenkt, dass Sender und Empfänger ja nicht direkt miteinander verbunden sind.
Ach so, na man kann wegen des unvorhersehbaren Routings ja gar nicht genau angeben wie lange eine ACK-Nachricht auf sich warten lässt.

Und was macht TCP deshalb?
Ähhh…

Naja, hatten wir ja eigentlich schon besprochen.
Ach so, NACK-Messages statt in den Timeout laufen lassen.

Braucht man für jedes Paket einen Timer?
Ja.

<u>Anmerkungen:</u>
Reihenfolge und Wortlaut der Fragen war manchmal vielleicht anders. Ich habe versucht die Fragen und Antworten sinngemäß wiederzugeben. Ich denke es fehlen auch noch ein paar Fragen.
Ich fand die Benotung äußerst fair. Ich war zwar durchaus gut vorbereitet, hätte aber nach der Prüfung eher eine um ein- bis zweidrittel schlechtere Note erwartet.
Es war eine große Anzahl von Fragen, die man jeweils relativ kurz beantworten konnte, was mir mehr liegt als über ein Thema frei und länger zu referieren. Oft hätte ich zu einer Frage noch mehr sagen können/wollen, Herr Haake stellte aber bereits die nächste Frage, war also mit der Antwort bereits zufrieden.
Mein Eindruck ist, dass es besser ist bei einer Frage zu sagen, dass man sie nicht beantworten kann um die Zeit zur Beantwortung anderer Fragen zu nutzen, anstatt minutenlang herumzueiern.
Herr Haake ist freundlich, verbindlich und versucht einen immer auf die Sprünge zu helfen, formuliert also z.B. eine Frage immer in einem Zusammenhang und gibt einen Hinweis wenn man nicht sofort auf die Antwort kommt. Wenn man dann die Frage beantworten kann, haben diese Hinweise (offensichtlich) keinen negativen Einfluss auf die Note, also sollte man das auch nutzen.
Ich empfehle zur Vorbereitung auf die Prüfung das Studium der Prüfungsprotokolle bzw. des Prüfungsfragenkatalogs. Vielen Dank an Werner Zeindl-Nagele für seine Kurszusammenfassungen und an „F.J.L.“ für die Erstellung des Prüfungsfragenkatalogs.