Algorithmen und Datenstrukturen
Projektarbeiten

Christian Heitzmann (heia@zhaw.ch)
Philippe Nahlik (xnah@zhaw.ch)

HS 2012/2013

1 Aufgabe und Lernziele

Ziel der Projektarbeiten ist es, dass Sie zusammen mit Ihrer Gruppe je ein Thema aus dem Bereich der fortgeschrittenen Algorithmen vertieft erarbeiten. Dabei handelt es sich um Themen, welche aus Zeitgründen nicht in einer Grundlagenvorlesung zu Algorithmen und Datenstrukturen behandelt werden können. Sie sollen mit dem im Laufe des Semesters erworbenen Grundlagenwissen weitestgehend selbständig die grundsätzlichen Ideen und Konzepte eines solchen fortgeschrittenen Algorithmus herausarbeiten und verstehen. Dabei entwerfen Sie auch ein eigenes Softwarebeispiel (in einer Programmierumgebung Ihrer Wahl), welches die Funktionsweise des Algorithmus anschaulich demonstriert.

Die Kernstücke des Algorithmus präsentieren Sie als Theorieteil, zusammen mit einem veranschaulichenden Beispiel als Praxisteil, am Ende des Semesters vor allen anderen Studenten. Dabei ist wichtig, dass es sich nicht um eine „Formelschlacht“ handelt, sondern um eine publikumsgerechte Aufbereitung und Darstellung der theoretischen und praktischen Inhalte. Durch das Vortragen erhalten Sie auch eine weitere wertvolle Übungsgelegenheit in Präsentations- und Vortragstechnik.

Ihre Mitstudenten im Publikum sollen im Rahmen der präsentierten Projektarbeiten nur einen ersten Einblick in diese Algorithmen erhalten. Das primäre Ziel für Ihre Zuhörer ist, dass sie über die Existenz und potentiellen Anwendungsgebiete solcher fortgeschrittenen Algorithmen zunächst nur Bescheid wissen. Bei einem konkreten Bedarf in der zukünftigen Informatiker-Laufbahn kann man sich im Idealfall daran erinnern, „bereits einmal etwas darüber gehört zu haben“, und sich konkret mit der notwendigen Literatur befassen, bevor man selbst Lösungen erfindet, für die es wahrscheinlich schon seit vielen Jahrzehnten fertig entwickelte Lösungen gibt.

2 Aufwand und Bewertung

Für das Bearbeiten des Projektarbeitsthemas ist für diese Lehrveranstaltung ein Arbeitsaufwand von ca. 15 Stunden pro Person vorgesehen. Dies kumuliert sich bei einer Gruppe von drei bis vier Studenten auf grob 45 – 60 Stunden für ein Thema, wodurch sich schon einiges an inhaltlicher Tiefe erreichen lässt. Je nach Thema und der Bereitschaft, für eine sehr gute Note auch ein paar Stunden mehr z. B. in eine eigene Beispielsoftware zu investieren, kann es sich bei den angegeben Zeiten natürlich nur um Richtwerte handeln.

Wir Dozenten stellen am Ende der Vorträge gerne noch ein bis zwei Fragen, um die wesentlichen Punkte noch einmal zu wiederholen oder bzgl. Ihres eigenen Wissens grob „auf den Zahn zu fühlen“. Es ist wichtig zu betonen, dass es sich dabei um keine „Diplomprüfung“ handelt, sondern um einen letzten Check, ob sich jemand normal auf den Vortrag vorbereitet und sich in der Zeit davor auch wirklich mit dem Projekt beschäftigt hat.

Ihre Vortragsfolien dienen als Handout für Ihre Mitstudentinnen und -studenten. Stellen Sie diese gemäss später noch bekannt zu gebenden Terminen den Dozenten zur Verfügung, damit diese rechtzeitig auf Moodle gestellt werden können. Darüber hinaus brauchen Sie *kein* Paper oder dergleichen zu verfassen. Ebenso brauchen Sie Ihre eigenen Quellcodes zu den Demonstrationsprogrammen nicht herauszugeben, es sei denn, Sie wünschen dies ausdrücklich.

Die Bearbeitung und Präsentation des Projektarbeitsthemas bildet 1/3 der Semesternote und zählt damit insgesamt zu 1/6 für die Kursnote. Es gibt durchaus Möglichkeiten, dabei eine ungenügende Note zu erreichen, z. B.

- … wenn der Eindruck entsteht, dass erst vor drei Tagen begonnen wurde, sich mit dem Thema zu beschäftigen.
- … wenn sich herausstellt, dass die Idee des Projektes darüber hinaus nicht verstanden wurde.
- … wenn der Vortrag überwiegend auf die Unterhaltung der Mitstudenten anstelle der Informationsvermittlung abzielt.
- … wenn sich die Inhalte nicht wesentlich von den Vorträgen der Vorgängerjahrgänge unterscheiden.
- … wenn Wikipedia die einzige „Quelle“ ist.
- … wenn im Sinne einer falsch verstandenen Arbeitsteilung nur diejenige Person den Vortrag hält, die sich mit dem Projekt davor gar nicht beschäftigt hat.
- …

Noch viel mehr Möglichkeiten gibt es allerdings, eine gute bis sehr gute Note zu erreichen, z. B.

+ ... wenn Sie eine eigene Software zu Demonstrationszwecken entwickeln.
+ ... wenn Sie selbstständig und umfassend Literatur recherchieren.
+ ... wenn Sie das Thema „auf den Punkt bringen“ und dementsprechend „verkaufen“ können.
+ ... wenn Sie einen professionellen Vortrag und saubere Folien / Handouts präsentieren.
+ ... wenn Sie im Wesentlichen die Vortragszeiten einhalten.
+ ... wenn Sie sich konstruktiv mit den Dozenten austauschen.
+ ...

Ein Wort zu Wikipedia

Mit *Wikipedia* hat sich in den letzten zehn Jahren für jedermann eine Plattform in Form einer Enzyklopädie erschlossen, die in Sachen Umfang und Aktualität unerreicht und darüber hinaus sogar in sehr verständlicher Sprache geschrieben ist. Wikipedia hat sich quer über alle Generationen als schnelles Nachschlagewerk und „erste Anlaufadresse“ für Wissensanfragen aller Art herausgestellt. Es wäre falsch, Wikipedia beim Verschaffen eines Überblicks zu einem Thema nicht heranziehen zu wollen.

Auf der anderen Seite muss man sich bewusst sein, dass die Artikel der Wikipedia von Personen „wie du und ich“ verfasst werden. Sobald die Themen sehr speziell werden (im Bereich der Algorithmen und Datenstrukturen kommen wir diesem Bereich schon sehr nahe), fällt auch die kollektive und demokratische Kontrollinstanz anderer Autoren weg. Sofern bei derartigen Themen die jeweiligen Aussagen nicht genau referenziert und damit überprüfbar sind, kann dort „alles“ stehen, ohne dass die Korrektheit auch nur ansatzweise gewährleistet ist. Fachliche Fehler können sich dann in Form von Unvollständigkeit, Uneinheitlichkeit, schlechtem Stil oder schlicht falschen Aussagen manifestieren.

Zu bedenken gibt, dass man Wikipedia heute vermehrt nur noch als *einzige* „Quelle“ heranzieht, seien dies Schüler an Kantonsschulen für Referatsthemen oder heute sogar Verlagshäuser. Unakzeptierbar ist es spätestens dann, wenn die Inhalte nicht einmal mehr aufbereitet werden.

Sowohl wir Dozenten als auch Sie Studenten brauchen uns keine Vorträge anhören, die wir alle selbst in einer ruhigen Minute auf Wikipedia nachlesen können. Bitte seien Sie sich daher bewusst, dass die Inhalte der Wikipedia per se keine Quelle darstellen, geschweige denn auch nur ansatzweise wissenschaftliche Relevanz hätten.

3 Vorgehen und Zeitplan

1. Stellen Sie Gruppen von jeweils drei bis vier Personen zusammen. Die Gruppen sollten gemischt sein und z. B. nicht aus reinen SoftwareentwicklerInnen, SupporterInnen, etc. bestehen.
2. Suchen Sie sich drei für Sie interessante Themen mit einer entsprechenden Priorisierung aus.
3. Senden Sie Herr Nahlik (`xnah@zhaw.ch`) per E-Mail die Namen der Projektmitglieder inklusive der priorisierten Liste mit den gewünschten Themen bis am **Freitag, 02.11.2012**.
4. Nach Erhalt der E-Mails werden Herr Nahlik und Herr Heitzmann die Themengebiete aufgrund Ihrer Priorisierung definitiv an die jeweiligen Gruppen verteilen.
5. Schicken Sie *beiden* Dozenten Ihr Konzept bis *spätestens* **Freitag, 28.12.2012**. Ihr Konzept muss einen klaren Überblick über die Themen und den Umfang Ihrer Präsentation verschaffen. Details wie Layout oder eine vollständig fertiggestellte Demonstrationssoftware sind natürlich noch nicht nötig.

   Die Einhaltung dieses Termines ist obligatorisch.
6. Die Vorträge finden am **Donnerstag, 24.01.2013** sowie am **Donnerstag, 31.01.2013** jeweils von 18:30 – 21:50 Uhr im Zimmer ZL O6.12 statt. Bitte notieren Sie sich diese Termine bereits, da diese präsenzpflichtig sind.
7. Bitte überprüfen Sie ausreichend vor der Schlusspräsentation, ob Ihr Computer mit dem Beamer funktioniert.
8. Die Dauer des Vortrages soll maximal 15 Minuten betragen. Zusätzlich kommen etwa 5 – 10 Minuten für Fragen oder Diskussionen hinzu.

4 Themen

4.1 Balancierte Bäume — Dozent: Christian Heitzmann

Ein grosses Problem binärer (Such-)Bäume ist die Gefahr der sogenannten Entartung. Werden einem binären Baum die Einträge in einer „falschen“ Reihenfolge zugefügt, so unterscheidet er sich kaum mehr von einer verketteten Liste, und die Datenstruktur verliert ihre eigentlichen Vorteile. In der Praxis werden daher häufig selbstbalancierende Bäume verwendet, die sich ohne Zutun des Aufrufers automatisch um eine gleichmässige Verteilung der Knoten des Baumes kümmern. In diesem Projekt geht es um die Vorstellung der verschiedenen Möglichkeiten und Algorithmen sowie Implementierung und Visualisierung einer solchen Datenstruktur.

4.2 Compiler-Optimierungen — Dozent: Christian Heitzmann

Der Compilerbau ist ein eigenständiges Gebiet in der Informatik und besteht im Wesentlichen aus einer Analysephase, die den Quelltext analysiert und daraus einen Syntaxbaum erzeugt und aus einer Synthesephase, die daraus das Zielprogramm erzeugt. Üblicherweise bietet der Compiler Optionen für verschiedene Optimierungen mit dem Ziel, die Laufzeit der einzelnen Programmschritte oder den Speicherplatzbedarf des Zielprogramms zu minimieren. Ziel ist es, sich in die Optimierungsmöglichkeiten wie „Einsparung von Maschinenbefehlen“, „Elimination von toten Programmcodes“ etc. einzuarbeiten und schliesslich ein Beispiel präsentieren zu können, welches eine dieser Optimierungsmöglichkeiten darstellt und erklärt.

4.3 Evolutionäre Algorithmen — Dozent: Philippe Nahlik

Evolutionäre Algorithmen werden vor allem für komplexe Optimierungsverfahren eingesetzt. Die Idee der Evolutionären Algorithmen stammt aus der biologischen Evolution. Mittels Mutation, Rekombination und Selektion nähert sich der Algorithmus dem Optimum an. Sie ermöglicht das Lösen komplexer Aufgaben mit erstaunlich einfachen Mitteln. Wünschenswert ist eine Darstellung, wie sich der Algorithmus Schritt für Schritt der gesuchten Lösung annähert.

4.4 Fehlerkorrigierende Codes — Dozent: Philippe Nahlik

Fehlerfreie digitale Datenübertragung ist ein Mythos: Die Frage lautet nicht, ob, sondern wie häufig ein Bit falsch beim Empfänger ankommt. Fehlerkorrigierende Codes verhindern, dass das zu oft passiert. Doch damit korrigiert werden kann, muss zuerst erkannt werden, wo der Fehler passiert ist. Die einfachste Art, Fehler in binären Nachrichten zu erkennen, sind Paritätskontrollen. Blöderweise lassen sich dadurch nur Einzelfehler erkennen . . . Erarbeiten Sie einen Überblick über die fehlerkorrigierenden und fehlererkennenden Codes und implementieren Sie ein interessantes Beispiel.

4.5 Fuzzy Logic — Dozent: Philippe Nahlik

Immer wieder benötigt man Vorbedingungen, die entweder wahr oder falsch sein können, und deren Aktionen, die je nach dem Wert der Vorbedingungen, ausgeführt werden oder nicht. Genau so formulieren viele menschliche Experten ihre Regeln oft unscharf (fuzzy), in denen nicht genau festgelegt ist, wann eine Vorbedingung erfüllt ist. Z. B. „Wenn die Temperatur um die 20 Grad Celsius beträgt, dann muss das Heizungsventil ein wenig geöffnet werden.“ Menschen sind in der Lage diese Regel zu verstehen und können danach handeln. Für Computer stellen jedoch unscharfe Begriffe ein Problem dar. Ab wann soll ein Expertensystem eine Temperatur als „ungefähr 20 Grad

Celsius“ interpretieren? Es hat sich gezeigt, dass viele komplexe Probleme durch die unscharfen Regeln schnell und einfach beschrieben werden können. Speziell in der Regelungstechnik feiert Fuzzy Logic Erfolge.

4.6 Garbage Collectors — Dozent: Philippe Nahlik

Der Garbage Collector ist ein Mechanismus der virtuellen Maschine, der die Aufgabe hat, nicht mehr benötigte Objekte aus dem Speicher zu entfernen. Er läuft im Hintergrund und sucht nach Objekten, die nicht mehr referenziert werden. Hierzu gibt es verschiedene Verfahren wie z. B. Mark-and-Sweep oder Mark-and-Compact, welcher zusätzlich die noch referenzierten Objekte herauskopiert und dadurch das Problem der Speicherfragmentierungen bekämpft.

4.7 Komprimierungen — Dozent: Christian Heitzmann

Bei den Komprimierungen geht es nebst der Unterscheidung zwischen verlustbehafteter und verlustfreier Komprimierung um die Vorstellung je eines konkreten Verfahrens beider Kompressionsprinzipien. Für die verlustbehaftete Komprimierung bieten sich Audiodaten (z. B. MP3) oder Bilddaten (z. B. JPEG) an, für die verlustfreie Komprimierung Verfahren wie Lauflängencodierung, Huffman-Codierung oder Lempel-Ziv-Codierung (z. B. ZIP). Eine Implementierung „from scratch“ wird höchstwahrscheinlich nicht möglich sein, aber Matlab enthält bereits viele Grundoperationen (z. B. die Fouriertransformation), um die Funktionalität der verlustbehafteten Komprimierung simulieren zu können.

4.8 Künstliche neuronale Netze — Dozent: Philippe Nahlik

Bei den KNNs liegt das „Wissen“ über den Vorgang im „Haufwerk“ der synaptischen Gewichtsfaktoren des Netzwerks fixiert. Das Netzwerksystem erwirbt diese Gewichtsfaktoren durch Training, d. h. durch langwieriges Lernen, Lernen und abermals Lernen von Datensätzen real gemessener Zusammenhänge. KNNs erarbeiten ihr Wissen durch Erlernen, ähnlich wie wir Menschen beispielsweise eine Fremdsprache erlernen. Ziel ist es, dass Sie ein kleines Neuronales Netzwerk erstellen und allenfalls demonstrieren, wie das KNN durch Training bessere Resultate erzielt.

4.9 Kürzeste Pfade — Dozent: Christian Heitzmann

Bei den kürzesten Pfaden geht es um eine relativ konkrete Beschreibung zweier Algorithmen, welche in gewichteten Graphen denjenigen Pfad bestimmen, welcher die minimalen Kosten hat. Die beiden Algorithmen von Dijkstra und Bellman-Ford finden

z. B. intensiv Anwendung im Bereich des Netzwerk-Routings oder in Routenplanern. Aufgrund des fachlich eher kleinen Umfangs wird erhöhter Wert auf die theoretische Beschreibung der Algorithmen und der Transfer in Praxissituationen gelegt.

4.10 Parallele Programmierung — Dozent: Christian Heitzmann

In der heutigen Zeit, in der sogar Smartphones über mehrere Prozessorkerne verfügen, gewinnt die Softwareentwicklung für Mehrprozessorsysteme zunehmend an Bedeutung. Aber nicht jedes Programm lässt sich einfach durch n teilen und dadurch n-fach schneller parallel ausführen. Jedes Programm verfügt über einen unvermeidbaren sequenziellen Teil, und auch diejenigen Abschnitte, die sich auf mehrere Prozessoren aufteilen lassen, müssen streng miteinander synchronisiert werden. In diesem Projekt sollen Sie die Möglichkeiten und Schwierigkeiten der parallelen Programmierung aufzeigen. Eine Demonstration könnte z. B. darin bestehen, ein gängiges Problem zu parallelisieren und die tatsächlichen Laufzeitunterschiede in der Praxis festzustellen.

4.11 Physikalische Simulationen — Dozent: Christian Heitzmann

Physikalische Simulationen erlauben es, auf relativ einfache Art und Weise komplex anmutende Prozesse aus der wirklichen Welt auf dem Computer zu simulieren. Grundlagen hierzu bilden beispielsweise Masse-Feder-Systeme, welche alle zu simulierenden Objekte virtuell mit Federn verbindet, die dann mit jedem Simulationsschritt anhand physikalischer Gesetze ausgewertet werden, um dessen neue Positionen und Geschwindigkeiten zu bestimmen. Eine weitere einfache Simulationsmöglichkeit besteht in Partikelsystemen, wo in Kombination mit Gravitation, Dämpfung und externen Kraftfeldern viele chaotisch erscheinende Prozesse nachgebildet werden können. Eine Implementierung mit grafischer Ausgabe (2D oder 3D) ist erwünscht.

4.12 Randomisierte Algorithmen — Dozent: Christian Heitzmann

Randomisierte Algorithmen sind Verfahren, in denen mit Hilfe von zufälligen Ablaufsteuerungen versucht wird, Probleme, die sich deterministisch nur schwer lösen lassen, zeitlich wesentlich effizienter zu lösen. Dabei nimmt man in Kauf, dass die Ergebnisse nicht ganz der optimalen Lösung entsprechen, sondern die Resultate auch mehr oder weniger falsch sein können. Die Wahrscheinlichkeit, mit der grobe Fehler passieren, ist im Verhältnis zu der Zeitersparnis, die man erreichen kann, aber meist so vernachlässigbar klein, dass sich randomisierte Algorithmen als ein unabdingbares Werkzeug erwiesen haben, algorithmisch schwere Probleme anzugehen und zu lösen. Inhalt dieses Themas sind vor allem die Grundideen von randomisierten Algorithmen und einige wenige Fallbeispiele.

4.13 Reguläre Ausdrücke — Dozent: Philippe Nahlik

Ein regulärer Ausdruck beschreibt eine oder mehrere Zeichenfolgen, für die in einem Textkörper beim Durchsuchen eine Übereinstimmung gefunden werden soll. Der Ausdruck dient als Zeichenmuster, das mit dem gesuchten Text verglichen wird. Mithilfe von regulären Ausdrücken können Sie in einer Zeichenfolge nach Mustern suchen, Text ersetzen und Teilzeichenfolgen extrahieren. So können Sie unter anderem ein Muster innerhalb einer Zeichenfolge suchen oder z. B. eine Eingabezeichenfolge dahingehend testen, ob darin ein Muster für eine Telefonnummer oder eine Kreditkartennummer vorkommt. Dies entspricht der Datenvalidierung. Als Beispiel: Angenommen, Sie müssten eine gesamte Website durchsuchen, veraltetes Material entfernen und diverse HTML-Formatierungstags ersetzen. In diesem Fall können Sie anhand eines regulären Ausdrucks feststellen, ob das Material oder die HTML-Formatierungstags in jeder Datei vorkommen. Dadurch wird die Liste der betroffenen Dateien auf die Dateien eingegrenzt, die Material enthalten, das zu entfernen oder zu ändern ist. Anschliessend können Sie das veraltete Material mit einem regulären Ausdruck entfernen. Und schliesslich können Sie einen regulären Ausdruck verwenden, um die Tags zu suchen und zu ersetzen.

4.14 Eigene Themen — Dozent: Ihre Wahl ;-)

Sollten Sie und Ihre Gruppe eigene Wünsche oder Ideen zu Themen der fortgeschrittenen Algorithmen haben, die vom Umfang her zu den oben vorgestellten Themen passen, dürfen Sie uns diese gerne mitteilen.