Lehrstuhl für Algorith.
und Datenstrukturen
Prof. Dr. Hannah Bast

Algorithmen und Datenstrukturen
(ESE) WS 2010 / 2011

http://ad-wiki.informatik.uni-freiburg.de/teaching

UNI FREIBURG

Klausur

Mittwoch 30. März 2011, 14:00 - 16:30 Uhr

Es gibt 5 Aufgaben. Für jede Aufgabe gibt es maximal 10 Punkte. Sie haben insgesamt 2 1/2 Stunden Zeit. Das sind pro Aufgabe im Durchschnitt eine halbe Stunde.

Sie dürfen eine beliebige Menge an Papier, Büchern, etc. verwenden. Sie dürfen keinerlei elektronische Geräte wie Notebook, Mobiltelefon, etc. verwenden, insbesondere keine Geräte, mit denen Sie mit Dritten kommunizieren oder sich mit dem Internet verbinden können.

Schreiben Sie bitte auf jedes Blatt Papier, das Sie abgeben, oben gut lesbar Ihren Namen (in Blockbuchstaben bitte) und Ihre Matrikelnummer, und numerieren Sie die Blätter vor der Abgabe bitte durch.

Wir wünschen Ihnen viel Erfolg!

Aufgabe 1 (Dynamische Felder, 10 Punkte)

In dieser Aufgabe geht es um die dynamischen Felder aus der 1. Vorlesung. Sei s_i die Größe (size) und c_i die Kapazität (capacity) des Feldes nach der i-ten Operation, und seien s_0 und c_0 die Größe und die Kapazität am Anfang, vor der ersten Operation auf dem Feld.

Nehmen wir allgemein an, dass wenn die i-te Operation ein Element am Ende hinzufügt (*push_back*) und $c_{i-1} < s_i$, dann ist $c_i = A \cdot s_i$, und dass, wenn die i-te Operation das letzte Element entfernt (*pop_back*) und $s_i \leq c_{i-1}/B$, dann $c_i = \lceil c_{i-1}/C \rceil$, und in allen anderen Fällen $c_i = c_{i-1}$.

1.1 (3 Punkte) Nehmen wir an, dass $s_0 = c_0 = 0$ und $A = 2$, $B = 3$ und $C = 2$. Nehmen wir dann an, dass die folgenden *zehn* Operationen hintereinander ausgeführt werden (angefangen beim leeren Feld):

push_back(1), *push_back(2)*, *push_back(3)*, *push_back(4)*, *push_back(5)*,
pop_back(), *pop_back()*, *pop_back()*, *pop_back()*, *pop_back()*.

Geben Sie den Zustand des Feldes (enthaltene Elemente, Größe und Kapazität) nach jeder dieser Operationen an.

1.2 (3 Punkte) Betrachten Sie jetzt, für dieselben Werte von s_0, c_0, A, B und C wie in Aufgabe 1.1, eine beliebige Folge von n *push_back* Operationen. Bei welchen Operationen wird die Kapazität des Feldes verändert, d.h. für welche i gilt $c_i > c_{i-1}$. Geben Sie eine Formel an, mit der man alle diese i erhält, sowie die zugehörigen Werte von s_i und c_i. Beweisen Sie dann Ihre Formel mit vollständiger Induktion.

1.3 (1 Punkt) Falls $A > 1$ und $B > 1$ und $C < B$ gilt, dass die Kosten einer beliebigen Folge von n Operationen (angefangen beim leeren Feld) in $O(n)$ liegen. Kann man eine dieser drei Bedingungen abschwächen, d.h. gilt diese Aussage auch dann noch wenn $A = 1$ oder wenn $B = 1$ oder wenn $C = B$? Mit Begründung bitte.

1.4 (3 Punkte) Nennen Sie einen Vorteil und einen Nachteil davon, A groß zu wählen. Nennen Sie einen Vorteil und einen Nachteil davon, B groß zu wählen. Nennen Sie einen Vorteil und einen Nachteil davon, B/C groß zu wählen.

Aufgabe 2 (O-Notation und IO-Effizienz, 10 Punkte)

2.1 (3 Punkte) Wir hatten die O-Notation so definiert, dass für zwei Funktionen $f, g : \mathbb{N} \to \mathbb{R}$ gilt dass $f = O(g)$, wenn es zwei Konstanten n_0 und C gibt, so dass für alle $n \geq n_0$ gilt, dass $f(n) \leq C \cdot g(n)$, und analog für Ω und Θ. Zeigen Sie mit Hilfe dieser Definition (ohne Verwendung des Grenzwertbegriffes), dass gilt:

$$n^2 + 3n - 1 = \Theta(n^2).$$

2.2 (3 Punkte) Geben Sie für jede der folgenden Aussagen an, ob Sie richtig oder falsch ist, mit Begründung über den Grenzwert:

$$(\log n)^5 = O(\sqrt{n})$$

$$\sqrt{n} = \Omega(n)$$

$$n/1000 = \Theta(n)$$

2.3 (2 Punkte) Betrachten Sie folgende Funktion zur zufälligen Permutation der Elemente eines gegebenen Feldes:

```
void permute(std::vector<int>& A)
{
  size_t n = A.size();
  for (size_t i = 0; i < n; ++i)
  {
    size_t j = i + drand48() * (n - i);  // Random number from the interval [i..n-1].
    std::swap(A[i], A[j]);  // Swap the entries at positions i and j.
  }
}
```

Bestimmen Sie die Laufzeit dieser Funktion (in Θ-Notation) im schlechtesten Fall und im besten Fall, in Abhängigkeit von der Größe n des gegebenen Feldes. Sie können dabei annehmen, dass jeder Aufruf der Funktion *drand48* und der Funktion *std::swap* die Kosten $\Theta(1)$ hat.

2.4 (2 Punkte) Bestimmen Sie die Anzahl Blocktransfers der Funktion aus der vorherigen Aufgabe 2.3 (ebenfalls in Θ-Notation) im schlechtesten Fall und im besten Fall, in Abhängigkeit von der Größe n des gegebenen Feldes und der Blockgröße B.

Aufgabe 3 (Hashtabellen, 10 Punkte)

Wir hatten in der Vorlesungen Hashtabellen vom Typ *std::vector<std::vector<std::pair<KeyType, ValueType>> >* betrachtet, wobei *KeyType* der Typ des Schlüssels und *ValueType* der Typ des Wertes eines Elementes war.

3.1 (2 Punkte) Sei $h(x) = (2 \cdot x) \bmod 5$ unsere Hashfunktion und habe entsprechend unsere Hashtabelle die Größe 5. Betrachten Sie die folgenden Operationen (beginnend mit einer leeren Hashtabelle):

insert(1), insert(6), insert(2), insert(4), remove(1), insert(21)

Geben Sie den Zustand der Hashtabelle nach jeder dieser Operationen an.

3.2 (2 Punkte) Geben Sie, für dieselbe Hashfunktion wie in Aufgabe 3.1 und für ein beliebig gegebenes n, eine Folge von n *insert*-Operationen und eine darauf folgende *lookup*-Operation an, so dass die Kosten dieser *lookup*-Operation so groß wie möglich sind.

3.3 (2 Punkte) Bestimmen Sie die Wahrscheinlichkeit, dass zwei zufällig und voneinander unabhängig gewählte Schlüssel von der Hashfunktion aus Aufgabe 3.1 auf denselben Wert abgebildet werden. Begründen Sie Ihre Berechnung und schreiben Sie nicht einfach nur das Endergebnis hin.

3.4 (2 Punkte) Bestimmen Sie den Erwartungswert der Anzahl der Elemente, die bei der Einfügung von n zufälligen und voneinander unabhängig gewählten Schlüssel auf den Wert i der Hashtabelle aus Aufgabe 3.1 abgebildet werden, für ein gegebenes $i \in \{0, \ldots, 4\}$. Begründen Sie Ihre Berechnung und schreiben Sie nicht einfach nur das Endergebnis hin.

3.5 (2 Punkte) Wie ändert sich das Ergebnis von Aufgabe 3.3 und 3.4, wenn als Hashfunktion $h(x) = (8 \cdot x) \bmod 5$ genommen wird? Wie ändert sich das Ergebnis der beiden Aufgaben, wenn als Hashfunktion $h(x) = (10 \cdot x) \bmod 5$ genommen wird? Es reicht jeweils eine kurze Begründung.

Aufgabe 4 (Dijkstra's Algorithmus, 10 Punkte)

Sei $G = (V, E)$ der gerichtete Graph mit Knotenmenge $\{A, B, C, D, E\}$ und acht gewichteten Kanten, die durch die folgenden Tripel gegeben sind: $(A, B, 8)$, $(A, C, 2)$, $(A, D, 4)$, $(A, E, 5)$, $(C, D, 1)$, $(D, B, 4)$, $(D, E, 1)$, und $(E, B, 1)$. Für jedes Tripel ist dabei die erste Komponente der Startknoten der Kante, die zweite Komponente der Endknoten der Kante und die dritte Komponente das Gewicht der Kante.

4.1 (1 Punkt) Zeichnen Sie diesen Graphen möglichst übersichtlich, insbesondere mit genügend Abstand zwischen den einzelnen Knoten.

4.2 (2 Punkte) Führen Sie Dijkstra's Algorithmus auf diesem Graphen aus, mit Startknoten A und Zielknoten B. Veranschaulichen Sie dabei den Fortgang des Algorithmus anhand eines einzigen Bildes des Graphen. Aus Ihrem Bild sollte ersichtlich sein, in welcher Reihenfolge die verschiedenen Schritte ausgeführt wurden. Insbesondere sollte ersichtlich werden, wieviele Schritte benötigt wurden, und was die Kosten des kürzesten Weges von A nach B sind.

4.3 (2 Punkte) Geben Sie ein Beispiel eines Graphen mit mindestens einem negativem Kantengewicht, für den Dijkstra's Algorithmus *nicht* das richtige Ergebnis berechnet, und erklären Sie, warum das der Fall ist.

4.4 (2 Punkte) Geben Sie ein Beispiel eines Graphen mit mindestens einem negativen Kantengewicht, für den Dijkstra's Algorithmus *trotzdem* das richtige Ergebis berechnet, und erklären Sie, warum das der Fall ist.

4.5 (3 Punkte) Bestimmen Sie für jeden Knoten X aus dem Graph aus Aufgabe 4.1 die Kosten $c(X, B)$ eines billigsten Weges zum Zielknoten B. Führen Sie dann den A^* Algorithmus auf diesem Graphen aus, wieder mit Startknoten A und Zielknoten B, wobei Sie als Wert für die Heuristikfunktion h für einen Knoten X gerade $c(X, B)$ nehmen sollen.

Aufgabe 5 (String Matching, 10 Punkte)

Gegeben folgender Text T und folgendes Muster P:

```
T = ANAPANAMAMANANAPANAPANA
P = ANAPANA
```

5.1 (3 Punkte) Führen Sie den Knuth-Morris-Pratt (KMP) Algorithmus auf dieser Eingabe aus und schreiben Sie die Ausführung so nieder, dass sich die einzelnen Schritte des Algorithmus nachvollziehen lassen.

Wir betrachten jetzt einen neuen Algorithmus. Wie beim KMP Algorithmus wird dabei das Muster von links nach rechts über den Text geschoben und dabei an ausgewählten Stellen Übereinstimmung geprüft. Stimmt das Muster an einer solche Stelle mit dem Text überein, wird ein Treffer ausgegeben und das Muster um eins nach rechts verschoben. Stimmt das Muster nicht überein und kommen alle im Text nicht übereinstimmenden Buchstaben im Muster vor, wird das Muster ebenfalls um eins nach rechts verschoben. Stimmt das Muster nicht überein und kommt einer der im Text nicht übereinstimmenden Buchstaben im Muster nirgends vor, wird das Muster hinter den am weitesten rechts stehenden solchen Buchstaben im Text verschoben. Hier die ersten Schritte am obigen Beispiel:

```
ANAPANAMAMANANAPANAPANA
```

`ANAPANA`	Treffer, Muster wird um eins nach rechts verschoben.
` ANAPANA`	Kein Treffer, `T[8] = M` kommt im Muster nicht vor, Muster wird nach Position 9 verschoben.
`  ANAPANA`	Kein Treffer, `T[10] = M` kommt im Muster nicht vor, Muster wird nach Position 11 verschoben.
`   ANAPANA`	Kein Treffer, alle Buchstaben im Text kommen im Muster vor, Muster wird um eins nach rechts verschoben.
`    ANAPANA`	Usw.

5.2 (3 Punkte) Schreiben Sie eine Funktion *findMatches*, in Java oder in C++, die diesen Algorithmus ausführt.

5.3 (2 Punkte) Geben Sie eine stichhaltige Begründung, warum der Algorithmus korrekt ist.

5.4 (2 Punkte) Geben Sie ein Beispiel für einen Text einer gegebenen Länge n und für ein Muster einer gegebenen Länge m, für die der Algorithmus seine bestmögliche Laufzeit erreicht, sowie ein Beispiel auf dem der Algorithmus seine schlechtestmögliche Laufzeit erreicht.