Technische Universität München
Fakultät für Informatik
Prof. R. Bayer, Ph.D.,
Dipl.-Inform. Michael Bauer
Dr. Gabi Höfling

WS 2003/04,
Datenbanksysteme I,
Übungsblatt 1-Lösungen

27.10. 2003

1.In einem ersten Schritt sollen die Anzahl der Einträge in der verwendeten Datenstruktur und der dafür benötigte Speicher abgeschätzt werden. Beschreiben Sie bei der Berechnung des Speicherbedarfs die pro Eintrag benötigten Daten, verwenden Sie Standarddatentypen und ignorieren Sie den Overhead eventueller Hilfsstrukturen wie Listen, Bäume usw.

Einträge:

1.Verfahren: 2 * |Bahnhof| *|Züge pro Tag am Bahnhof | ≈ $3 * 10^5$

2. Verfahren: |Bahnhof|*|Bahnhof-1|*|Verbindungen pro Tag| ≈ $1{,}4 * 10^9$

3. Verfahren: |Bahnhof|*|Züge pro Tag am Bahnhof| ≈ $1{,}5 * 10^5$

Speicher:

Pro Eintrag:

1. *Verfahren*: Abfahrtszeit, (Bahnhof, Ankunftszeit)*, analog Ankunftszeit = 4+12*(2+4) = 76 Bytes
2. *Verfahren*: Abfahrtszeit, Ankunftszeit, Zug, (Umsteigebahnhof, Umsteigezeit, Umsteigezug)* = 4+4+2 + 2*(2+4+2)= 26 Byte
3. *Verfahren*: Start, Ziel, Abfahrtszeit, Ankunftszeit = 2+2+4+4 = 12 Bytes

Zu den Datentypen: Zug und Bahnhof sind jeweils Nummern (16 bit), Zeit ist ein UNIX-Timestamp (32 bit).

Gesamtspeicher (ohne Zugriffsstrukturen):

1. *Verfahren*: 22 MByte
2. *Verfahren*: 34,6 GByte
3. *Verfahren*: 1,73 MByte

2.Im nächsten Schritt soll der Aufwand zur Suche einer einzelnen schnellsten Verbindung zu einem bestimmten Zeitpunkt abgeschätzt werden. Überlegen Sie sich dazu, wie die Verfahren auf den einzelnen Datenstrukturen arbeiten und schätzen die Anzahl der Schritte ab. Falls Sie einen Standardalgorithmus verwenden, kann die O-Notation verwendet werden.

2. Verfahren:
Lookup Konstant, aber Suche nach Umsteigebahnhöfen und Umsteigezeiten innerhalb der Liste notwendig:

Bei Hashing ebenfalls konstant, aber Speicheroverhead/Kollisionen/ teure Hashfunktion.

Bei Baumsuche logarithmisch, z.B. Binärbaum $\log_2$ |Einträge| ≈ 26 Schritte bis Bahnhof und Ziel, danach linerare oder binäre Suche über Einzelverbindungen

Spezialfall: Auslagern der Daten auf Platte und erneutes Einlesen:

Pro Bahnhof etwa 5 MB, pro Bahnhof und Ziel etwa 650 Byte, gute Indexstrukturen und Bufferverwaltung wichtig.

1. und 3. Verfahren:

Nach Dijkstra: O(Kanten+Knoten log Knoten) =>
O(|Verbindungen|+|Bahnhof| log |Bahnhof|) =>
O(|Züge pro Bahnhof|*|Bahnhof|+|Bahnhof| log |Bahnhof|)
Zur Abschätzung direktes Ausmultiplizieren, Logarithmus zur Basis 10
≈ 180.000 Schritte

Optimierung: Spezialfall für 1. Verfahren: Direktverbindung (auf selbem Zug):
Stops/2*log(Züge pro Bahnhof) < 100 Schritte

3.Neben der Suche nach Verbindungen müssen die Daten bei Änderungen im Fahrplan gepflegt werden, z.B. verändert sich die Fahrzeit durch die Eröffnung einer Neubaustrecke. Beurteilen Sie die Verfahren im Bezug auf Updates:

a) Wie sehen jeweils die Verfahren für Updates aus

3. Verfahren: Änderung der Kantengewichte und/oder Kanten einfügen/entfernen

1. Verfahren: Ausgehend von den Anfangs/Endpunkten der geänderten Direktverbindungen eines durchgehenden Zuges müssen die „Reststrecken“ der betroffenen Züge angepasst werden. Die Zeiten können an einem Bahnhof durch Addition bzw. Subtraktion durchgeführt werden und müssen dann auf allen restlichen Bahnhöfen eingetragen werden. Diese können über die Züge gefunden werden.

2. Verfahren: Wenn keine weiteren Hilfsstrukturen verwendet werden (Zugliste), muss die gesamte Verbindungsliste durchsucht werden, ob ein darin verwendeter Zug eine der neuen Verbindungen verwendet. Falls dies der Fall ist, müssen für jede dieser Verbindungen die Zeiten angepasst werden. Falls kein Zugwechsel stattfindet, ist eine einfache Subtraktion bzw. Addition ausreichend, andernfalls muss ein Suchverfahren ähnlich zu 3 oder 1 gestartet werden, um die nunmehr beste Verbindung zu suchen.

b) Wo können Konsistenzprobleme auftreten

3. Verfahren: problemlos (da wir Folgeänderungen der Fahrzeiten als zulässig und problemlos ansehen)

1. Verfahren:

- Teilstrecken nicht neu berechnet: z.B. alte Fahrzeiten nach der geänderten Strecke (bei allen Stationen) => Fehlerhafte Fahrzeiten ab einem bestimmten Punkt, als Folge auch falsche Umsteigezeiten für Anschlusszüge
- Änderungen nicht bei allen Bahnhöfen durchgeführt. (Alte Fahrzeiten bei bestimmten Bahnhöfen)

2. Verfahren:

- Verbindung beim Update übersehen: alte Fahrzeiten, evtl. nicht vorhandene Anschlusszüge
- Fehler bei Berechnung der neuen Verbindung
  - Addition/Subtraktion vergessen, falsche Restzeiten/Startzeiten
  - Fehler bei Suche nach alternativen Restverbindungen: suboptimale Verbindungen, Sprungstellen bei Verbindungsdauer

4. Welche Änderungen sind notwendig, falls nach anderen Kriterien als der Verbindungsdauer gesucht wird? Betrachten Sie dazu die Suche nach billigsten Verbindungen und Verbindungen mit einer möglichst geringen Anzahl von Zugwechseln.

3. Verfahren: Einfügen zusätzlicher Kantengewichte (Preis), „Strafgewichte“ für Zugwechsel und Markierung der Kanten nach Zügen,
Festlegung des relevanten Gewichts bei der Anfrage.

1. Verfahren:

- Preise: Annotation der Preise zusätzlich zu den Zügen. Dies muss pro Teilstrecke geschehen, wodurch ein Verfahren ähnlich wie 3 verwendet werden kann. Falls sich die Preise der einzelnen Teilstrecken einfach aufaddieren lassen, kann auch der Preis zu Teilzielen angegeben werden.
- Zugwechsel: Ähnlich wie beim 3. Verfahren, Suchalgorithmus modifizieren um möglichst lange beim selben Zug zu bleiben.

2. Verfahren: Anlegen einer neuen Verbindungstabelle für jedes gewünschten Kriterium

5. Überlegen Sie, welche Verfahren man wohl in der „Realität“ einsetzt.

Ausnutzen der Topologie, Heuristiken, $\alpha\beta$-Suche, branch and bound