A-Sort

Algorithmus

Idee/Vorgehen

ARRAY von vorne nach hinten durcharbeiten
- neue Zahl immer zuhinterst hinstellen
- soweit nach vorne rücken, bis richtiger Platz erreicht (= um # Inversionen vorschieben)

Summe aller Abstände jeweils vom rechten Rand = # Inversionen der Ausgangsfolge

Laufzeit/Lösungsgüte

Thomas Briner, brinerth@iiic.ethz.ch

Abstrakter Datentyp

Theorie

Idee/Vorgehen

Der ADT ist eine Hülle, die vom Server bereitgestellt wird mit genau definierten Schnittstellen.

Der Benutzer erweitert den ADT um die Inhalte, die er braucht, liefert ev. weitere Prozeduren (Vergleich etc.).
Er kümmert sich aber nicht um die eigentlichen Vorgänge mit seinen Daten sondern nur um die Ergebnisse.

Der Client hat nur auf die exportierten Prozeduren und Objekte Zugriff

Laufzeit/Lösungsgüte

Thomas Briner, brinerth@iiic.ethz.ch

Additionsrätsel

Algorithmus

Idee/Vorgehen

ARRAY mit n Ziffern,
6 9 2 5 0 2 1 3 --> 69 + 2 + 5 + 0 + 21 + 3 = 100
Wie zusammennehmen/plus setzen, so dass Summe = 100?

```
PROCEDURE Add(i, sum, aktsum)
        BEGIN
        IF (i=n) OR  (sum+aktsum>100) THEN
                RETURN sum+aktsum=100;
        ELSE RETURN Add(i+1, sum+aktsum, A[i])
                 OR Add(i+1, sum, aktsum*10+A[i]);
        END;
```

Variante dynamisch:

O(n^3)

Laufzeit/Lösungsgüte

Thomas Briner, brinerth@iiic.ethz.ch

Aktienaufgabe, sooft wie man will einsteigen

Algorithmus

Idee/Vorgehen

immer mit Nachfolgetag vergleichen
falls A[i+1] höher als A[i] dann kaufen/behalten, sonst verkaufen

Laufzeit/Lösungsgüte O(n)

Thomas Briner, brinerth@iiic.ethz.ch

Algorithmenanalyse, Oh, Theta, Omega

Theorie

Idee/Vorgehen

O: Es gibt eine obere Grenze, einen möglichst kleine Fkt g(n), für die gilt:
$f(n) <= c1*g(n) + c2$
oder
$f(n) <= c*g(n)$ für alle $n >= n_0$

Omega: Es gibt eine möglichst grosse untere Grenze mit Funktion h(n):
h(n) e O(f(n))

Theta: Der Algorithmus hat die gleiche Fkt als obere und untere Schranke

Theta = O & Omega

Vorsicht: Diese Zusammenfassung ist nicht mit Sicherheit richtig!

Laufzeit/Lösungsgüte

Thomas Briner, brinerth@iiic.ethz.ch

Amortisierte Kostenanalyse (Bankkontenparadigma)

Theorie

Idee/Vorgehen

Z.B: Multidq in Q:

Bei jedem enQ wird zusätzlich zu effektiven Kosten (=1 pro enQ) noch ein 2. Franken einbezahlt. Damit wird das deQ und multidQ gratis

bal i : Kontostand nach i-ter Operation, balance
bal 0:=0
bal ist gleich der # der Elemente in Q

zB. Zähler 010001010001001:

Bei jedem Bit „1“ muss 1 Franken bereit liegen, weil ein Wechsel an dieser Stelle einen weiteren Wechsel an der vorderen Stelle nach sich zieht. Bei jedem Weiterzählen werden also 2 Franken bezahlt (einer für Konto)

a: amortisierte Kosten
t: tatsächliche Kosten

a i = t i + (bal i - bal i-1)
(Stand Konto vorher nachher)

Für alle Operationen aufgerechnet:

sum(a i) = sum(t i) + sum (bal i - bal i-1)
= sum(t i) + (bal Schluss - bal iAnfang)

--> sum(a i) >= sum(t i)

Vorsicht: Diese Zusammenfassung ist nicht mit Sicherheit richtig!

Laufzeit/Lösungsgüte

Thomas Briner, brinerth@iiic.ethz.ch

AVL-Bäume

Adelson-Velskij, Landis

Datenstruktur

Idee/Vorgehen

Für jeden Knoten gilt:
Differenz Höhe linker TB und rechter TB <=1
(-1: links höher, +1: rechts höher)
--> Jeder Knoten führt noch einen Parameter Balance mit

Bei Einfügen, Entfernen sind ev. Rotationen nötig
(Einfach bzw. Doppelrotationen)

Rotationen: Einfügen: <=1
Entfernen: <= log(n) schlääääächt!

Anzahl Blätter möglichst klein, # Blätter=Fib (H+2)

Höhe ~1.44 * log (n)

Vorsicht: Diese Zusammenfassung ist nicht mit Sicherheit richtig!

Laufzeit/Lösungsgüte Suche O(log n); Ins O(n); Del O(log n)

Thomas Briner, brinerth@iiic.ethz.ch

B-Bäume

Bayer, Mc Creight

Datenstruktur

Idee/Vorgehen

Viel Zeit wird verbraucht für Festplattenzugriff, wenn benachbarte Knoten an verschiedenen Orten gespeichert sind. --> viele Knoten zu einem Knoten zusammenfassen.

Effizienzmass: # Plattenzugriffe statt # Schlüsselvgl

B-Baum der Ordnung 3:

- alle Blätter haben gleiches Niveau
- jeder Knoten ausser Wurzel hat mind. m/2 Kinder, m/2-1 keys

- Wurzel: >=2 Kinder

Blätter bei Höhe H (= # keys +1)

	mind	höchstens
	2.(m/2)^h	m^h

--> n<= (m/2) -Logartihmus von ((n+1)/2)

Speicherplatznutzung >=50%, Im Mittel 69%

Was ist Differenz zu 2-3 Baum?

Vorsicht: Diese Zusammenfassung ist nicht mit Sicherheit richtig!

Laufzeit/Lösungsgüte alles $O(\log_m (n))$

Thomas Briner, brinerth@iiic.ethz.ch

Backtracking n Damen

Algorithmus

Idee/Vorgehen

Allgemeiner Backtracking Algorithmus

```
ok:=FALSE;

PROCEDURE FindeLsg(i: Index, VAR ok: BOOL, VAR L: Lösung);
BEGIN
        WHILE ~ok DO
         IF Teillösung bisher ist gültig THEN
                erweitere um einen Schritt
                IF Lsg noch nicht vollständig
                FindeLsg(i+1, ok, L);
                        IF ~ok THEN Backtracking
                        nimm den letzten Schritt
zurück
                END;
                ELSE ok:=TRUE;
        END;
        Wähle nächstes Element zum Ausprobieren
END:
```

Vorsicht: Diese Zusammenfassung ist nicht mit Sicherheit richtig!

Laufzeit/Lösungsgüte

Thomas Briner, brinerth@iiic.ethz.ch

Bellmannsche Optimalitätsbedingung

Theorie

Idee/Vorgehen

Wenn Gesamtlösung optimal ist, muss auch jede Teillösung optimal sein.

Aus diesem Grund lassen sich die Probleme mit der dynamischen Programmierung lösen, weil dort immer für kleine Teilbereiche eine optimale Lösung gesucht wird.

Laufzeit/Lösungsgüte

Thomas Briner, brinerth@iiic.ethz.ch

Bin-Packing-Problem

Algorithmus

Idee/Vorgehen

online Algorithmus, Behälter (Bins), alle gleich gross, es kommen immer ein Gegenstand nach dem andern, der verpackt werden muss.

Wenn er noch Platz hat, kommt er in den gleichen, wenn nicht, kommt er in den nächsten, in dem er Platz hat

Lösungsgüte = 2

Es gibt höchstens 1 Bin, der weniger als halbvoll ist (sonst hätte ich ja den Ggstand im 2. halbvollen zum
1. halbvollen reingetan)

Laufzeit/Lösungsgüte

Thomas Briner, brinerth@iiic.ethz.ch

Binäre Suche

Algorithmus

Idee/Vorgehen

In geordnetem ARRAY:

Es wird ein bestimmtes Element zB Mittelelement m ausgewählt, mit key verglichen.

Falls key>m, in rechter, sonst in linker Hälfte weitersuchen.

Bei IF Abfrage nicht zuerst x=m fragen!

Laufzeit/Lösungsgüte O(log n)

Thomas Briner, brinerth@iiic.ethz.ch

Bipartites perfektes Matching

Algorithmus

Idee/Vorgehen

Männer, Frauen, Kante bedeutet: will heiraten

online Algorithmus:

Alle Männer schon gegeben, Frauen kommen einzeln dazu

Wähle die erste Kante zu einem noch freien Mann

ergibt eventuell kein perfektes (maximum) Matching,

aber immerhin ein maximal Matching, d.h. es ist keine Verbesserung des Resultats durch hinzufügen von Kanten möglich

Laufzeit/Lösungsgüte

Thomas Briner, brinerth@iiic.ethz.ch

Blumenschaufenster

Algorithmus

Idee/Vorgehen

Es hat n Vasen, die fest positioniert sind, dazu kommen n Blumen, m<=n. Zu jeder Kombination Vase und Blume gibt es einen Wert. Welches ist die optimale Anordnung?

Dynamisch:
Senkrecht Blumen, waagrecht Vasen
Immer 1. Blume einer Reihe: Diagonal von oben übertragen und W[i,j] zum Wert dazuaddieren.
Für den Rest der Zeile: Max(Feld links (Blume nicht in diese Vase), Feld diagonal links oben + W[i,j] (Blume kommt in diese Vase))

Laufzeit/Lösungsgüte

Thomas Briner, brinerth@iiic.ethz.ch

Bohnen im Topf

Theorie

Idee/Vorgehen

Beliebige Anzahl weisse und schwarze Bohnen im Topf. Immer 2 ziehen: Falls 2 gleiche: beide raus, 1 schwarze rein, Falls 2 verschiedene: schwarze raus, weisse zurück.

terminiert? ja: nach jedem Ziehen hat sich die Anzahl Bohnen um 1 verringert

Schlusszustand:
je nach Anfangszahl weisse: Falls Gerade: Ende mit 1 Schwarzen, sonst mit 1 Weissen.
(Invariante: immer entweder -2 oder +-0 weisse --> gerade/ungerade bleibt immer erhalten)

Laufzeit/Lösungsgüte

Thomas Briner, brinerth@iiic.ethz.ch

Bubble-Sort

Algorithmus

Idee/Vorgehen

Immer zwei benachbarte tauschen, falls sie in der falschen Reihenfolge sind.

Im Worst Case muss das hinterste ganz nach vorne --> n mal die Vertauschungen durchlaufen

```
PROCEDURE Bubblesort (VAR A: ARRAY OF INT);
        VAR   changed:BOOLEAN;
              i, temp:INTEGER;
        BEGIN
              REPEAT
                    changed:=TRUE;
                    FOR i:= 1 TO (N-1) DO
                          IF A[i] > A[i+1] THEN
                             temp:=A[i]; A[i]:=A[i+1];
                             A[i+1]:=temp;
                             changed:=FALSE;
                           END;
                    END;
              UNTIL changed:
        END Bubblesort;
```

Vorsicht: Diese Zusammenfassung ist nicht mit Sicherheit richtig!

Laufzeit/Lösungsgüte O(n^2)

Thomas Briner, brinerth@iiic.ethz.ch

Computertomographie

Algorithmus

Idee/Vorgehen

quadratische Matrix, immer pro Reihe und pro Spalte ist die Anzahl schwarzer Felder vermerkt, Wie findet man eine korrekte Matrix?

Zeile und Spalte werden mit DIV bzw MOD berechnet, damit kontinuierlich vorwärts gezählt werden kann.

Dann wird das ganze rekursiv laufen, 1. Variante: leer lassen --> zum nächsten Feld
2. Variante: setzen, ist nur mögliche falls in dieser Zeile und Spalte noch # schwarze > 0, nach setzen in entsprechender Zeile, Spalte je 1 abzählen

Schlussbedingung: IF pos=n*m THEN

Laufzeit/Lösungsgüte

Thomas Briner, brinerth@iiic.ethz.ch

Das Spiel Go

Algorithmus

Idee/Vorgehen

Anzahl Blöcke einer Farbe finden -
-> für jedenStein der Farbe kontrollieren,
ob er an einen eigenen anstösst
(in 4 Richtungen ausschwärmen,
wenn ja, Stein löschen,
wenn nein: INC (# Blöcke)

Gibt es einen Block der Farbe, der mit einem gegnerischen Stein eingeschlossen werden könnte?

Für jeden Stein # Luftlöcher ermitteln. Falls Stein nur einen freien Platz hat, kennzeichnen und zurückmelden, dass nur 1 Luftloch

Laufzeit/Lösungsgüte

Thomas Briner, brinerth@iiic.ethz.ch

Dijkstra Programmiersprache

Dijkstra

Theorie

Idee/Vorgehen

Nur 4 Operationen:

- Nulloperation , skip

- Zuweisung (mehrfach möglich, simultan (x,y := 1, x+1)

- Bewachte Anweisung, offen falls Boolsche Wache =
  TRUE
  if a<b --> x:=b fi

- Alternative: Falls mindestens eine Boolsche Wache offen, dann wähle eine offene aus. Wenn keine mehr offen: Fehlerhalt
  if a<b --> x:=b | a>=b --> x:=a fi

Repetition: Solange mindestens eine Bool Wache offen, wähle eine offene aus und führe sie aus
  do a<b --> a,b := b,a
  | b<c --> b,c := c,b
  od {a <= b <= c } (Kommentar)

nicht deterministisch, mehrere Abläufe möglich, ist ideal um einen Ort zu finden, an dem mehrere Variablen gleich sind (a nicht kleiner als b, b nicht kleiner als c, c nicht kleiner als a --> a=b=c)

Vorsicht: Diese Zusammenfassung ist nicht mit Sicherheit richtig!

Laufzeit/Lösungsgüte

Thomas Briner, brinerth@iiic.ethz.ch

Durchschnittsberechnung (Zero-Knowledge)

Algorithmus

Idee/Vorgehen

3 Einkommen, A, B und C, es soll Durchschnitt berechnet werden, ohne dass man ein Einkommen preisgeben muss:

Jeder teilt sein Einkommen in 3 beliebige Teile, zB A1, A2 und A3.

So erhält jeder 3 Teile zB A2, B2 und C2. Aus diesen 3 Teilen bildet er nun die Summe und meldet die zurück.

Aus diesen 3 Summanden kann nun der Durchschnitt gebildet werden.

Laufzeit/Lösungsgüte

Thomas Briner, brinerth@iiic.ethz.ch

EBNF Notation

Theorie

Idee/Vorgehen

Mögliche Verbindungen:
- Konkatenation: A B (aneinanderhängen)
- Alternative: A | B
- Option: [A] (entweder 1 x oder 0 x das „A“ = entweder A oder nichts)
- Repetition: { A } (entweder 0x oder 1x oder 2x oder.... das A = „ „ oder „ A „ oder „AA“ oder „AAA“ oder...)

Extended Backus Naur Form

EBNF beschreibt sich selber:

Syntax = { Production }.
Production = Name „ = „ Expression „ . „.
Expression = Term { „ | „ Term }.
Term = Factor { Factor }
Factor = Name | String | „ { „ Expression „ } „ | „ [„ Expression „] „ | „ („ Expression „) „.

Laufzeit/Lösungsgüte

Thomas Briner, brinerth@iiic.ethz.ch

Editierdistanz

Algorithmus

Idee/Vorgehen

Wie nahe sind 2 Worte beeinander?

Es wird immer das ein Zeichen vom oberen und 1 vom unteren Wort betrachtet.
Fall 1: oben und unten gleich --> keine Abweichung
Fall 2: oben und unten verschieden --> Abweichung +1
Fall 3: oben leer --> Abweichung +1
Fall 4: unten leer --> Abweichung +1

Dynamisch lösen:

1 Wort senkrecht, anderes waagrecht
Diagonal: +1 falls verschieden, + 0 falls gleich
nach rechts: + 1
nach unten: + 1

immer Minimum wird ausgewählt

rechts unten bestes Resultat

Laufzeit/Lösungsgüte

Thomas Briner, brinerth@iiic.ethz.ch

Effizientes Mergesort mit versch. Bändern

Theorie

Idee/Vorgehen

Bänder verschiedener Länge, L4, L15, L5, L2

Wie ist der beste Ablauf, um möglichst wenig Operationen zu haben?

zB. L4 + L15 --> 19 Operationen
L2 + L5 --> 7 Operationen
Resultate verschmelzen: 26 Operationen
TOTAL: 52 Operationen

Ideal: L2 + L4, Ergebnis mit L5, Ergebnis mit L15
= 6+11+26 Operationen = 43 Operationen

Ideal, wenn die Läufe möglichst gleich lang sind
(bei Merge L2 und L15 viele vergebliche Op.)

Laufzeit/Lösungsgüte

Thomas Briner, brinerth@iiic.ethz.ch

Exponentielle Suche

Algorithmus

Idee/Vorgehen

Binäre Suche nur bei endlicher Datenmenge möglich

bei unendlicher Menge, sortiert:

> Beginn bei A[1], vergleichen, falls key > A[1]:
> um 2, 4, 8, 16, ... Schritte nach rechts gehen

O(log k); k: effektive Position des Schlüssels

Laufzeit/Lösungsgüte O (log k)

Thomas Briner, brinerth@iiic.ethz.ch

Färben von Graphkanten online

Algorithmus

Idee/Vorgehen

Bei jeder Kante, die neu dazukommt, wird bei Farbe 1 begonnen und solange hochgezählt, wie die Farbe schon bei einem der beiden Endknoten benutzt wird.

--> Lösungsgüte <=2

Grund: x:= höchste im Graph vorkommende Knotengradzahl

vor Einfügen der Kante haben beide Knoten höchstens x-1 Kanten, die höchste Farbe, die ich für die neue Kante also wähle ist (2*x)-1, da aber der Graph einen Knoten mit Grad x hat, brauche ich auf jeden Fall mindestens x Farben

Laufzeit/Lösungsgüte

Thomas Briner, brinerth@iiic.ethz.ch

fehlerresistenter Zahleneinleser

Algorithmus

Idee/Vorgehen

Es sollen Ganzzahlen eingelesen werden,
bei jeder Ziffer wird der aktuelle Wert berechnet,
bei einer falschen Eingabe soll kein Trap entstehen

--> Eingabe als CHAR
ORD(ch) - ORD(„0“) berechnen
Falls das zwischen 0 und 9 ist:
alte Zahl *10 + neue Ziffer

Für komplexere Eingabemöglichkeiten:

gerichteten Graphen aufzeichnen

Laufzeit/Lösungsgüte O (n)

Thomas Briner, brinerth@iiic.ethz.ch

Ferien von 5 Freunden planen

Algorithmus

Idee/Vorgehen

Wollen gemeinsam Ferien machen, gibt es einen Tag, an dem es allen geht?

Datenstruktur?

- ARRAY N+1,356 OF INTEGER,
  pro Freund eine Zeile + 1 Auswertung

1 Tag: Auswertung berechnen,
Finde, ob A[0,i]=5

14 Tage mit möglichst vielen:
ARRAY 5 OF INTEGER, Init mit 356
darin immer wenn in Auswertung ansteigt, eintragen.
wenn fällt: Für alle Werte grösser als Ausertung neu schauen, ob schon 14 Tage seit Anstieg, wenn ja vgl mit Maxbisher, ev. ändern

Laufzeit/Lösungsgüte

Thomas Briner, brinerth@iiic.ethz.ch

Finde 2 Zahlen mit Summe = 100

Algorithmus

Idee/Vorgehen

ARRAY mit Ganzzahlen, pos und neg, aufsteigend geordnet.

1.Var: von links nach rechts:
nimm Zahl, such im Rest vom ARRAY mit
binärer Suche ob Gegenstück vorhanden
O(n log n)

2. Var: 2 Indexvar, eine am Anfang, eine am Schluss,
betrachte Summe
IF sum < 100 THEN INC (links);
ELSIF sum> 100 THEN DEC (rechts);
ELSE RETURN found;
END;

Laufzeit/Lösungsgüte

Thomas Briner, brinerth@iiic.ethz.ch

Finde Nullstelle, Dijkstra

Algorithmus

Idee/Vorgehen

Einheitsintervall mit garantierter Nullstelle,
Funktion streng monoton steigend,
effizienter Algorithmus gesucht,
Treffergenauigkeit: Abweichung <0.001

```
x:=0; step:=0.5;
do  (step>0.001) & (f(x)>0) --> x:=x-step; step:=step/2
      (step>0.001) & (f(x)<0) --> x:=x+step; step:=step/2
od;
```


Laufzeit/Lösungsgüte

Thomas Briner, brinerth@iiic.ethz.ch

Geschichtete Bäume

Datenstruktur

Idee/Vorgehen

auch: stratified tree

4 Grundtypen sind zugelassen (statt nur 1 Knoten), ev. für Wurzel noch spezielle Regeln

Einfügen: aus Blatt inneren Knoten machen,
Kontrolle ob immer noch einer der Grundtypen,
sonst: Push up, die Schichten werden nach oben verlagert, falls bis Wurzel weitergeht:
Baum wächst in der Höhe um 1

Suche: O(log n)

Einfügen: Suchen: O(log n)
Strukturänderung O(1)
Schichänderung O(log n)

Entfernen dito

Laufzeit/Lösungsgüte

Thomas Briner, brinerth@iiic.ethz.ch

Graphenrepräsentation

Datenstruktur

Idee/Vorgehen

Adjazenzmatrix:

gut geeignet für viele Kanten, braucht n^2 Speicherplatz, Eintrag ist Verbindungslänge bzw. TRUE falls Verbindung besteht.

Adjazenzliste:

Liste mit allen Knoten drin, angehängt an jedem die Liste mit den Knoten, mit denen er verbunden ist.

Laufzeit/Lösungsgüte

Thomas Briner, brinerth@iiic.ethz.ch

Graphentraversierung (breadth, depth)

Algorithmus

Idee/Vorgehen

Breitensuche:

Beginn bei Knoten v, alle Knoten, die von v aus direkt erreicht werden können, werden in eine Queue eingetragen. Dann wird der vorderste Knoten der Queue abgearbeitet und wieder alle Knoten, die von dort erreichbar sind, angehängt.

Vorteil: Kreisförmiges Suchen, es wird mit Sicherheit die kürzeste Verbindung zu einem anderen Knoten zuerst gefunden

Tiefensuche:

Vom Startknoten aus werden die direkt erreichbaren Knoten auf einen Stack geschrieben. Nun wird das gleiche immer mit dem obersten Knoten auf dem Stack wiederholt, bis gefunden oder bis Sackgasse, dann nächster Knoten --> Backtracking

Laufzeit/Lösungsgüte

Thomas Briner, brinerth@iiic.ethz.ch

Handelsreisenden Stationen, Dijkstra

Idee/Vorgehen

4 Handelsreisende, gleiche Routen,
machen an verschiedenen Orten Station,
schreiben sich immer Tageskilometer auf.
Haben sie einmal an gleichem Ort übernachtet?

Bedingung: A[i]=B[j]=C[k]=D[l|

```
i,j,k,l:=0,0,0,0
do      A[i]<B[j] --> i:=i +1
        B[j] < C[k] --> j:=j + 1
        C[k] < D[l] --> k:=k+1
        D[k] < A[i] --> l:=l + 1
od
```

Vorsicht: Diese Zusammenfassung ist nicht mit Sicherheit richtig!

Laufzeit/Lösungsgüte

Thomas Briner, brinerth@iiic.ethz.ch

Hashing mit Ueberlauf

Datenstruktur

Idee/Vorgehen

Die Kollisionen werden als lineare Liste an das ARRAY angehängt. Falls der key nicht am richtigen Ort im ARRAY steht, wird die Liste traversiert

Nachteil:
- 2 verschiedene Datenstrukturen
- Zeitverlust beim Traversieren, falls viele Kollisionen auftreten

Laufzeit/Lösungsgüte

Thomas Briner, brinerth@iiic.ethz.ch

Hashing, offen; double hashing

Datenstruktur

Idee/Vorgehen

Bei Kollisionen werden die Daten in noch freie Plätze eingewiesen.
Konstante Sprungweite ungeschickt.
--> 2 Hashfkt.
h(x): 1. Adresse wird erzeugt
g(x): Sprungweite wird festgelegt

Wenn Key nicht an 1. Adresse gefunden wird, springe nach unten bis du hast. Falls du auf leeres Feld triffst --> nicht in Liste enthalten

Vorteil: - nützt den Speicherplatz gut aus
Nachteil: - kompliziert

Problem delete: Bei Löschen muss ein Dummy „deleted“ eingefügt, damit nicht Suche fälschlicherweise abgebrochen wird.

Aber: irgendwann meiste Zellen voll oder deleted
--> lange Suche

delete nicht gut gelöst

Loadfactor = # Einträge / # Zellen

sinnvoll zwischen 0.4 und 0.8

Laufzeit/Lösungsgüte

Thomas Briner, brinerth@iiic.ethz.ch

Hashing, perfekt

Datenstruktur

Idee/Vorgehen

Alle Einträge müssen schon bekannt sein,
die Hashfunktion wird so gewählt, dass keine Kollisionen auftreten

zB: Compiler, reservierte Wörter

ev. mit Fallunterscheidung

```
IF x=0 THEN h(x):=8;
ELSE h(x):=x DIV 3;
END;
```


Laufzeit/Lösungsgüte

Thomas Briner, brinerth@iiic.ethz.ch

Hashing, uniform

Datenstruktur

Idee/Vorgehen

Alle Ausgangswerte im ARRAY S werden gleichoft gesucht, S sehr gross

S wird einfach komprimiert, d.h.es werden jeweils mehrere Daten in gleiche Adresse in A geschrieben

Die Ordnung bleibt erhalten

ist aber gar nicht der Normalfall

Laufzeit/Lösungsgüte

Thomas Briner, brinerth@iiic.ethz.ch

Heap

Datenstruktur

Idee/Vorgehen

Elemente eigentlich in einem ARRAY gespeichert.
In einem max-Heap ist jedes Element i grösser als die Elemente 2*i bis 2*(i+1)-1; in Baumdarstellung grösser als seine 2 Kinder --> Max ist immer an Wurzel.

Heap erstellen:
ab Mitte (len DIV 2) bis nach vorne immer Elem. versickern lassen

Siftdown:
Das Element tauscht mit dem grösseren seiner Kinder die Position. Das wird sooft wiederholt, bis das Element keine Kinder mehr hat.

Neue El in Heap einfügen:
an hinterste neue Position stellen, falls < als Vorgänger nach oben wandern lassen.

Laufzeit/Lösungsgüte Erstellen: O(n)

Thomas Briner, brinerth@iiic.ethz.ch

Heapsort

Algorithmus

Idee/Vorgehen

Ungeordnetes ARRAY

zuerst Heap herstellen (Vordere Hälfte Siftdown)

dann:

- Maximum aus Wurzel mit höchstem Index (n) tauschen
- Wurzel versickern lassen
- wiederholen für ARRAY 1..n-1

Laufzeit/Lösungsgüte O(n log n)

Thomas Briner, brinerth@iiic.ethz.ch

Horner-Schema

Algorithmus

Idee/Vorgehen

Möglichkeit für Polynom-Herstellung

$(((((an)*x+a_{n-1})*x + a_{n-2})*x + a_{n-3})*x + ...)*x+ a_0$

Laufzeit/Lösungsgüte

Thomas Briner, brinerth@iiic.ethz.ch

Huffman-Bäume

Huffman

Datenstruktur

Idee/Vorgehen

optimaler (kürzester) Präfixfreier Code für Schlüssel mit gegebenen Häufigkeiten:

--> immer die 2 mit den kleinsten Häufigkeiten zusammenfassen, ein Ast 0, ein Ast 1

A	B	C	D
0.5	0.1	0.1	0.3

0.2

0.5

1

A=0, B=100, C=101, D=11

Wird in Heap verwaltet:
Min1 holen, Min 2 holen, (Min1 + Min2) einfügen

Laufzeit/Lösungsgüte

Thomas Briner, brinerth@iiic.ethz.ch

Induktives Beweisen

Theorie

Idee/Vorgehen

Eine zB. durch Teleskopieren gefundene Formel wird mit Induktion bewiesen.

Aus Teleskopieren:

T(2)=1,
T(n)=2*T(n/2)+2 ist von der Grössenordnung 3/2*n -2

Ind. Verankerung: n=2:

3/2*2-2 = 1, ok.

Ind. Schritt: n-->2*n

T(2n) =2*T(n)+2= 2*(3/2*n-2)+2
=3n+2=3/2*2n+2, ok.

Vorsicht: Diese Zusammenfassung ist nicht mit Sicherheit richtig!

Laufzeit/Lösungsgüte

Thomas Briner, brinerth@iiic.ethz.ch

Interpolationssuche

Algorithmus

Idee/Vorgehen

In geordnetem ARRAY:
Aufgrund Annahme der ungefähren Gleichverteilung kann mit Hilfe der # Elemente und des Wertebereichs die ungefähre wahrscheinlichste Position ermittelt werden.

$$(A[n] - A[0]) / (n-0) = (key - A[0]) / (pos - 0)$$

Worst Case: O(n)

im Mittel (bei Gleichverteilung): O(log(log(n))

Laufzeit/Lösungsgüte zwischen O(log(log n)) und O(n)

Thomas Briner, brinerth@iiic.ethz.ch

Intervalleinschluss

Algorithmus

Idee/Vorgehen

Problem: Finde Intervalle, die ganz in einem anderen Intervall enthalten sind.

naiv: Jedes mit jedem Vergleichen O(n^2)

Induktion:

Bedingung: Intervall n ist immer das Int. mit rechtestem linkem Endpunkt.

bisher weitester rechter Endpunkt xrechts muss gespeichert werden.

Falls xrechts > rechter Rand von Int n
--> eingeschlossen
Gibt es Int., die Int. n einschliesst?
--> Müssten gleichen linken Rand haben
--> Zusatzbedingung: nimm bei mehreren mit gleichem linken Ende immer zuerst das Längste

Vergleiche: O(n)
Intervalle nach linkem Rand ordnen: O(n log n)

Vorsicht: Diese Zusammenfassung ist nicht mit Sicherheit richtig!

Laufzeit/Lösungsgüte

Thomas Briner, brinerth@iiic.ethz.ch

K - Zentrum

Algorithmus

Idee/Vorgehen

gegeben eine Menge von Punkten in 2 Dim., gegeben eine Anzahl k Clusters/Gruppen

Wie kann ich die Clusters so wählen, dass die Radien der Cluster möglichst klein sind?

- wähle einen beliebigen Punkt als Zentrum 1.
- nimm als nächstes den Punkt, der am weitesten vom Zentrum 1 entfernt ist als neues Zentrum.
- suche nun den Punkt, dessen Distanz zum nächsten Zentrum maximal ist (alle anderen Punkte haben innerhalb einer kleineren Distanz irgend ein Zentrum) und wähle den als neues Zentrum

wiederhole sooft, bis k Zentren gefunden, nun müssen die Punkte noch den k Clustern zugeordnet werden

Lösungsgüte = 2

Laufzeit/Lösungsgüte

Thomas Briner, brinerth@iiic.ethz.ch

Kürzeste Wege im Graphen

Ford, Bellman

Algorithmus

Idee/Vorgehen

Breitensuche in Graph
Anfangsbedingungen: Startknoten hat 0 als Distanz, alle anderen haben MAX(INT).

Nun werden alle Knoten angeschaut, die vom Startknoten aus zu erreichen sind. (=Rand)
Falls der Wert bei diesem Knoten grösser ist, als bei einem anderen Nachbarknoten + Weglänge zwischen diesen beiden, so wird der Wert geändert (relabel)

Laufzeit/Lösungsgüte

Thomas Briner, brinerth@iiic.ethz.ch

Kürzeste Wege im Graphen

Dijkstra

Algorithmus

Idee/Vorgehen

Aufteilung in Kern, Rand und unbekannt.
Anfang: Kern: Startknoten
Rand: alle Knoten, die vom Startknoten direkt erreicht werden können.

Nun wird der Randknoten(=r) mit der kleinsten Distanz zum Start dazugenommen. Jede Linie die von r weggeht wird bearbeitet.

Fall 1: zu Knoten im Kern: nichts tun.
Fall 2: zu Randknoten: Weg via r mit bisherigem Eintrag vergleichen, ev. verbessern
Fall 3: zu unbekanntem Knoten: wird neu in Rand aufgenommen mit der Distanz via r

Aufwand: jedesmal Minimum aus Rand suchen O (n)
jedesmal Pfeilen folgen c*n

O(n^2)

Idee: Rand als Heap verwalten, falls nicht zuviele Kanten ist das besser (O (m* log(n)))

Laufzeit/Lösungsgüte

Thomas Briner, brinerth@iiic.ethz.ch

Kunstwerk im Kunstwerk

Algorithmus

Idee/Vorgehen

Implementation für :
Kunstwerk ist Quadrat mit Seitenlänge S,
kann entweder mit einer von 4 Farben ausgemalt
werden oder mit 4 x 4 neuen Kunstwerken rekursiv
gefüllt werden.

EBNF:
Kunstwerk = rot | blau | gelb | grün | 4x4 Kunstwerk.

```
PROCEDURE Kunstwerk*(x,y,Seitenlänge:INT);
        BEGIN
        IF Read="r" THEN....
        ELSIF Read="Kunst" THEN
              FOR i:=0 TO 3 DO
                FOR j:=0 TO 3 DO
                      Kunstwerk (x+i*Seit, y+j*Seit, Seit/4);
                END;
              END;
        END;
```

Vorsicht: Diese Zusammenfassung ist nicht mit Sicherheit richtig!

Laufzeit/Lösungsgüte

Thomas Briner, brinerth@iiic.ethz.ch

Lineare Liste für Wörterbuch

Datenstruktur

Idee/Vorgehen

Erfolglose Suche:
unsortiert:
Search : n Operationen
sortiert:
Search : n/2 Operationen

Worst Case:
unsortiert:
Search O(n)
Insert: O(1)
Delete: O(n)

sortiert:
Search O(n)
Insert O(n)
Delete O(n)

Laufzeit/Lösungsgüte

Thomas Briner, brinerth@iiic.ethz.ch

Lineare Suche

Algorithmus

Idee/Vorgehen

In ARRAY oder linearer Liste (geordnet oder ungeordnet)

Die Elemente werden von links nach rechts kontrolliert.

Im Mittel: n/2 Kontrollen

Worst Case: n Kontrollen

Laufzeit/Lösungsgüte O(n)

Thomas Briner, brinerth@iiic.ethz.ch

Liniensegment - Schnittproblem

Algorithmus

Idee/Vorgehen

horizontale und vertikale Liniensegmente im R^2 verteilt.

von jeder Linie Anfang(x,y) und Ende(x,y) bekannt.

naiv: jede Linie mit jeder vergleichen --> Laufzeit O n^2

Variante:

Sweep- Line, Scanner von links her, Menge der y-Werte der aktuellen horizontalen Linien mitführen, bei Beginn bzw Ende einer horizontalen Linie Menge wieder aktualisieren.

Immer von einer vertikalen Linie zur nächsten springen, kontrollieren, hat es in Menge einen y Wert, der zwischen Anfang und Ende der vertikalen Linie liegt? Wenn ja: Schnittpunkt noch registrieren

statt Menge mitführen auch möglich: y Werte der horizontalen Linien in einem Baum ablegen, eventuell Blätter untereinandern noch verknüpfen,
Dann einfach nach yEndwerten der senkrechten Linie suchen.

Laufzeit/Lösungsgüte O (n log (n))

Thomas Briner, brinerth@iiic.ethz.ch

Lokale Suche

Algorithmus

Idee/Vorgehen

Beginn mit einer willkürlichen Lösung

Vergleiche mit unmittelbaren Nachbarn,
falls einer der Nachbarn besser ist, wechsle¨

falls innerhalb einer bestimmten Anzahl Schritte kein Wechsel passiert ist, brich ab.

Je nach Lösungslandschaft kann es sein, dass man nur ein lokales, nicht sehr gutes Maximum findet.

Vorsicht: Diese Zusammenfassung ist nicht mit Sicherheit richtig!

Laufzeit/Lösungsgüte

Thomas Briner, brinerth@iiic.ethz.ch

Longest common subsequence

Algorithmus

Idee/Vorgehen

2 Zeichenfolgen, welches ist längstmögliche Zeichenfolge, die in beiden vorkommt (Buchstaben dürfen ausgelassen werden)

X= A B C D E D A G N D G E E
Y= T A B W D W G N A A E G E

Lcs= A B D G N G E

Dynamisch:
Gitter, 1 Wort senkrecht, das andere Waagrecht
0 initialisieren

3 Möglichkeiten:

1. Diagonal nach rechts unten: beide Buchstaben genommen, Wert von links oben +1 eintragen
2. Von oben: im senkrechten Wort 1 Buchst ausgelassen
3. Von links: im waagrechten Wort 1 Buchst. ausgelassen,

Immer Maximum wird eingetragen, im Feld rechts unten steht die maximale Länge der LCS

Vorsicht: Diese Zusammenfassung ist nicht mit Sicherheit richtig!

Laufzeit/Lösungsgüte

Thomas Briner, brinerth@iiic.ethz.ch

Masse für Vorsortiertheit

Theorie

Idee/Vorgehen

zB # Runs

Ein Run innerhalb einer Folge ist eine Teilfolge, in der die Werte nur ansteigen

zB # Inversionen

Wieviele Elemente in Folge hinter Element x sind kleiner als Element x

~Anzahl Vertauschungen, um zur richtigen Reihenfolge zu kommen

Laufzeit/Lösungsgüte

Thomas Briner, brinerth@iiic.ethz.ch

Matrixmultiplikation

Strassen

Algorithmus

Idee/Vorgehen

Ueblicherweise Matrixmulitplikation
m x n * n x p - Matrizen braucht m*n*p Multipl/Add.

---> O (n^3)

Dank Zerlegung in 7 Blockmatrizen nun nur noch O (n^2.81....)

Laufzeit/Lösungsgüte O (n^3) bzw. O (n^2.81.....)

Thomas Briner, brinerth@iiic.ethz.ch

Max Punktzahl in Prüfung

Algorithmus

Idee/Vorgehen

N Aufgaben, jede Aufgabe gibt Punkte W[i], braucht Zeit Z[i], wie findet man optimale Lösung (angefangene Aufg geben keine Pkt)

immer bester Quotient W[i]/Z[i] nehmen ergibt nicht unbedingt beste Lösung

Dynamisch lösen

ARRAY, senkrecht Aufgaben, wagrecht Zeit,
Aufg nicht lösen: Zahl von Reihe oben übertragen
Aufg lösen: Zahl in oberer Reihe, Z[i] Positionen weiter links + W[i]
nimm immer Max

beste Lösung: Zahl am weitesten unten links

Lösungsmenge finden: von unten aufrollen, falls gleiche Zahl obendran: --> nicht genommen

Laufzeit/Lösungsgüte

Thomas Briner, brinerth@iiic.ethz.ch

Max und Min gleichzeitig finden

Algorithmus

Idee/Vorgehen

naiv: zuerst Max suchen --> n-1 Vgl
dann Min suchen --> n-2 Vgl

Divide and conquer: Halbieren,
Annahme: in jeder Hälfte Max, Min bekannt,
dann nur noch max1 und max2 vgl etc.
bei nur 2 Elementen: 1 Vergleich

--> 3/2n + 2

Laufzeit/Lösungsgüte O(n)

Thomas Briner, brinerth@iiic.ethz.ch

Maximum finden

Algorithmus

Idee/Vorgehen

ARRAY oder Liste, ungeordnet

Von links her scannen,
immer bisheriges Maximum mit aktuellem Wert vergleichen,
ev. übernehmen

Alternative: Min-Heap bilden (O(n))

für i-kleinstes: i * Min rausnehmen,
versickern lassen, i*log n

oder mit Median-of-median in O(n)

Laufzeit/Lösungsgüte O(n)

Thomas Briner, brinerth@iiic.ethz.ch

Maximum Subarray

Algorithmus

Idee/Vorgehen

naiv: von jedem aus bis zu jedem alles aufsummieren
--> O (n^3)

Altern: Summe immer übernehmen und nur noch letztes Element dazuzählen
O (n^2)

Altern: Preprocessing
Vorher Array berechnen, wieviel Differenz von jedem zum ersten Element (als absolute Zahlen und nicht mehr als Schritt im Vergleich zum Vorgänger)
Dann immer Differenz Einstiegstag - Ausstiegstag berechnen
O(n^2)

Ultimo: Scanlinie von links immer bisheriges max und randmax bis zur Scanlinie mitführen.
IF randmax + A[aktuell]< 0 THEN randmax:=0; (gar nicht einsteigen)
ELSE randmax:=randmax+A[aktuell].
Falls randmax > max, als neues max übernehmen.
O(n)

Laufzeit/Lösungsgüte O (n)

Thomas Briner, brinerth@iiic.ethz.ch

Median suchen

Blum

Algorithmus

Idee/Vorgehen

naiv: grösstes raus, 2.grösstes raus, ...
bis zum Mittleren, O(n^2)

Wie Median finden, zB als gutes Pivot für BinSearch?

- ARRAY in 5er Blöcke aufteilen,
- jeden Block sortieren
- mittlere Elemente wieder in ARRAY
- Verfahren rekursiv anwenden

--> Median der Mediane ist gutes Pivot

Analyse: T(n)<= f*n + e*n + T(n/5) + T(7/10n + 6)

n0=91, O(n)

Laufzeit/Lösungsgüte O(n)

Thomas Briner, brinerth@iiic.ethz.ch

Mergesort

Algorithmus

Idee/Vorgehen

- ARRAY wird in 2 Hälften geteilt,
- jede Hälfte wird in sich geordnet
- die beiden Teile werden verschmolzen
- rekursiv aufgerufen

```
PROCEDURE Mergesort....
        ...     m:=(l+r) DIV 2; Mergesort(A, l,m);
          Mergesort(A, m+1,r);
          Merge(A, l, m, r);

PROCEDURE Merge
          VAR B: ARRAY OF INTEGER
       ....       WHILE (i<=m) & (j<=r) DO
                  IF A[i]<A[j] THEN B[k]:=A[i]; INC i;
                  ELSE.....
          IF i>m THEN          (*1. Teil schon fertig*)
                  FOR s:=1 TO r-j DO
                          B[k+s]:=A[s+j]; END;....
```

Es können auch mehr Stränge miteinander gemerget werden (zB 5 mit je 60 Läufen von je 5 Zeichen), immer 1 Lauf von allen 5 Bändern auf ein Zielband --> auf Zielband 60 Läufe mit je 25 Zeichen, dann wieder auf 5 Bänder verteilen, je 12 Läufe à 25 Zeichen etc.

Laufzeit/Lösungsgüte

Thomas Briner, brinerth@iiic.ethz.ch

Minimum Spanning Tree

Kruskal

Algorithmus

Idee/Vorgehen

Sortiere alle Kanten nach der Länge,
nimm immer die kürzeste,
wähl sie aus, sofern sich kein Zyklus ergibt

Mit Union- Find Struktur verwalten, ob es ein Zyklus ist.

Laufzeit/Lösungsgüte O (m * log(n))

Thomas Briner, brinerth@iiic.ethz.ch

Minimum Spanning Tree

Dijkstra

Algorithmus

Idee/Vorgehen

gleicher Ansatz wie kürzester Weg mit Kern, Rand, unbekannt.
Start: Nimm beliebigen Knoten als Kern

Wähle kürzeste Kante, die über den Schnitt Kern - Rand geht.
Nimm den Endknoten im Rand zum Kern dazu, definiere den neuen Schnitt, nimm Nachbarknoten zum Rand dazu

Laufzeit/Lösungsgüte O (n^2)

Thomas Briner, brinerth@iiic.ethz.ch

Missionare und Kanibalen

Algorithmus

Idee/Vorgehen

Zustandsraum zusammenstellen

ungültige Positionen rausstreichen

mögliche Züge aufzeichnen

im Zustandsraum Weg von A nach B suchen

Laufzeit/Lösungsgüte

Thomas Briner, brinerth@iiic.ethz.ch

Natürliche Suchbäume

Datenstruktur

Idee/Vorgehen

Jeder Baum 2 Zeiger, geordnet nach key,

Einfügen: entlang Baum nach unten, dann in Blatt key einfügen

Abbilden auf Strahl: Immer abwechselnd Blatt, innerer Knoten

Problem: Degeneration zu Liste i m Worst Case
Suche O(n) statt O(log n)

Mittel über alle Permutationen: O(log n)
Mittel über alle möglichen Baumstrukturen: O(sqrt(n))

Variante: Blattsuchbaum,Schlüssel nur in Blättern, braucht etwas mehr Platz

Variante: Sentinel unten einfügen, Key für Suche in Sentinel schreiben

Laufzeit/Lösungsgüte

Thomas Briner, brinerth@iiic.ethz.ch

Nearest Neigbour in 2 Dim

Algorithmus

Idee/Vorgehen

nächster Nachbar in 1 Dim:
Sortieren, berechne immer Differenz zw. 2 Elementen, Find Min

in 2 Dim: 1.Var:
Induktion, immer 1 Punkt dazu-nehmen, mit allen n-1 bisherigen Pkt vergleichen , O (n^2)

2. Var:
Divide and conquer, zuerst nach y-Koordinate sortieren (für nachher bei Mittelstreifenkontrolle
mittlere x - Koordinate finden, senkrechte Linie durch diesen Median legen, Annahme: in jeder Hälfte nächste Nachbarn gefunden
min 1 und min2 vergleichen. Entlang Mittelstreifen noch mit kleinerem Minimum kontrollieren, ob noch Punkte in diesem Band näher sind , neues Min.

y Koord sortieren: O (n log n), Dann log n Schritte jedes Mal: Median x finden: O (n log n), Punkte entlang steifen suchen: O (n log n),

Laufzeit/Lösungsgüte

Thomas Briner, brinerth@iiic.ethz.ch

Objektorientierung

Theorie

Idee/Vorgehen

Zusätzliche Variablenart: Prozedurvariablen

damit lassen sich im ADT vorsehen, was mit einem bestimmten Objekt zu tun ist, ohne es schon im voraus zu wissen.

Jedes Objekt hat die Prozeduren, mit denen es bearbeitet wird, gerade schon in seinen eigenen Variablen drin fixiert.

Laufzeit/Lösungsgüte

Thomas Briner, brinerth@iiic.ethz.ch

Optimale Suchbäume

Datenstruktur

Idee/Vorgehen

Suchbäume, bei denen die Zugriffhäufigkeiten auf die Knoten bekannt sind.

optimal: Ž(#Zugriffe * Weglänge) = minimal

Weglänge zu Wurzel = 1

Der optimale Suchbaum ergibt sich nicht immer, wenn die Knoten in Reihenfolge der Häufigkeit eingefügt werden.

	key 1	key 2		key 3	
Zw.raum	Zw.raum	Zw.raum		Zw.r.	
4	3	5	3	2	1	6

wij: wie oft wird auf den Bereich von 0 bis 2 zugegriffen?
--> 0+4+3+5+3+2 Mal = 17, Gitter ausfüllen

Pij: Intervall aufteilen, Minimum suchen für P 0,2 :
(P 0,1 + P 1,2) oder (P 0,0 + P0,2) + w 0,2

Gitter Root, worin festgehalten wird, wo aufgeteilt wurde --> Wurzel des Teilbaums

Vorsicht: Diese Zusammenfassung ist nicht mit Sicherheit richtig!

Laufzeit/Lösungsgüte

Thomas Briner, brinerth@iiic.ethz.ch

Optimales Produkt mehrerer Matrizen

Algorithmus

Idee/Vorgehen

n Matrizen, wie soll am besten geklammert werden?

Lösungsgitter, Diagonalen immer 0, nur oberer Teil wird benötigt

Wenn nur 2 Matrizen: # Multiplikationen = n*q * r, eintragen in Matrix

Weitere Felder: immer diagonaleweise füllen, bei jeder die neu dazukommt entscheiden, wo klammern, Minimum eintragen

--> rechts oben wird das bestmögliche Resultat erscheinen

Laufzeit/Lösungsgüte

Thomas Briner, brinerth@iiic.ethz.ch

Palindromtest

Algorithmus

Idee/Vorgehen

a man a plan a canal: panama

WHILE i<=n DIV 2 & A[i]=A[n-i] DO INC(i); END

Laufzeit/Lösungsgüte

Thomas Briner, brinerth@iiic.ethz.ch

Polynomauswertung

Algorithmus

Idee/Vorgehen

Polynom n-ten Grades berechnen:
$Pn(x) = a_n*x^n+a_{n-1}*x^(n-1)+ .... + a_0$

mit Induktion 1:

Annahme Pn-1(x) schon berechnet
Pn(x) = Pn-1(x) + an*x^n; (= an*x*x*x*.....*x)
$P0(x) = a_0$
jeder Schritt n Multiplikationen --> O(n^2)

mit Induktion 2:

Annahme Pn-1 (x) und x^(n-1) schon berechn.
Pn(x) = Pn-1(x) + an * x * x^(n-1)
jeder Schritt 2 Multiplikationen --> O(n)

Anderer Induktionsansatz von hinten:

Qn-1(x) = an*x^n + an-1 * x^n-1 + ... + a1*x
= x*(an*x^n-1+an-1*x^n-2+....+a1)
hinterer Teil Rn-1(x) hat wieder gleiche Form wie Pn(x)

Pn(x) = x* Rn-1(x) + a0

jeder Schritt 1 Add, 1 Mult --> O(n)

Vorsicht: Diese Zusammenfassung ist nicht mit Sicherheit richtig!

Laufzeit/Lösungsgüte

Thomas Briner, brinerth@iiic.ethz.ch

Priority-Queue

Datenstruktur

Idee/Vorgehen

Die Elemente enthalten einen Key, der die Priorität angibt. Sie werden nicht hinten in die Q eingehängt, sondern an dem Platz, wo sie nach key hingehören.

Für die Priority-Q als ADT wird beim init noch eine Vergleichsprozedur vom Client benötigt

SERVER: siehe Q in Liste, ausser

```
TYPE  comp: PROCEDURE (a,b: Elem): BOOLEAN;
      Qdesc=RECORD
            compare:comp;
            END;
PROCEDURE init*(vgl:comp):Q;
      BEGIN
            NEW(Q);
            Q^.compare:=vgl; .....

PROCEDURE push* (VAR q: Q; e:Elem);
      VAR before, cur: Elem;
      BEGIN
            cur:= Q^.first; before:=Q^.first;
            WHILE Q^.compare(cur, e) DO
                  cur:=cur^.next;
            END;
            e^.next:=cur;
            before^.next:=e; END push;
```

Vorsicht: Diese Zusammenfassung ist nicht mit Sicherheit richtig!

Laufzeit/Lösungsgüte enq: O(n), deq: O(1); init: O(1)

Thomas Briner, brinerth@iiic.ethz.ch

pseudopolynomielle Laufzeit

Theorie

Idee/Vorgehen

Ein Algorithmus, der eine Laufzeit hat, die nicht nur von der Anzahl der Elemente abhängt, sondern auch noch von einem Wert, der von der Eingabe abhängt

z.B: Rucksackproblem: O(n * G)
G ist die Gewichtslimite in diskrete Teile unterteilt

Laufzeit/Lösungsgüte

Thomas Briner, brinerth@iiic.ethz.ch

Queue in ARRAY

Datenstruktur

Idee/Vorgehen

FiFo

2 Zeiger, vorne, hinten
enqueue: hängt bei hinten an, INC (hinten)
dequeue: nimmt bei vorne weg, INC (vorne)
ismt?: vorne>hinten?

ohne Rückkoppelung: bei Erreichen bestimmter Länge wieder alles nach vorne schieben
mit Rückkoppelung: beim Füllen hinten angelangt einfach wieder vorne weitermachen, bis hinten+1=vorne --> full
Zusatzfkt: multidq(key): Alles weg bis key kommt oder Q leer ist

Vorsicht: Diese Zusammenfassung ist nicht mit Sicherheit richtig!

Laufzeit/Lösungsgüte push: O(1), pop O(1), ismt: O(1), suchen: O(n)

Thomas Briner, brinerth@iiic.ethz.ch

Queue in Liste

Datenstruktur

Idee/Vorgehen

FiFo

Zwei POINTER, first, last
push: bei last einhängen
pop: bei first einhängen
ismt? first=NIL?
init: NEW (Q^.first); NEW (Q^.last);


```
TYPE Elem=POINTER TO Elemdesc
     Elemdesc=RECORD
          next: Elem;
          END;
     Q=POINTER TO Qdesc;
     Qdesc=RECORD
          first:Elem;
          last: Elem;
          END;
```

Laufzeit/Lösungsgüte push: O(1), pop O(1), ismt: O(1), suchen: O(n)

Thomas Briner, brinerth@iiic.ethz.ch

Quicksort

Algorithmus

Idee/Vorgehen

Ungeordnetes ARRAY
- Pivot-Element wird ausgewählt;
- mit 2 Scannern von rechts und links über ARRAY fahren, bis je ein Element gefunden wird, das auf falscher Seite von Pivot steht --> vertauschen, beim Pivot teilen,
- rekursiv aufrufen

```
PROCEDURE Quicksort( VAR A:ARRAY; l,r: INT);
    ... IF r>l THEN
            i:=l-1; j:=r; v:=A[r];
            LOOP
                REPEAT INC (i) UNTIL A[i]>=v;
                REPEAT DEC(j) UNTIL a[j]<=v;
                IF i>=j THEN EXIT; END;
                temp:= A[i]; A[i]:= A[j]; A[j]:= temp;
            END;
            t:=A[i]; A[i]:= A[r]; A[r]:=temp;
            Quicksort (A, l, i-1);
            Quicksort (A, i+1, r);
        END;....
```

nicht stabil!

Vorsicht: Diese Zusammenfassung ist nicht mit Sicherheit richtig!

Laufzeit/Lösungsgüte O (n log n)

Thomas Briner, brinerth@iiic.ethz.ch

Römische Zahl umwandeln

Algorithmus

Idee/Vorgehen

Immer noch die vorhergehende Zahl anschauen

Falls A[i-1]<A[i] THEN sum:=sum+A[i]-a[i-1]; i:=i+2;
Sonst sum:=sum+A[i-1]; INC (i)

Laufzeit/Lösungsgüte

Thomas Briner, brinerth@iiic.ethz.ch

Rucksackproblem

Algorithmus

Idee/Vorgehen

n Gegenstände, Gewichtslimite L, Werte w[i] imd Gewochte g[i] für jeden Gegestand sind gegeben.

1. Gewicht diskretisieren, in zB 100g Päckchen aufteilen.
2. Matrix, senkrecht: Gegenstände, waagrecht: Gewichtszähler.
3. Zum Anfangen: überall 0 einfüllen (= kein Gegenstand gewählt --> kein Wert erzielt)
4. Nun beginnt man bei Zeile 1, wird Ggstand gewählt: um g[i] nach rechts und dort neuer Wertestand eintragen; nicht gewählt: von oberer Zeile übertragen,
5. Es wird immer das Optimum dieser beiden Möglichkeiten gewählt.

Beste Packung: unten rechts beginnen, nach links wandern, bis keine 0 mehr steht,

falls oberhalb gleiche Zahl: Ggstand nicht gewählt, falls nicht: g[i] Schritte nach linksgehen, Ggstand gewählt, eins nach oben

Laufzeit/Lösungsgüte O (n* G) pseudopolynomiell

Thomas Briner, brinerth@iiic.ethz.ch

Selbstanordnende Bäume Move to Root

Datenstruktur

Idee/Vorgehen

Der zugegriffene Knoten wird mit Rotationen neu an die Wurzel befördert.

Ergibt einen degenerierten Baum, O(n)

sehr schlääääääächt!

Laufzeit/Lösungsgüte

Thomas Briner, brinerth@iiic.ethz.ch

Selbstanordnende Bäume Splay-Trees

Tarjan

Datenstruktur

Idee/Vorgehen

Idee gleich wie bei Move to Root, aber:
Bei Situation x und Zugriff auf z wird die ganze Gruppe / aufs Mal gedreht.

```
    y
   /
  z
```

--> statisch optimal, gleich gut wie optimaler Suchbaum bei bekannten Zugriffshäufigkeiten

--> degenerieren nicht, sind gleichgut balanciert wie AVL, immer O (log n)

Falls Suche erfolgreich: key in Wurzel
Falls Suche erfolglos: letzter Vater vor NIL in Wurzel

Join: (alle key in TB 1 kleiner als in TB 2)
- Suche nach Max in TB 1, nach oben splayen
- in TB 2 unten einhängen

Split:
- nach Trennpunkt suchen
- an Wurzel teilen

Insert:
- Suche nach x
- falls nicht vorhanden: Split,
- x einsetzen, Join

Delete:
- Suche nach x
- Wurzel weg
- Teilbäume Join

Vorsicht: Diese Zusammenfassung ist nicht mit Sicherheit richtig!

Laufzeit/Lösungsgüte Alles in O(log n)

Thomas Briner, brinerth@iiic.ethz.ch

Selbstanordnende Liste Frequency count

Datenstruktur

Idee/Vorgehen

Lineare Liste, jedes Elem hat Zugriffszähler,
bei Zugriff wird Zähler um 1 erhöht,
wird die ganze Zeit neu nach Zugriffshäufigkeit angeordnet

```
TYPE  Elem=POINTER TO Elemdesc
      Elemdesc=RECORD
            next:Elem;
            freq: INTEGER;
            END;
```


Laufzeit/Lösungsgüte

Thomas Briner, brinerth@iiic.ethz.ch

Selbstanordnende Liste Move to Front

Datenstruktur

Idee/Vorgehen

Lineare Liste,
nach jedem Zugriff wird das zugegriffene Element an die vorderste Position gebracht,

bei Worst Case: Auch O(n)

kompetitive Analyse: Vergleich mit idealem Algorithmus

amortisierte Analyse:
balance = # Inversionen zwischen
imaginärem Gegner und mtf

???

Laufzeit/Lösungsgüte

Thomas Briner, brinerth@iiic.ethz.ch

Selbstanordnende Liste Transpose

Datenstruktur

Idee/Vorgehen

Lineare Liste,
bei jedem Zugriff wird das zugegriffene Element mit dem vorhergehenden vertauscht
(d.h. um eine Position nach vorne geschoben)

Vorteil: Funktioniert auch im ARRAY ohne Schwierigkeiten

Bei Worst Case O(n)
(immer auf hinterstes El zugreifen)

Laufzeit/Lösungsgüte

Thomas Briner, brinerth@iiic.ethz.ch

Set-Cover

Algorithmus

Idee/Vorgehen

S = Gesamtmenge mit n Elementen
T1, T2, T3, ... Teilmengen von S mit verschiedenen Anzahl von Elementen.

Suche minimale # Teilmengen, die alle Elemente aus S abdecken.

Nimm immer die Menge mit am meisten noch nicht abgedeckten Elementen

für Lösungsgüte: Elementkosten für Teilmenge berechnen: 5 noch nicht abgedeckte Elemente --> Preis pro Element = 1/5

via Durchschnittspreis, Summe der Elementenpreise ergibt sich

Lösungsgüte = 1 + log (n)

Laufzeit/Lösungsgüte

Thomas Briner, brinerth@iiic.ethz.ch

Skip-Listen

Datenstruktur

Idee/Vorgehen

Zeiger von verschiedener Höhe,
Zeiger = 2n -1= n+n/2+n/4+...

Suchstrategie:
oberste Ebene beginnen,
nach rechts, falls zu gross:
eine Ebene tiefer nach rechts;
falls zu klein: gleiche Ebene einen Pfeil weiter

randomisiert:
neues Element einfügen:
- richtige Stelle suchen
- Höhe zufällig wählen (Random Uniform, falls >0.5 dann ein Niveau hinauf)
- Zeiger nachführen
(beim Suchen Weg registrieren)

Laufzeit/Lösungsgüte Einfügen: O(log n); Suchen O (log n)

Thomas Briner, brinerth@iiic.ethz.ch

Skirental-Problem

Algorithmus

Idee/Vorgehen

Ausrüstung kaufen oder mieten

Kauf: Fr. 300.-
Miete: Fr 30.-/Tag

i Tage mieten, dann Kaufen --> i*30 + 300

Optimum: = Min(i*30, 300)

competitive ratio

für zB kaufen am 1. Tag, nachher aufgeben:
c.r.= 300/30 =10

für zB 5 Tage mieten, dann kaufen, 6. Tag aufgeben
c.r.= (5*30 + 300)/180 =2.5

Laufzeit/Lösungsgüte

Thomas Briner, brinerth@iiic.ethz.ch

Sortieren vorsortierter Folgen

Algorithmus

Idee/Vorgehen

Für Folgen, die schon nach einem bestimmten Mass vorsortiert sind, kann es Algorithmen geben,
die schneller als die Worst Case Zahl O(n log n) sind.

zB. A-Sort für eine Folge mit wenigen Inversionen

Laufzeit/Lösungsgüte

Thomas Briner, brinerth@iiic.ethz.ch

Stabile Heirat

Gusfield

Algorithmus

Idee/Vorgehen

Jeder Mann, jede Frau hat Prioritätenliste,
zB a: 1 , 2, 3, 4
b: 1, 3, 4, 2

1: c, b, d, a
2: d, a, c, b

(a, 1); (b, 2) ist nicht stabil: (b, 1) würde von b und von1 bevorzugt.

Gesucht: stabile Heirat

Algorithmus, der immer terminiert:
zuerst bei neuem Mann Prioritätenliste durchschauen bis zur 1. Frau,

falls noch nicht vergeben: zuordnen,
sonst: bei Frau kontrollieren, ob Mann vor dem eigentlich schon zugeordneten Partner kommt.

Wenn ja, dann Paar neu bilden,
alten Eintrag löschen, bei altem Mann um eine Position weiter.

Wenn nein, dann bei neuem Mann eine Position weiter

Laufzeit/Lösungsgüte

Thomas Briner, brinerth@iiic.ethz.ch

Stapel in Array, beschränkte Grösse

Datenstruktur

Idee/Vorgehen **FiLo**

push: Immer hinten anhängen,
Zähler für hinterstes Element mitführen
pop: Hinterstes Element ausgeben und löschen,
DEC(Zähler)
ismt?: Zähler auf 0?
Init: Neues Array initialisieren
full?: Zähler=n-1?

Laufzeit/Lösungsgüte push: O(1), pop O(1), ismt: O(1), suchen: O(n)

Thomas Briner, brinerth@iiic.ethz.ch

Stapel in Liste

Datenstruktur

Idee/Vorgehen

FiLo

Lin. Liste, Stack mit POINTER top;
Elemente als ADT mit Zeiger next;
vorne einhängen/aushängen
ismt?: top-POINTER =NIL?
init: NEW (S^.top);

```
TYPE  Element=POINTER TO Elementdesc;
      Elemdesc=RECORD
            next:Element;
            END;
      Stack=POINTER TO Stackdesc;
      Stackdesc=RECORD
            top:Element;
            END;

PROCEDURE push* (VAR S:Stack; e:Element);
      BEGIN
            e^.next:=S^.top;
            S^.top:=e;
      END push;
```

Vorsicht: Diese Zusammenfassung ist nicht mit Sicherheit richtig!

Laufzeit/Lösungsgüte push: O(1), pop O(1), ismt: O(1), suchen: O(n)

Thomas Briner, brinerth@iiic.ethz.ch

Star in der Menge

Algorithmus

Idee/Vorgehen

Star: Alle kennen ihn, er kennt niemanden

naiv: in Adjazenzmatrix alles kontrollieren n*(n-1)
O(n^2)

mit Induktion:
3 Fälle: Star in bisheriger Menge, neuer ist Star, oder es gibt keinen, je nachdem jeder aus schon bearbeiteten 2 Fragen stellen (kennst du ihn ? Kennt er dich? 2*(n-1)+2*(n-2)....
O(n^2)

Duellvariante:
immer 2 wählen, Kennt A B? Ja -->A nicht Star, Nein-->B nicht Star, bei letztem potentiellem Star Zeile und Spalte der Matrix kontrollieren
(n-1)+2*(n-1) ---> O(n)

Laufzeit/Lösungsgüte

O(n)

Thomas Briner, brinerth@iiic.ethz.ch

Sternfiguren-Fraktale

Algorithmus

Idee/Vorgehen

5-zackiger Stern, in jeder Ecke soll wieder ein Stern eingefügt werden usw.
Eingabe n: # Iterationen

```
PROCEDURE Stern(n,Seitenlänge:INTEGER)
        BEGIN
        FOR i:=1 TO 5 DO
              Turtle.Go (Seitenlänge);Turtle.Turn (72);
               IF n>0 THEN Stern (n-1, Seitenlänge/2);
              END;
        END;
```


Laufzeit/Lösungsgüte

Thomas Briner, brinerth@iiic.ethz.ch

Suchbaumtraversen

Algorithmus

Idee/Vorgehen

Preorder: (HauptRF)
- Wurzel
- linker TB
- rechter TB

Inorder: (Symm. RF)
- linker TB
- Wurzel
- rechter TB

--> ergibt aufsteigende Reihenfolge

Postorder: (NebenRF)
- linker TB
- rechter TB
- Wurzel

Laufzeit/Lösungsgüte

Thomas Briner, brinerth@iiic.ethz.ch

Teleskopieren

Theorie

Idee/Vorgehen

T(2) = 1
T(n) = T(n/2)+T(n/2)+2
= 2*T(n/2)+2
= 2*(2*T(n/4)+2)+2 = 4*T(n/4)+4+2
= 4*(2*T(n/8)+2)+4+2
= 8*T(n/8)+8+4+2
= 2^i*T(n/2^i)+2^i+2^(i-1)+....+2

Wann hört auf? n/2^i:=2 --> n/2 = 2^i

= n/2*T(n/(n/2))+n/2 + n/4 + n/8 +....+ 2
= n/2*1 + n-2

= 3/2*n + 2

ist aber noch kein Beweis. Jetzt mit Induktion drüber!

Vorsicht: Diese Zusammenfassung ist nicht mit Sicherheit richtig!

Laufzeit/Lösungsgüte

Thomas Briner, brinerth@iiic.ethz.ch

Text Pattern Matching

Knuth/Morris/Pratt, Boyer/Moore, Rabin/Karp

Algorithmus

Idee/Vorgehen

Text der Länge n, Pattern der Länge m, n>>m
Ist Pattern im Text enthalten?

Naiv: Von vorne nach hinten, immer ganzes Pattern kontrollieren
--> Laufzeit m*n

Knuth/Morris/Pratt: Hilfsstruktur Verschiebungsarray der Länge m vorher berechnen zB wenn an Pos 5 ein Mismatch auftritt, so haben alle vorangehenden Positionen gestimmt --> in Pos 5 des Verschiebungsarrays steht, um wieviel Positionen man das Muster nun schieben kann, bei denen sicher die Lösung nicht übergangen wird.

Boyer/Moore: Pattern an den Anfang des Texts, Von hinten im Pattern beginnen, falls Mismatch, kontrollieren, ob falscher Buchstabe überhaupt in Pattern vorkommt, sonst um Länge n schieben

Rabin/Karp: Hashfunktion des Patterns berechnen, zB ORD(Pos1)*B^2 +ORD(Pos2)*B+ORD(Pos3), dann vom Text das gleiche berechnen, Verschiebung um eines lässt sich inkrementell berechnen:
h neu= (h alt - ORD(Pos1)*B^2)*B + ORD(Pos4)

Laufzeit/Lösungsgüte

Thomas Briner, brinerth@iiic.ethz.ch

Topologisches Sortieren

Algorithmus

Idee/Vorgehen

Finde eine Reihenfolge, so dass alle Pfeile im gerichteten Graphen befolgt werden.
Bedingung: kein Zyklus

Verankerung: Wenn nur 1 Knoten im Graph, bekommt er die Nummer n

Annahme: {1, 2, ..., n-1} schon bekannt

Schritt: 1 Knoten im Graph weglassen, von dem nur Pfeile weggehen, bekommt die Nr. 1; übrige Knoten Nr. 2 - n nummerieren; wiederholen

Implementation:
Adjazenzliste, alle Knoten durchgehen, auf welchen zeigt kein Pfeil? --> X auf Stack
Pfeile, die von X weggehen, löschen

O(|V| + |E|) ??

Laufzeit/Lösungsgüte

Thomas Briner, brinerth@iiic.ethz.ch

Transitive Hülle

Algorithmus

Idee/Vorgehen

Falls Verbindung von A nach B und von B nach C
--> C ist auch von A zu erreichen

Trans. Hülle: Finde alle Erreichbarkeiten

Adjazenzmatrix, Diagonal 1en einfügen, multipl. ergibt Wege der Länge 2

Induktion über Weglänge:
immer an grösstem Zwischenknoten k aufteilen

Annahme: Wege mit Zwischenknoten < k sind schon bekannt

Schritt: k dazunehmen, neuer Weg muss k benutzen

Laufzeit/Lösungsgüte O(n^3)

Thomas Briner, brinerth@iiic.ethz.ch

Transitive Hülle

Warshall

Algorithmus

Idee/Vorgehen

Welche Knoten sind von einem Startknoten alle erreichbar?

Adjazenzmatrix, Diagonalen 1 setzen, Induktion

Annahme: Alle Wege über Zwischenknoten bis Nr. j sind schon bekannt. Nun kommt noch j+1 dazu.

```
FOR i:=1 TO NofNodes DO A[i,i]:=1;
FOR j:=1 TO NofNodes DO
        FOR i:=1 TO NofNodes DO
                IF A[i,j]=1 THEN
                        FOR k:=1 TO NofNodes Do
                                IF A[j,k]=+ THEN
                                A[i,k]:=1; END;
                        END;
                END;
        END;
END;
```


Laufzeit/Lösungsgüte O(n^3)

Thomas Briner, brinerth@iiic.ethz.ch

Travelling Salesman Problem (Dreiecksungl.)

Algorithmus

Idee/Vorgehen

Approximation mit MST (gilt nur, wenn Dreiecksungleichung AB + AC >= BC gilt)

MST berechnen,
aussen herum entlangfahren, wenn Knoten schon besucht, dann überspringen und Abkürzung zum nächsten noch nicht besuchten Knoten eintragen

| MST | < | Rundreise minus 1 Kante | < | opt TSP |

ohne Abkürzungen wäre der app TSP = 2 * MST

wegen Dreiecksungleichung: mit Abkürzungen sicher nicht schlechter als ohne

--> Lösungsgüte <=2

Variante: MST erstellen,
Knoten mit deg = gerade --> unverändert
Knoten mit deg = ungerade --> Kante angehängt, die immer 2 solche Knoten verbindet.

Lösungsgüte = 1,5

Vorsicht: Diese Zusammenfassung ist nicht mit Sicherheit richtig!

Laufzeit/Lösungsgüte

Thomas Briner, brinerth@iiic.ethz.ch

Türme von Brahma

Algorithmus

Idee/Vorgehen

3 Stäbe, n Scheiben, müssen von A nach B mit Hilfe von C;
Rekursiv lösen:
Falls nur 2 Scheiben vorhanden sind:
kleine Scheibe auf C, grosse Scheibe (= n-1 restliche Scheiben) auf B, kleine Scheibe auf B drauf.

Lässt sich so für n Scheiben rekursiv lösen:

```
PROCEDURE Brahma (n: INTEGER; A,B,C :Säulen );
BEGIN
        IF n=1 THEN ...
        ELSE Brahma (n-1,....
```


Laufzeit/Lösungsgüte

Thomas Briner, brinerth@iiic.ethz.ch

Typerweiterung

Theorie

Idee/Vorgehen

RECORD und POINTER können in Oberon erweitert werden.

```
TYPE  Adr1= RECORD
              Nummer:INTEGER; END;
      Adr2= RECORD (Adr1)
              Name: ARRAY OF CHAR; END;

VAR   a: Adr1; b: Adr2;
```

Jetzt hat a nur das Feld a.Nummer, b hingegen die Felder b.Nummer und b.Name (dh. b ist Erweiterung).

a:=b; ist gültig, alle Felder von a sind in b auch definiert, die zusätzlichen Infos gehen in a verloren

b:=a; ist ungültig, b hat Felder, die nicht gefüllt werden können. Bei anfrage b.Name hats nichts dort.

Vorsicht: Diese Zusammenfassung ist nicht mit Sicherheit richtig!

Laufzeit/Lösungsgüte

Thomas Briner, brinerth@iiic.ethz.ch

Union-Find

Datenstruktur

Idee/Vorgehen

verwalten von Mengen
Jedes Element zeigt auf ein anderes, das zur gleichen Menge gehört, bis zum „Wurzelelement“, das auf sich selber zeigt.

Find: Pfeilen entlang, bis ein Element auf sich selber zeigt --> Wurzel

Union: Die kleinere Menge wird in die grössere Menge eingehängt, d.h. die Wurzel der kleineren Menge zeigt neu auf die Wurzel der grösseren Menge

Nachteil: kann grosse Pfadlänge=Suchzeit ergeben

Vorsicht: Diese Zusammenfassung ist nicht mit Sicherheit richtig!

Laufzeit/Lösungsgüte

Thomas Briner, brinerth@iiic.ethz.ch

Union-Find Pfadverkürzung

Algorithmus

Idee/Vorgehen

path splitting:

Beim Find wird jeder Knoten auf dem Weg mit seinem Grossvater verbunden. Dadurch wird der Pfad in 2 separate Wege aufgeteilt.

path halving:

Es wird beim Find nur jeder 2. Knoten auf dem Weg zur Wurzel mit seinem Grossvater verbunden. Dadurch ergibt sich ein verästelter halb so langer Weg

Laufzeit: Ackermannfkt alpha, viel kleiner als log

collapse:

Alle Knoten werden direkt an Wurzel angehängt. Dafür muss der Weg zweimal zurückgelegt werden: 1. Mal um die Wurzel zu finden, 2. Mal zum Einhängen

Laufzeit/Lösungsgüte Find O(alpha (m,n))

Thomas Briner, brinerth@iiic.ethz.ch

Union-Find Pfadverkürzung collapse

Datenstruktur

Idee/Vorgehen

Jedes Element wird direkt an der Wurzel angehängt.

--> grosser Zeitgewinn beim Find, O(1)
--> Riesenaufwand bei Union, alle Pfeile müssen
umgehängt werden O(n*log n)

--> nicht so toll

Laufzeit/Lösungsgüte

Thomas Briner, brinerth@iiic.ethz.ch

Untere Schranke für Sortierverfahren

Theorie

Idee/Vorgehen

Vergleichsbasierte Sortierverfahren basieren auf einem Entscheidungsbaum, an jedem Knoten eine Entscheidung „a1> a2?“ --> Pfade führen zu allen möglichen Permutationen in den Blättern

Permutiationen für n Objekte: n!

Suchbaumpfade nicht alle gleich lang; kürzester Pfad (a1 < a2 < .... < an) ist mindestens n-1 lang (mit Transitivität kann nach a1 < a2, a2 < a3 gefolgert werden a1 < a3).
Maximale Pfadlänge:
Auf einer Ebene sind 2^höhe # Blätter --> alle Permutationen, falls auf der gleichen Höhe:
2^n = n!, mit Stirlingscher Formel --> maximal mögliche Pfadlänge O (log n) --> untere Schranke für Sortieralgorithmus:
O (n log n)

Laufzeit/Lösungsgüte O (n log n)

Thomas Briner, brinerth@iiic.ethz.ch

Wächter im Graph (Vertex Cover)

Algorithmus

Idee/Vorgehen

Wie viele Wächter müssen auf die Knoten positioniert werden, damit jede Kante überwacht wird?

Idee 1: Zuerst die Knoten mit den höchsten Graden, alle Kanten, die dadurch überwacht werden wegstreichen

Lösungsgüte: log(n)

Idee 2: Wähle Knoten v, gehe einer Kante entlang und wähle auch noch Nachbarknoten w. Nun entferne alle Kanten, die dadurch abgedeckt sind

Lösungsgüte: 2

Grund: in der optimalen Lösung müsste ich auch entweder Knoten v oder Knoten w wählen. Ich habe also mit diesem Algorithmus höchstens das doppelte erwischt.

Laufzeit/Lösungsgüte

Thomas Briner, brinerth@iiic.ethz.ch

Weg finden in dag

Algorithmus

Idee/Vorgehen	Breitensuche Knoten in Queue speichern

Laufzeit/Lösungsgüte

Thomas Briner, brinerth@iiic.ethz.ch

Zahlen in 4 Grp sortieren, Dijkstra

Algorithmus

Idee/Vorgehen

N-1 positive REAL, in 4 Grp einteilen,
0 < o <=1; 1 < p<=5<, 5 < q <=50; r> 50;
jede Zahl nur einmal anschauen,
kein zusätlicher Speicherplatz

x,i,j,k:=0,0,0,N-1

```
do      0<A[x]<=1 --> swap x,i; i:=i+1; j:=j+1; x:=x+1
        1<A[x]<=5 --> swap x,j;j:=j+1, x:=x+1
        5<A[x] <=50 --> x:=x+1;
        50<A[x] --> swap x,k; k:=k-1; x:=x+1
od
```

Vorsicht: Diese Zusammenfassung ist nicht mit Sicherheit richtig!

Laufzeit/Lösungsgüte

Thomas Briner, brinerth@iiic.ethz.ch

Ziege vor dem Zaun, Exploration

Algorithmus

Idee/Vorgehen

suchen nach bester Strategie,

zB immer pendeln: 1 Haus links, 2 Häuser rechts, 3 Häuser links etc.

competitive ratio = (1 + 2 + 3 + 4 + ...d) / d =~ d

besser: immer Distanz verdoppeln

Laufzeit/Lösungsgüte

Thomas Briner, brinerth@iiic.ethz.ch