Technische Universität Braunschweig

Institute of Operating Systems and Computer Networks

Algorithmen und Datenstrukturen
Große Übung #5

Phillip Keldenich, Arne Schmidt

24.11.2016

Heute: Kombinatorische Probleme

Kombinatorische Probleme - Allgemein

Gegeben

- Objekt M
- Beschreibung der Lösungsmenge L (abhängig von M)
- Prädikat $P : L \to \{true, false\}$ (gültige Lösung?)
- Funktion $f : L \to \mathbb{R}$ (Wert der Lösung)

Gesucht Lösung $x \in L$ mit $P(x) = true$ und $f(x)$ minimal.

Kombinatorische Probleme - Lösungsansatz

Naiver Ansatz: *Brute Force*

function BRUTEFORCE(M, L, P, f)
    $val \leftarrow \infty$
    OPT $\leftarrow$ *NIL*
    for $x \in L$ **do**
        if $P(x) \wedge f(x) < val$ **then**
            OPT $\leftarrow x$
            $val \leftarrow f(x)$
        end if
    end for
end function

Kurzfassung: Überprüfe jede mögliche Lösung und nimm die beste.
Laufzeit? $\Theta(|L|)$.

Kombinatorische Probleme - Sortieren

M Array der Länge n

L Permutationen von M

$P(x)$ Ist *true* $\Leftrightarrow x_1 \leq \cdots \leq x_n$

$f(x)$ Ist 0 für alle $x \in L$

Laufzeit von BruteForce: $\Theta(|L|) = \Theta(n!)$

Kombinatorische Probleme - Kürzester Weg in Graphen

M Graph (V, E)

L Potenzmenge von E (kurz: 2^E oder $\mathcal{P}(E)$)

$P(x)$ Ist *true* $\Leftrightarrow$ x ist ein Weg von p nach q.

$f(x)$ $= |x|$, also die Länge des Weges

Laufzeit von BruteForce: $\Theta(|L|) = \Theta(2^{|E|})$

Kombinatorische Probleme - TSP

Wir betrachten einen vollständigen Graphen $G = (V, E)$ mit Gewichten $c : E \to \mathbb{R}^+$ und suchen eine gewichtminimale Rundreise.

M Graph (V, E)

L Permutationen von V

$P(x)$ Ist *true* für alle $x \in L$

$f(x) = \sum_{i=1}^{|V|} c(\{x_i, x_{(i+1) \bmod n}\})$, also das Gewicht der Rundreise

Laufzeit von BruteForce: $\Theta(|L|) = \Theta(|V|!)$

Kombinatorische Probleme - Laufzeiten

Wie schnell sind diese Probleme lösbar?

- Sortieren: $\Theta(n \log n)$, mit Mergesort
- Kürzeste Wege: $\Theta(|V| + |E|)$, mit BFS
- TSP: ?? (bekannt: $O(2^n n^2)$ mit *Dynamic Programming*($\rightarrow$ AuD 2))

Entscheidend ist die Struktur der zulässigen Lösungen ($P(x)$) und nicht die Größe von L.
Zum Beispiel sind Teile eines kürzesten Weges wieder kürzeste Wege.

Wie ist das mit TSP? Gibt es eine Struktur die sich gut fassen lässt?
(Vermutlich nicht $\rightarrow$ siehe P vs. NP)

Divide & Conquer

Prinzip *Teilen und Herrschen*: Löse kleinere Probleme, um Große zu lösen. Ein Divide-and-Conquer Algorithmus besteht aus drei Teilen:

- Teilen falls das Problem zu groß ist (Wo? Wie? Wie oft?)
- Lösen von Teilproblemen (Rekursion)
- Kombinieren von Teillösungen (Wie?)

Divide & Conquer - Mergesort

Konkret bei Mergesort:

- Teile einmal in der Mitte des Arrays falls mehr als ein Element existiert
- Sortiere linke und rechte Hälfte (Rekursion) oder gib Array zurück, falls nur ein Element existiert.
- Merge beide Hälften! (Gleich an der Tafel!)

Laufzeit von Mergesort ist $\Theta(n \log n)$ (siehe VL)

Divide & Conquer - Multiplikation

Gegeben: Zwei Zahlen x, y in Dezimaldarstellung mit jeweils n Ziffern.
Gesucht: Produkt von $x \cdot y$.
„Schriftliche Multiplikation“ liefert einen Algorithmus mit Laufzeit $O(n^2)$. Divide-and-Conquer schafft das schneller!

function $\mathrm{M}(x, y)$
  if $n = 1$ **then**
    return $x \cdot y$
  end if
  $k \leftarrow \frac{n}{2}$
  Wähle A, B, C und D mit $x = A \cdot 10^k + B$ und $y = C \cdot 10^k + D$
  return $\mathrm{M}(A, C) \cdot 10^{2k} + (\mathrm{M}(A, D) + \mathrm{M}(B, C)) \cdot 10^k + \mathrm{M}(B, D)$
end function

Divide & Conquer - Multiplikation

function $\mathrm{M}(x, y)$
  if $n = 1$ **then**
    return $x \cdot y$
  end if
  $k \leftarrow \frac{n}{2}$
  Wähle A, B, C und D mit $x = A \cdot 10^k + B$ und $y = C \cdot 10^k + D$
  return $\mathrm{M}(A, C) \cdot 10^{2k} + (\mathrm{M}(A, D) + \mathrm{M}(B, C)) \cdot 10^k + \mathrm{M}(B, D)$
end function

Sei $T(n)$ die Laufzeit von $\mathrm{M}(x,y)$. Dann ist

$$T(n) = 4 \cdot T(\frac{n}{2}) + c \cdot n$$

wobei in $c \cdot n$ alle Rechenkosten der Addidtion enthalten sind.

Divide & Conquer - Multiplikation

Sei $T(n)$ die Laufzeit von $\mathrm{M}(x,y)$. Dann ist

$$T(n) = 4 \cdot T(\frac{n}{2}) + c \cdot n$$

wobei in $c \cdot n$ alle Rechenkosten der Addidtion enthalten sind.

Wie löst man solche rekursiven Gleichungen? Das sehen wir nächste Woche in der Vorlesung! Mit geschickten Überlegungen kommt man auf $T(n) \in \Theta(n^2)$...?
Das ist nicht besser wie die schriftliche Multiplikation!

Divide & Conquer - Multiplikation

Was können wir tun, um die Laufzeit zu verbessern?

- Weniger Additionen? $\rightarrow$ ändert das c, aber nicht die Komplexität.
- Weniger rekursive Aufrufe? Könnte funktionieren. Aber wie?

Müssen $\mathrm{M}(A, D)$ und $\mathrm{M}(B, C)$ berechnet werden? Es geht anders:
Sei $E = A - B$ und $F = C - D$. Dann ist:

$$AD + BC = (AC + BD) - EF$$

Also ist $x \cdot y = AC \cdot 2^{2k} + (AC + BD - EF) \cdot 2^k + BD$. Da nur drei verschiedene Produkte berechnet werden müssen, müssen wir nur drei rekursive Aufrufe tätigen.
Das gibt uns die neue Laufzeit $T'(n) = 3\,T'(\frac{n}{2}) + c' \cdot n$.
Man kann nun zeigen, dass $T'(n) \in \Theta(n^{1.585})$ ist.