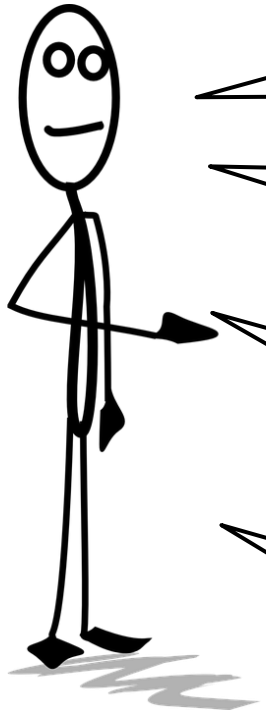
Technische Universität Braunschweig

Algorithmen und Datenstrukturen 2

Arne Schmidt

Kapitel 5 - Komplexität

Laufzeiten zum Lösen von Maximum Knapsack



Greedy $O(n \log n)$
Vorteil: Liefert schnell optimale Lösungen
Nachteil: Optimale Lösungen sind nicht immer da…

Dyn. Programming $O(nZ)$
Vorteil: Liefert immer optimale Lösungen
Nachteil: Langsam!

Branch-and-Bound $O(n2^n)$
Vorteil: Liefert immer optimale Lösungen
Nachteil: Langsam!

Greedy-k $O(n^{k+2})$
Vorteil: Liefert immer gute Lösungen
Nachteil: selten optimal!

Good, Fast, Cheap

Also…


Bisher nur Algorithmen, die…
1. „Schnell und optimal, aber nicht immer“
2. „Schnell und immer gut, aber nicht optimal“
3. “Immer optimal, aber nicht schnell“
Gibt’s denn keine bessere Lösung?

Fangen wir ganz Allgemein an

Gibt es Probleme, die einen Algorithmus besitzen, die „**Immer** und **schnell** eine **optimale** Lösung“ bestimmen können?

Kapitel 5.1 – Die Klassen P und NP

Sortieren:
- Lösbar in $O(n \log n)$mit Mergesort
- Wir erhalten eine korrekte Sortierung

→ Schnell, immer und "optimal Zeit".

Aufspannender Baum:
- Lösbar in $O(n + m)$ Zeit mit BFS/DFS
- Wir erhalten einen korrekten Spannbaum

→ Schnell, immer und "optimal".

Eulertouren:
- Lösbar in $O(n + m)$ Zeit mit Hierholzer
- Wir erhalten eine korrekte Eulertour

→ Schnell, immer und "optimal".

Median:
- Lösbar in $O(n)$ Zeit mit Median der Mediane
- Wir erhalten einen korrekten Median

→ Schnell, immer und "optimal".

Die Klassen P und NP

Definition 5.1
Ein algorithmisches Problem Π gehört zur Klasse P, wenn es für Π einen Algorithmus mit polynomieller Laufzeit (in der Inputgröße) gibt, der immer eine optimale Lösung findet.

Definition 5.2
Ein Problem Π gehört zur Klasse NP, wenn für jede Instanz I von Π die Gültigkeit einer Lösung in polynomieller Zeit (in der Inputgröße von I) nachgeprüft werden kann.

Bemerkung
In der Regel werden die Klassen P und NP über Entscheidungsprobleme definiert: “Gibt es eine Lösung?”
Bspw ist 0-1 Knapsack ein Entscheidungproblem.

Eine formalere Definition gibt es bspw. in Theoretische Informatik 2

Nachprüfen

{7, 13, 17, 24, 29, 31, 31, 35, 53}

{7, 13, 17, 24, 29, 31} {31, 35, 53}

Antwort: Keine Lösung!

{13, 17, 24, 31, 35} {7, 29, 31, 53}

Antwort: Lösung!

Nachprüfen ist hier ganz einfach:
Summiere Elemente beider Mengen auf und vergleiche die Summen.

Nachprüfen

{7, 13, 17, 20, 29, 31, 31, 35, 57}

{7, 13, 17, 20, 29, 31} {31, 35, 57}

Antwort: Keine Lösung!

{7, 13, 35, 29, 31} {17, 25, 31, 57}

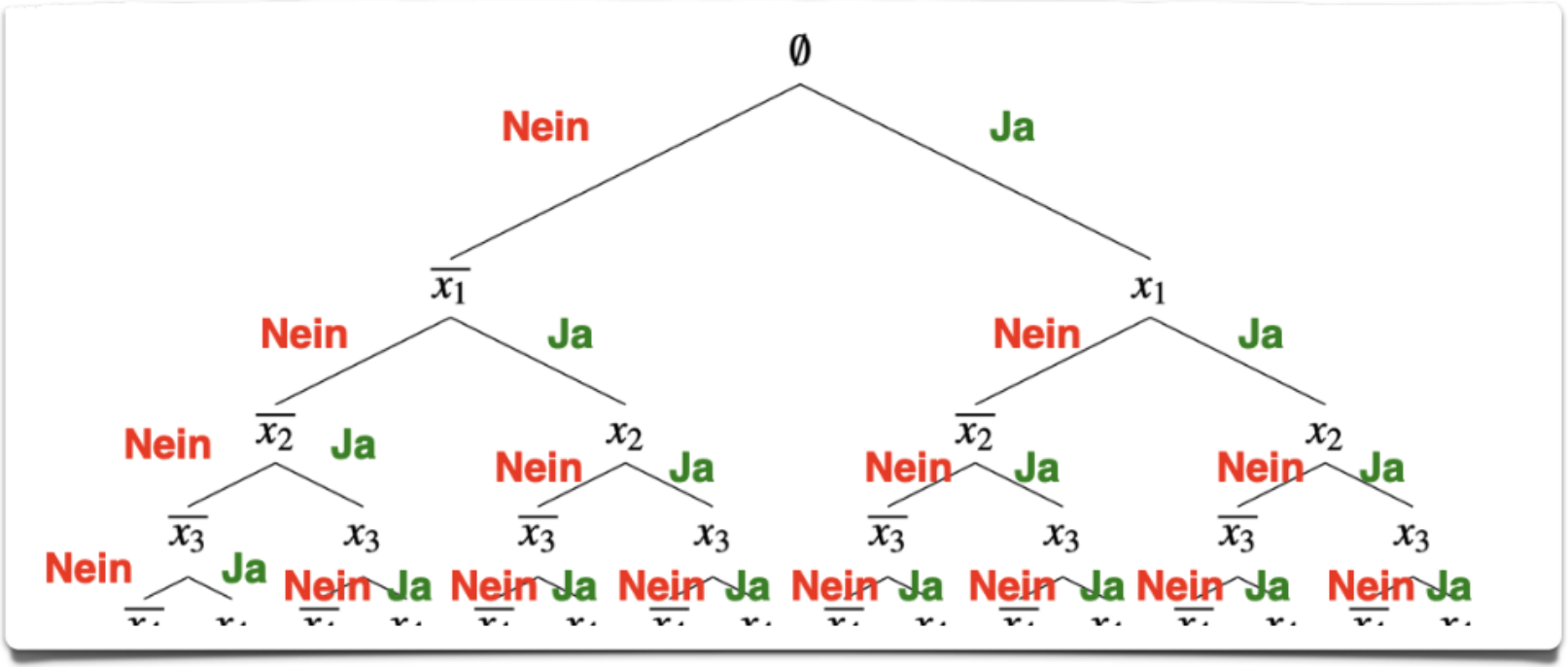
Antwort: Keine Lösung!

Auch hier ist Nachprüfen hier ganz einfach.

Es ist aber egal, welche Lösung wir betrachten. In diesem Fall ist die Antwort immer “keine Lösung”!

Im deterministischen Fall bleibt nichts anderes übrig als alles durchzutesten?!

Enumerationsbaum



Exponentiell viele Fälle!

Kann man irgendwo Arbeit sparen?!

Milleniumproblem

Clay Mathematics Institute

About Programs & Awards People The Millennium Prize Problems Online resources Events News

Home — The Millennium Prize Problems

The Millennium Prize Problems

P vs NP

If it is easy to check that a solution to a problem is correct, is it also easy to solve the problem? This is the essence of the P vs NP question. Typical of the NP problems is that of the Hamiltonian Path Problem: given N cities to visit, how can one do this without visiting a city twice? If you give me a solution, I can easily check that it is correct. But I cannot so easily find a solution.

Beweisbare Schwere eines Problems

Kapitel 5.2 – Ein Beispiel mit Logik (an der Tafel)