Dynamische Programmierung als Problemlösungsstrategie der Informatik

und ihre Anwedung in der Diskreten Mathematik und Graphentheorie

Fabian Cordt Enisa Metovic

Wissenschaftliche Arbeiten und Präsentationen, WS 2010/2011

Gliederung

1. Einleitung
   - Einführung in die Dynamische Programmierung
   - Fibonacci Folge
2. Problem des kürzesten Pfades
   - Problemstellung
   - Dijkstra's Algorithmus
3. Problem des Handlungsreisenden
   - Problemstellung
   - Optimierung mit Dynamischer Programmierung

Der Begriff “Dynamische Programmierung”

- Wurde 1940 von Richard Bellman eingeführt
- Bellman wandte diese Methode auf dem Gebiet der Regelungstheorie an

Einführung in die Dynamische Programmierung

- Divide and Conquer: “Teile und Herrsche”
- Löst ein Problem durch Kombination der Lösungen von Teilproblemen
- Jedes Teilproblem wird nur einmal gelöst
- Zwischenergebnisse werden für späteren Gebrauch in einer Tabelle gespeichert

Optimalitätsprinzip

- Richard Bellman, 1957
- "An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision."
- Jede Teillösung einer optimalen Lösung ist selbst eine optimale Lösung des betreffenden Teilproblems

Entwickeln eines Algorithmus mit Dynamischer Programmierung

Voraussetzung: Die optimale Lösung setzt sich aus optimalen Teillösungen zusammen

- Charakterisieren der Struktur
- Rekursive Definition des Wertes einer optimalen Lösung
- Berechnen des Wertes einer optimalen Lösung
- Konstruktion der optimalen Lösung aus berechneten Informationen

Probleme

- Die Lösung kleinerer Probleme garantiert nicht immer die Lösung eines größeren Problems
- Die Anzahl der zu lösenden Probleme kann beliebig groß werden

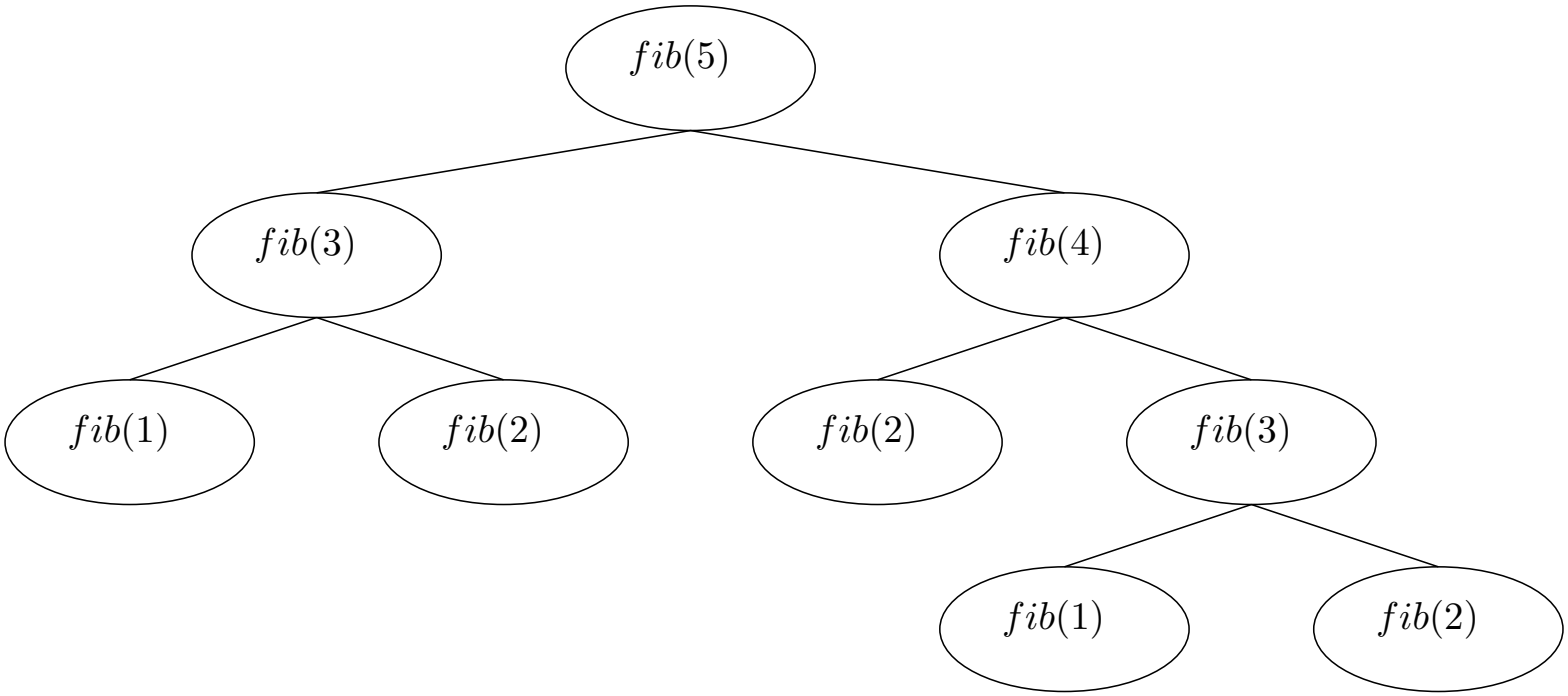
Die Fibonacci Folge

- Eine unendliche Folge von Zahlen, bei der sich die folgende Zahl durch Addition ihrer beiden Vorgänger ergibt
- Rekursive Beschreibung des Problemes:
  - $f_1 := 1$
  - $f_2 := 1$
  - $f_n := f_{n-1} + f_{n-2}$
- $1, 1, 2, 3, 5, 8, 13, \ldots$

Implementierung

```
int fib(int n)
{
  if(n == 1 || n == 2)
    return 1;

  return fib(n-1) + fib(n-2);
}
```

Rekursive Berechnungen



$\Rightarrow$ Exponentieller Aufwand $O(2^n)$

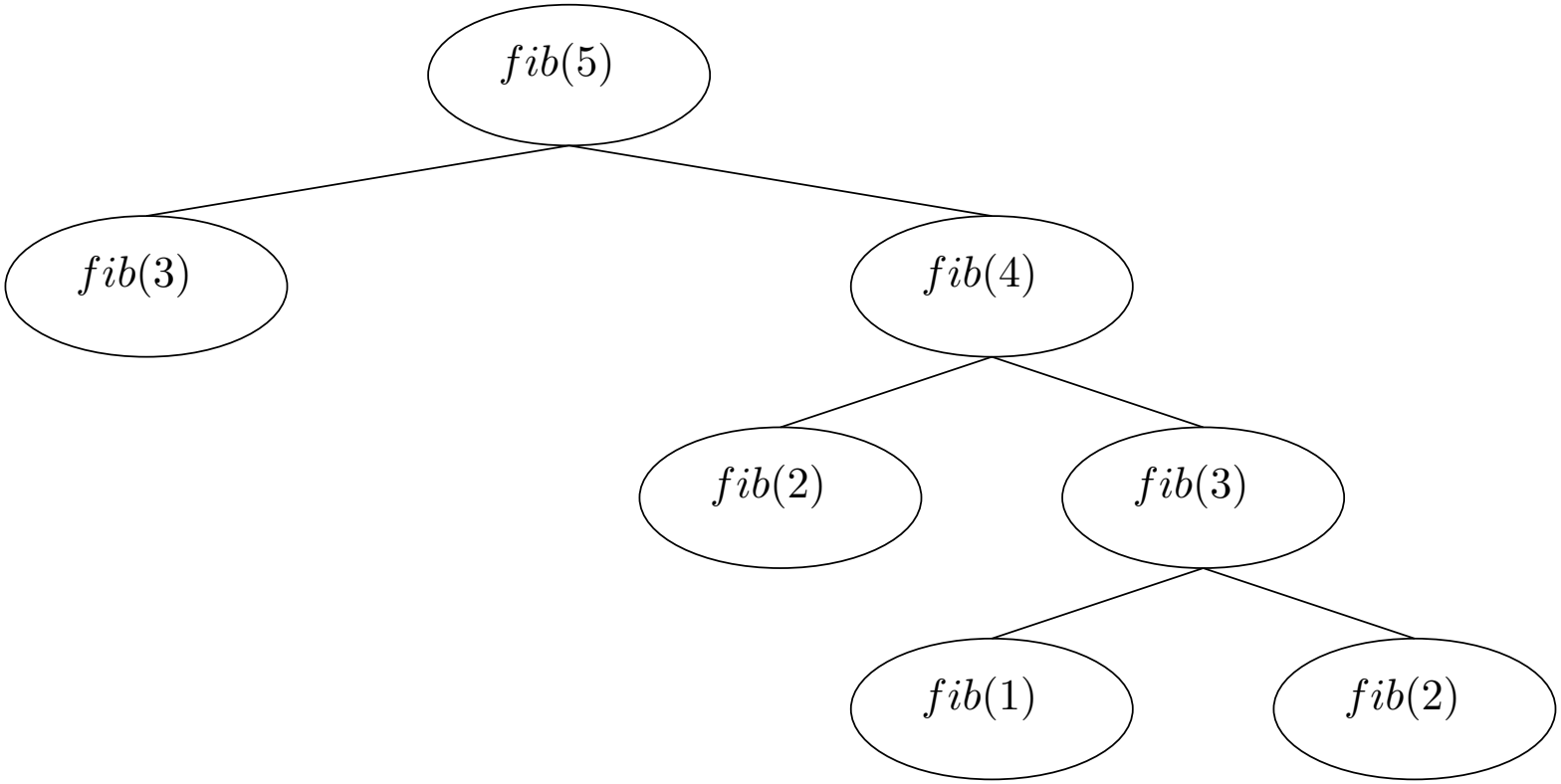
Optimierung mit Dynamischer Programmierung

```
static int knownF[] = new int[1000];

int fib(int n)
{
  if(knownF[n] != 0) {return knownF[n];}

  if(n == 1 || n == 2)
    return 1;

  return (knownF[n] = fib(n-1) + fib(n-2));
}
```

Optimierung mit Dynamischer Programmierung (con't)

- Weiterhin Rekursionen
- Ergebnisse werden zwischengespeichert
- Zuerst "lookup" in Tabelle

Optimierung mit Dynamischer Programmierung (con't)



$\Rightarrow$ Linearer Aufwand $O(n)$

Optimierung mit Dynamischer Programmierung (con't)

- Laufzeit für fib(50) = 12586269025
- Rekursiv: 13 Minuten, 12 Sekunden
- Rekursiv mit Dynamischer Programmierung: 0.004 Sekunden

Problem des kürzesten Pfades

- Den kürzesten Pfad ("Weg") zwischen zwei Punkten finden
- Viele Anwendungen (Navigation, Routing, Spiele)
- Viele Algorithmen
- Zeitkritisch

Rekursive Lösung

- Input
  - Menge V an n Knoten
  - Menge E an Kanten, wobei e_{ij} die Distanz zwischen den Knoten v_i und v_j beschreibt
  - Startknoten v_s, Endknoten v_e
- $p(v_i, K \subseteq V) = \min_{\forall v_k \in K}(p(v_k, K - \{v_k\}) + e_{ki})$
- $p(v_s, K) = 0$
- Um den kürzesten Pfad zu berechnen, muss man also $p(v_e, V - \{v_e\})$ ausführen

Probleme

- Große Menge an Rekursionen
- Redundante Berechnungen
- Überexponentielle Komplexität!
- Lösung: Iterative Berechnung der Wege, ausgehend von dem ersten Knoten
- Zwischenspeichern der bereits besuchten Knoten

Djikstra's Algorithmus

- Setze bei allen Knoten außer dem Startknoten die Distanz auf ∞
- Nimm den Knoten mit der kürzesten Distanz zum Startknoten
- Markiere diesen Knoten als besucht
- Berechne die Distanz aller direkt verbundenen Knoten, die noch nicht besucht wurden zum, Startknoten
- Speichere für diese Knoten den aktuellen Knoten als vorgehenden Knoten, falls die Distanz kleiner ist als die vorher vorhandene
- Führe dieses Verfahren so lange durch, bis alle Knoten besucht wurden

Eigenschaften

- Laufzeit $O(n^2)$
- Falls es einen optimalen Weg gibt, wird dieser gefunden
- Zusätzlich werden die optimalen Wege vom Startknoten zu allen anderen Knoten berechnet

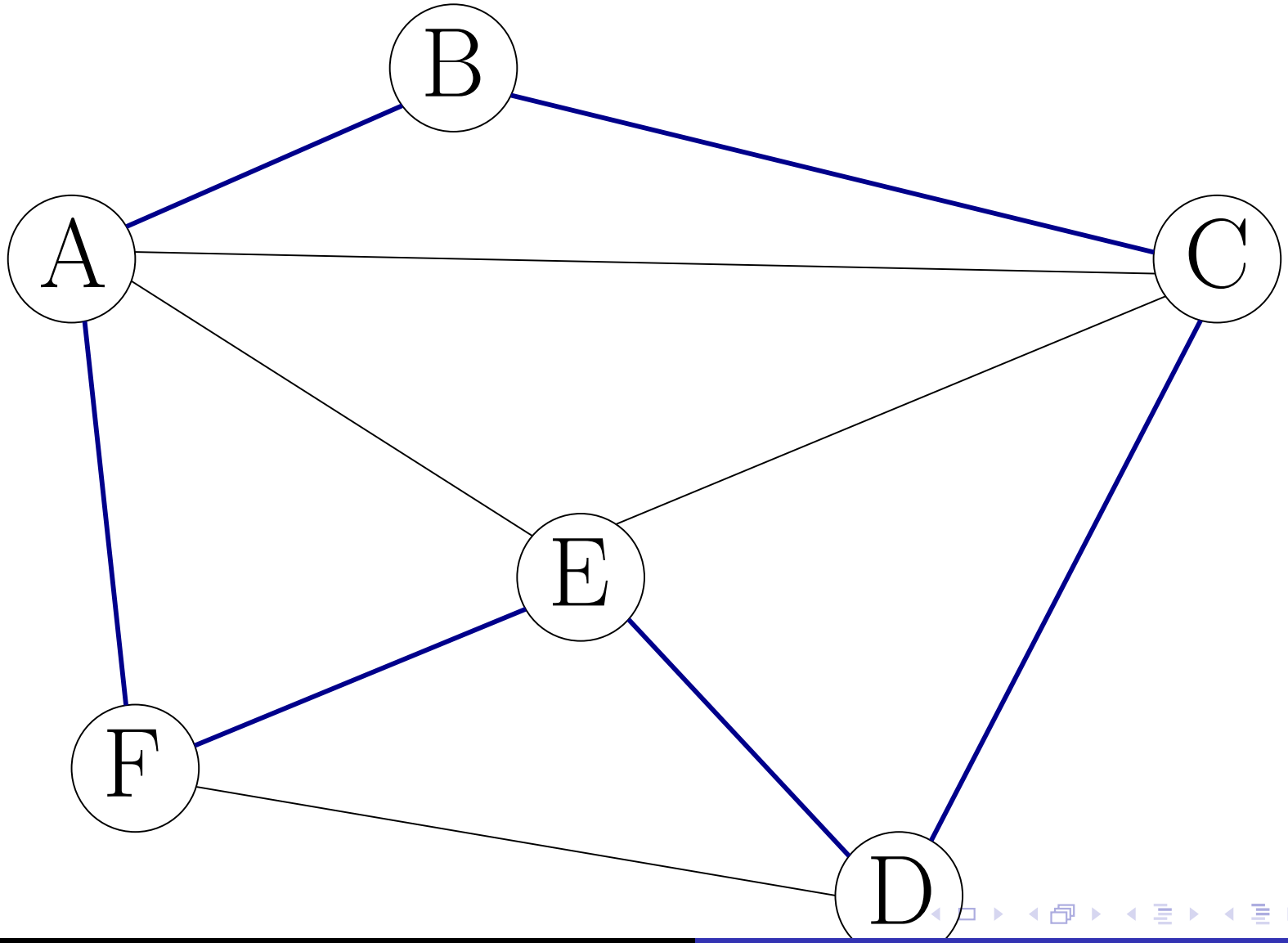
Eigenschaften der Dynamischen Programmierung

- Optimalitätsprinzip: Jeder Teilweg eines optimalen Weges ist selbst wiederum ein optimaler Weg
- Iterativer Durchgang durch alle Knoten
- Bereits gefundene Wege werden abgespeichert und bei weiteren Berechnungen wiederverwendet

Problem des Handlunsgreisenden

- Den kürzesten Pfad finden, ausgehend von einem Startknoten, der alle anderen Knoten genau (mindestens) einmal besucht, und dann wieder zum Startknoten zurückführt
- Anwendungen nicht nur in der Informatik
- Exakte als auch heuristische Lösungsverfahren

Beispielgraph



Komplexität

- $n!$ Permutationen
- Wahl des Startknotens irrelevant
- $(n-1)!$ für gerichtete Graphen, $\frac{(n-1)!}{2}$ für ungerichtete Graphen
- $\Rightarrow O(n!)$ bei brute-force Algorithmen!
- Schon bei kleiner Anzahl an Knoten unberechenbar $(15! = 1307674368000)$

Voraussetzungen

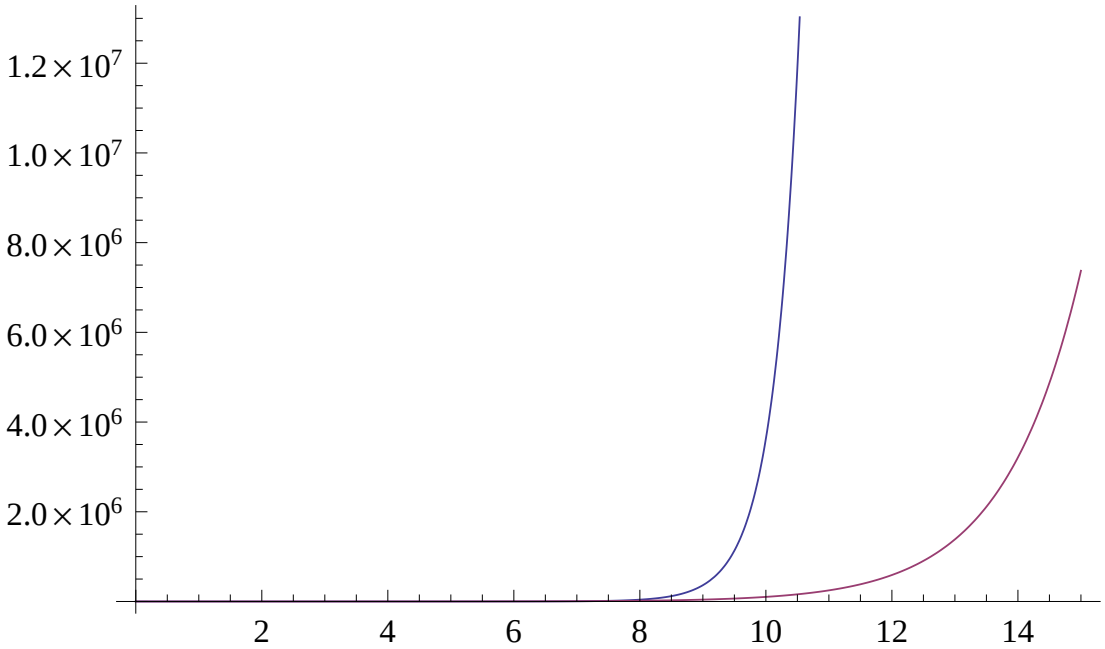
- Input
  - Menge V an n Knoten
  - Menge E an Kanten, wobei e_{ij} die Distanz zwischen den Knoten v_i und v_j beschreibt
- Die Funktion $t(v_i, K \subseteq V)$ berechnet die Länge des kürzesten Pfades zwischen dem Startknoten und v_i, wobei jeder Knoten aus K genau einmal besucht wird
  - $t(v_i, \emptyset) = e_{1i}$
  - $t(v_i, K) = \min_{\forall v_k \in K}(t(v_k, K - \{v_k\}) + e_{ki})$

Berechnung

- Optimaler Pfad: $\min_{j=2,3,\ldots,n}(t(v_j, V - \{v_j\}) + e_{j1})$
- Aufrufen der Funktion t für $|K| = 1, 2, 3, \ldots, n-2$
- Zwischenspeichern der bereits gefundenen Pfade in einer Tabelle
- Zugriff auf bereits berechnete Pfade anstatt rekursiven Aufrufens der Funktion

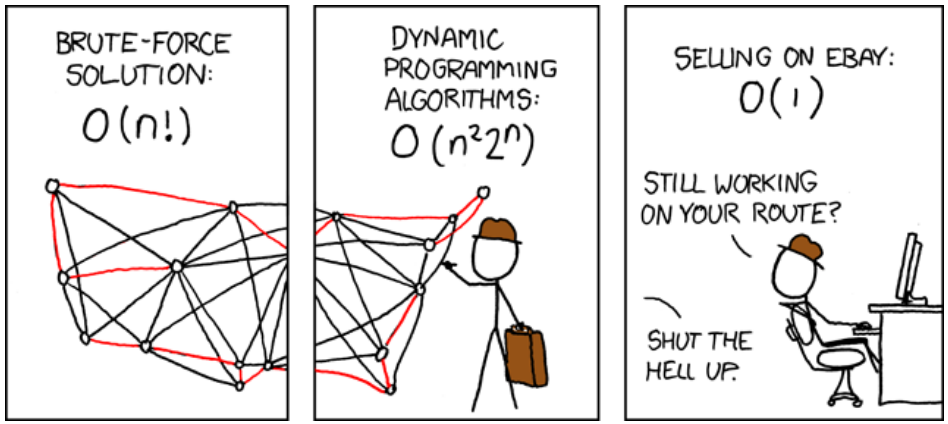
Resultate

- Exponentielle Speichernutzung
- Komplexität von $O(n^2 2^n)$ im Vergleich zu $O(n!)$



Ende

Vielen Dank für die Aufmerksamkeit!



xkcd: The travelling salesman problem

Quellenangabe

Stuard E. Dreyfuse, Averill M. Law

The Art and Theory of Dynamic Programming

Academic Press Inc., 1977

ISBN: 0122218604

M. Sniedovic

Dijkstra's algorithm revisited: the dynamic programming connexion

Control and Cybernetics, Vol. 35, No. 3: Dynamic Programming, 599 - 620

Jan W. Owsinki et. al

System research institute, Polish academy of sciences

2006

Quellenangabe con'd

xkcd: Traveling Salesman Problem
http://xkcd.com/399/