Informatik II

Übung 9

FS 2019

Heutiges Programm

1. Wiederholung Theorie
   - Editierdistanz
2. In-Class Exercise
   - Implement on CodeExpert

1. Wiederholung Theorie

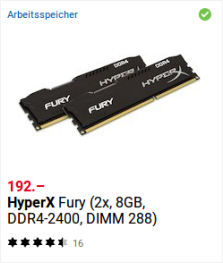
Dynamische Programmierung: Idee

- Aufteilen eines komplexen Problems in eine vernünftige Anzahl kleinerer Teilprobleme
- Die Lösung der Teilprobleme wird zur Lösung des komplexeren Problems verwendet
- Identische Teilprobleme werden nur einmal gerechnet

Dynamische Programmierung: Konsequenz

Identische Teilprobleme werden nur einmal gerechnet

$\Rightarrow$ Resultate werden zwischengespeichert



Wir tauschen Laufzeit gegen Speicherplatz

Dynamic Programming = Divide-And-Conquer ?

- In beiden Fällen ist das Ursprungsproblem (einfacher) lösbar, indem Lösungen von Teilproblemen herangezogen werden können. Das Problem hat *optimale Substruktur*.
- Bei Divide-And-Conquer Algorithmen (z.B. Mergesort) sind Teilprobleme unabhängig; deren Lösungen werden im Algorithmus nur einmal benötigt.
- Beim DP sind Teilprobleme nicht unabhängig. Das Problem hat *überlappende Teilprobleme*, welche im Algorithmus mehrfach gebraucht werden.
- Damit sie nur einmal gerechnet werden müssen, werden Resultate tabelliert. Dafür darf es *zwischen Teilproblemen keine zirkulären Abhängigkeiten* geben.

Minimale Editierdistanz

Editierdistanz von zwei Zeichenketten $A_n = (a_1, \ldots, a_n)$, $B_m = (b_1, \ldots, b_m)$.

Editieroperationen:

- Einfügen eines Zeichens
- Löschen eines Zeichens
- Änderung eines Zeichens

Frage: Wie viele Editieroperationen sind mindestens nötig, um eine gegebene Zeichenkette A in eine Zeichenkette B zu überführen.

TIGER ZIGER ZIEGER ZIEGE

Minimale Editierdistanz

Gesucht: Günstigste zeichenweise Transformation $A_n \to B_m$ mit Kosten

Operation	Levenshtein	LGT[1]	allgemein
c einfügen	1	1	ins(c)
c löschen	1	1	del(c)
Ersetzen $c \to c'$	$\mathbb{1}(c \neq c')$	$\infty \cdot \mathbb{1}(c \neq c')$	repl(c, c')

Beispiel

T	I	G	E	R		T	I	_	G	E	R		T→Z	+E	-R
Z	I	E	G	E		Z	I	E	G	E	_		Z→T	-E	+R

[1] Längste gemeinsame Teilfolge – Spezialfall des Editierproblems

Wie findet man den DP Algorithmus

0. Genaue Formulierung der gesuchten Lösung
1. Definiere Teilprobleme (und bestimme deren Anzahl)
2. Raten / Aufzählen (und bestimme die Laufzeit für das Raten)
3. Rekursion: verbinde die Teilprobleme
4. Memoisieren / Tabellieren. Bestimme die Abhängigkeiten der Teilprobleme
5. Lösung des Problems
   Laufzeit = #Teilprobleme $\times$ Zeit/Teilproblem

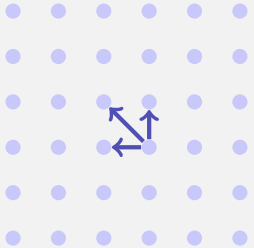
DP

0. $E(n, m)$ = minimale Anzahl Editieroperationen (ED Kosten) für $a_{1...n} \to b_{1...m}$
1. Teilprobleme $E(i, j)$ = ED von $a_{1...i}$. $b_{1...j}$. #TP $= n \cdot m$
2. Raten/Probieren Kosten$\Theta(1)$
   - $a_{1..i} \to a_{1...i-1}$ (löschen)
   - $a_{1..i} \to a_{1...i}b_j$ (einfügen)
   - $a_{1..i} \to a_{1...i_1}b_j$ (ersetzen)
3. Rekursion

$$E(i,j) = \min \begin{cases} \mathsf{del}(a_i) + E(i-1, j), \\ \mathsf{ins}(b_j) + E(i, j-1), \\ \mathsf{repl}(a_i, b_j) + E(i-1, j-1) \end{cases}$$

DP

4. Abhängigkeiten



$\Rightarrow$ Berechnung von links oben nach rechts unten. Zeilen- oder Spaltenweise.

5. Lösung steht in $E(n, m)$

Beispiel (Levenshteinabstand)

$$E[i,j] \leftarrow \min\left\{E[i-1,j]+1, E[i,j-1]+1, E[i-1,j-1]+\mathbb{1}(a_i \neq b_j)\right\}$$

	∅	Z	I	E	G	E
∅	0	1	2	3	4	5
T	1	1	2	3	4	5
I	2	2	1	2	3	4
G	3	3	2	2	2	3
E	4	4	3	2	3	2
R	5	5	4	3	3	3

Editierschritte: von rechts unten nach links oben, der Rekursion folgend.

Dynamische Programmierung

Eine vollständige Beschreibung eines dynamischen Programms behandelt **immer** die folgenden Aspekte :

Dynamische Programmierung

Eine vollständige Beschreibung eines dynamischen Programms behandelt **immer** die folgenden Aspekte :

- **Definition der DP-Tabelle**:

Dynamische Programmierung

Eine vollständige Beschreibung eines dynamischen Programms behandelt **immer** die folgenden Aspekte :

- **Definition der DP-Tabelle**: Welche Dimensionen hat die Tabelle? Was ist die Bedeutung jedes Eintrags?

Dynamische Programmierung

Eine vollständige Beschreibung eines dynamischen Programms behandelt **immer** die folgenden Aspekte :

- **Definition der DP-Tabelle**: Welche Dimensionen hat die Tabelle? Was ist die Bedeutung jedes Eintrags?
- **Berechnung eines Eintrags**:

Dynamische Programmierung

Eine vollständige Beschreibung eines dynamischen Programms behandelt **immer** die folgenden Aspekte :

- **Definition der DP-Tabelle**: Welche Dimensionen hat die Tabelle? Was ist die Bedeutung jedes Eintrags?
- **Berechnung eines Eintrags**: Wie berechnet sich ein Eintrag aus den Werten von anderen Einträgen? Welche Einträge hängen nicht von anderen Einträgen ab?

Dynamische Programmierung

Eine vollständige Beschreibung eines dynamischen Programms behandelt **immer** die folgenden Aspekte :

- **Definition der DP-Tabelle**: Welche Dimensionen hat die Tabelle? Was ist die Bedeutung jedes Eintrags?
- **Berechnung eines Eintrags**: Wie berechnet sich ein Eintrag aus den Werten von anderen Einträgen? Welche Einträge hängen nicht von anderen Einträgen ab?
- **Berechnungsreihenfolge**:

Dynamische Programmierung

Eine vollständige Beschreibung eines dynamischen Programms behandelt **immer** die folgenden Aspekte :

- **Definition der DP-Tabelle**: Welche Dimensionen hat die Tabelle? Was ist die Bedeutung jedes Eintrags?
- **Berechnung eines Eintrags**: Wie berechnet sich ein Eintrag aus den Werten von anderen Einträgen? Welche Einträge hängen nicht von anderen Einträgen ab?
- **Berechnungsreihenfolge**: In welcher Reihenfolge kann man die Einträge berechnen, so dass die jeweils benötigten anderen Einträge bereits vorher berechnet wurden?

Dynamische Programmierung

Eine vollständige Beschreibung eines dynamischen Programms behandelt **immer** die folgenden Aspekte :

- **Definition der DP-Tabelle**: Welche Dimensionen hat die Tabelle? Was ist die Bedeutung jedes Eintrags?
- **Berechnung eines Eintrags**: Wie berechnet sich ein Eintrag aus den Werten von anderen Einträgen? Welche Einträge hängen nicht von anderen Einträgen ab?
- **Berechnungsreihenfolge**: In welcher Reihenfolge kann man die Einträge berechnen, so dass die jeweils benötigten anderen Einträge bereits vorher berechnet wurden?
- **Auslesen der Lösung**:

Dynamische Programmierung

Eine vollständige Beschreibung eines dynamischen Programms behandelt **immer** die folgenden Aspekte :

- **Definition der DP-Tabelle**: Welche Dimensionen hat die Tabelle? Was ist die Bedeutung jedes Eintrags?
- **Berechnung eines Eintrags**: Wie berechnet sich ein Eintrag aus den Werten von anderen Einträgen? Welche Einträge hängen nicht von anderen Einträgen ab?
- **Berechnungsreihenfolge**: In welcher Reihenfolge kann man die Einträge berechnen, so dass die jeweils benötigten anderen Einträge bereits vorher berechnet wurden?
- **Auslesen der Lösung**: Wie lässt sich die Lösung am Ende aus der Tabelle auslesen?

Dynamische Programmierung

Eine vollständige Beschreibung eines dynamischen Programms behandelt **immer** die folgenden Aspekte :

- **Definition der DP-Tabelle**: Welche Dimensionen hat die Tabelle? Was ist die Bedeutung jedes Eintrags?
- **Berechnung eines Eintrags**: Wie berechnet sich ein Eintrag aus den Werten von anderen Einträgen? Welche Einträge hängen nicht von anderen Einträgen ab?
- **Berechnungsreihenfolge**: In welcher Reihenfolge kann man die Einträge berechnen, so dass die jeweils benötigten anderen Einträge bereits vorher berechnet wurden?
- **Auslesen der Lösung**: Wie lässt sich die Lösung am Ende aus der Tabelle auslesen?

Bottom-Up DP Algorithmus ED

1. **Dimension der Tabelle? Bedeutung der Einträge?**

Bottom-Up DP Algorithmus ED

Dimension der Tabelle? Bedeutung der Einträge?

1. Tabelle $E[0, \ldots, n][0, \ldots, m]$. $E[i, j]$: Minimaler Editierabstand der Zeichenketten $(a_1, \ldots, a_i)$ und $(b_1, \ldots, b_j)$

Bottom-Up DP Algorithmus ED

Dimension der Tabelle? Bedeutung der Einträge?

1. Tabelle $E[0, \ldots, n][0, \ldots, m]$. $E[i, j]$: Minimaler Editierabstand der Zeichenketten $(a_1, \ldots, a_i)$ und $(b_1, \ldots, b_j)$

Berechnung eines Eintrags

2.

Bottom-Up DP Algorithmus ED

Dimension der Tabelle? Bedeutung der Einträge?

1. Tabelle $E[0,\ldots,n][0,\ldots,m]$. $E[i,j]$: Minimaler Editierabstand der Zeichenketten $(a_1,\ldots,a_i)$ und $(b_1,\ldots,b_j)$

Berechnung eines Eintrags

2. $E[0,i] \leftarrow i \; \forall 0 \leq i \leq m$, $E[j,0] \leftarrow i \; \forall 0 \leq j \leq n$.
sonst mit $E[i,j] =$
$\min\{\mathsf{del}(a_i) + E(i-1,j), \mathsf{ins}(b_j) + E(i,j-1), \mathsf{repl}(a_i,b_j) + E(i-1,j-1)\}$

Bottom-Up DP Algorithmus ED

Berechnungsreihenfolge

3

Bottom-Up DP Algorithmus ED

Berechnungsreihenfolge

3 Abhängigkeiten berücksichtigen: z.B. Zeilen aufsteigend und innerhalb von Zeilen Spalten aufsteigend.

Bottom-Up DP Algorithmus ED

3

Berechnungsreihenfolge

Abhängigkeiten berücksichtigen: z.B. Zeilen aufsteigend und innerhalb von Zeilen Spalten aufsteigend.

4

Wie kann sich Lösung aus der Tabelle konstruieren lassen?

Bottom-Up DP Algorithmus ED

Berechnungsreihenfolge

3 Abhängigkeiten berücksichtigen: z.B. Zeilen aufsteigend und innerhalb von Zeilen Spalten aufsteigend.

Wie kann sich Lösung aus der Tabelle konstruieren lassen?

4 Beginne bei $j = m$, $i = n$. Falls $E[i,j] = \text{repl}(a_i, b_j) + E(i-1, j-1)$ gilt, gib $a_i \rightarrow b_j$ aus und fahre fort mit $(j,i) \leftarrow (j-1, i-1)$; sonst, falls $E[i,j] = \text{del}(a_i) + E(i-1,j)$ gib $\text{del}(a_i)$ aus fahre fort mit $j \leftarrow j-1$; sonst, falls $E[i,j] = \text{ins}(b_j) + E(i,j-1)$, gib $\text{ins}(b_j)$ aus und fahre fort mit $i \leftarrow i-1$. Terminiere für $i = 0$ und $j = 0$.

Längste aufsteigende Sequenz in Matrix

Gegeben $n \times m$ Matrix A:

9	27	42	41	48
35	39	8	3	5
12	49	2	38	4
15	47	29	28	6
19	1	25	33	10

Längste aufsteigende Sequenz in Matrix

Gegeben $n \times m$ Matrix A:

9	27	42	41	48
35	39	8	3	5
12	49	2	38	4
15	47	29	28	6
19	1	25	33	10

Gesucht längste aufsteigende Sequenz:

$$4, 6, 28, 29, 47, 49$$

Definition der DP-Tabelle

- Welche Dimensionen hat die Tabelle?

Definition der DP-Tabelle

- Welche Dimensionen hat die Tabelle?
  - $n \times m$

Definition der DP-Tabelle

- Welche Dimensionen hat die Tabelle?
  - $n \times m (\times 2)$

Definition der DP-Tabelle

- Welche Dimensionen hat die Tabelle?
  - $n \times m (\times 2)$
- Was ist die Bedeutung jedes Eintrags?

Definition der DP-Tabelle

- Welche Dimensionen hat die Tabelle?
  - $n \times m (\times 2)$
- Was ist die Bedeutung jedes Eintrags?
  - In $T[x][y]$ steht Länge der längsten aufsteigenden Sequenz, die im Feld $A[x][y]$ endet
  - In $S[x][y]$ steht Koordinaten des Vorgängers von (x, y) in aufsteigender Sequenz (falls existent)

Berechnung eines Eintrags

- Wie berechnet sich ein Eintrag aus den Werten von anderen Einträgen? Welche Einträge hängen nicht von anderen Einträgen ab?

Berechnung eines Eintrags

- Wie berechnet sich ein Eintrag aus den Werten von anderen Einträgen? Welche Einträge hängen nicht von anderen Einträgen ab?
  - Betrachte Nachbarn mit kleineren Eintrag in A.

Berechnung eines Eintrags

- Wie berechnet sich ein Eintrag aus den Werten von anderen Einträgen? Welche Einträge hängen nicht von anderen Einträgen ab?
  - Betrachte Nachbarn mit kleineren Eintrag in A.
  - Wähle von den kleineren Einträgen den mit dem grössten Eintrag in T

Berechnung eines Eintrags

- Wie berechnet sich ein Eintrag aus den Werten von anderen Einträgen? Welche Einträge hängen nicht von anderen Einträgen ab?
  - Betrachte Nachbarn mit kleineren Eintrag in A.
  - Wähle von den kleineren Einträgen den mit dem grössten Eintrag in T
  - Aktualisiere T und S. (S erhält Koordinaten vom ausgewählten Nachbar, T erhält Wert um eins erhöht vom ausgewählten Nachbar.)

Berechnung eines Eintrags

- Wie berechnet sich ein Eintrag aus den Werten von anderen Einträgen? Welche Einträge hängen nicht von anderen Einträgen ab?
  - Betrachte Nachbarn mit kleineren Eintrag in A.
  - Wähle von den kleineren Einträgen den mit dem grössten Eintrag in T
  - Aktualisiere T und S. (S erhält Koordinaten vom ausgewählten Nachbar, T erhält Wert um eins erhöht vom ausgewählten Nachbar.)

Berechnungsreihenfolge

- In welcher Reihenfolge kann man die Einträge berechnen, so dass die jeweils benötigten anderen Einträge bereits vorher berechnet wurden?

Berechnungsreihenfolge

- In welcher Reihenfolge kann man die Einträge berechnen, so dass die jeweils benötigten anderen Einträge bereits vorher berechnet wurden?

- Bottom-Up: Beginne mit kleinstem Element in A und so weiter. (Bedeutet dass man A sortieren muss.)

Berechnungsreihenfolge

- In welcher Reihenfolge kann man die Einträge berechnen, so dass die jeweils benötigten anderen Einträge bereits vorher berechnet wurden?

- Bottom-Up: Beginne mit kleinstem Element in A und so weiter. (Bedeutet dass man A sortieren muss.)

- Rekursiv: Beliebige Reihenfolge, falls Eintrag schon berechnet überspringen sonst rekursiv von kleinern Nachbarn berechnen.

Auslesen der Lösung

- Wie lässt sich die Lösung am Ende aus der Tabelle auslesen?

Auslesen der Lösung

- Wie lässt sich die Lösung am Ende aus der Tabelle auslesen?
  - Betrachte alle Einträge um den Eintrag zu finden, in dem eine längste Sequenz endet. Von dort aus können wir die Lösung rekonstruieren, indem wir dem entsprechenden Vorgänger folgen.

Program

Implement a DP solution in the prepared CodeExpert program.

Fragen oder Anregungen?