Dynamické programování

Postup návrhu algoritmu dynamickým programováním:

1. Rozmyslet rozdělení na (překrývající se) podproblémy.
2. Sestavit rekurzivní algoritmus.
   - Typicky hledáme optimální hodnotu, pro niž sestavíme rekurzivní vztah.
   - V tomto kroku se nestaráme o složitost!
3. Určit správné pořadí počítání podproblémů, které zajistí, že se každý spočítá právě jednou. (*bottom-up* přístup)
   - Alternativou je *top-down with memoization*, ale většinou preferujeme *bottom-up*. Proč?
4. Chceme-li kromě optimální hodnoty znát i její realizaci: Navrhnout způsob, jak řešení sestavit.
   - Použijeme napočítané hodnoty podproblémů, někdy je třeba mírná modifikace.

Dynamické programování

Platba mincemi

Mějme n druhů mincí hodnot $v_1, v_2, \ldots, v_n$ a částku C. Chceme zaplatit částku C co nejmenším počtem mincí. Předpokládáme, že mincí každého druhu máme neomezený počet.

- Jaký postup funguje pro sadu mincí 1, 2, 5, 10, 20, 50?

Dynamické programování

Platba mincemi

Mějme n druhů mincí hodnot $v_1, v_2, \ldots, v_n$ a částku C. Chceme zaplatit částku C co nejmenším počtem mincí. Předpokládáme, že mincí každého druhu máme neomezený počet.

- Jaký postup funguje pro sadu mincí 1, 2, 5, 10, 20, 50?
- Funguje stejný postup pro libovolnou sadu mincí?

Dynamické programování

Platba mincemi

Mějme n druhů mincí hodnot $v_1, v_2, \ldots, v_n$ a částku C. Chceme zaplatit částku C co nejmenším počtem mincí. Předpokládáme, že mincí každého druhu máme neomezený počet.

- Jaký postup funguje pro sadu mincí 1, 2, 5, 10, 20, 50?
- Funguje stejný postup pro libovolnou sadu mincí?
- Chceme použít dynamické programování:
  1. podproblémy
  2. rekurzivní vztah pro optimální hodnotu
  3. pořadí počítání podproblémů
  4. sestavení řešení

Dynamické programování

Souvislá podposloupnost s největším součtem
Máme na vstupu posloupnost celých čísel. Chceme najít souvislou podposloupnost této posloupnosti, která má co největší součet.

Příklad: Pro vstup 3, -5, 7, 0, -2, 6, 8, -9, 3 je řešením souvislá podposloupnost 7, 0, -2, 6, 8 (součet 19).

- Chceme použít dynamické programování:
  1. podproblémy
  2. rekurzivní vztah pro optimální hodnotu
  3. pořadí počítání podproblémů
  4. sestavení řešení

Dynamické programování

Hra na mřížce
Mějme mřížku velikosti $n \times m$. V každém poli mřížky je napsáno jedno celé číslo. Hráč položí na začátku kámen do horního levého rohu. V každém kole pak posune kamenem buď doprava nebo dolů. Hra končí, jestliže kámen opustí mřížku. Hodnotou hry je součet hodnot všech polí, která hráčův kámen navštívil. Chceme najít optimální strategii pro danou mřížku.

- Chceme použít dynamické programování:
  1. podproblémy
  2. rekurzivní vztah pro optimální hodnotu
  3. pořadí počítání podproblémů
  4. sestavení řešení

Dynamické programování

Zarovnávání kusů DNA

Mějme dva úseky DNA (řetězce se čtyřmi druhy znaků: A, C, T, G). Chceme do těchto úseků vkládat mezery, tak aby bylo následující bodování co největší:

- -2 body, pokud je v jednom z řetězců na dané pozici mezera,
- -1 bod, pokud se znaky na stejných pozicích nerovnají,
- $+1$ bod, pokud se znaky na stejných pozicích rovnají.

Příklad: úseky `CATGA` a `CTGCA` mají nejlepší zarovnání:

```
CATG␣A
C␣TGCA
```

Hodnocení: $1 - 2 + 1 + 1 - 2 + 1 = 0$ bodů.

- Chceme použít dynamické programování: 1, 2, 3, 4.