Dynamisk programmering

Dynamisk programmering

Et algoritme-konstruktionsprincip ("paradigme") for optimeringsproblemer.

Har en hvis lighed med divide-and-conquer: Begge opbygger løsninger til større problemer fra løsninger til mindre problemer.

Forskel:

- Divide-and-conquer: delproblemer typisk halvt så store, ingen gentagelser af delproblemer (heller ikke flere lag nede i rekursionen).
- Dynamisk programmering: delproblemer indeholder nogle hvis størrelse kun er reduceret med én, der er gentagelser blandt delproblemers delproblemer.

Eksempel

Du har en guldkæde med n led. Den kan deles i mindre længder (med n led tilsammen, dvs. ingen led går tabt). Guldsmeden køber guldkæder af forskellige længder til forskellige priser:

længde i	1	2	3	4	5	6	7	8	9	$\cdots$
pris p_i	\$1	\$5	\$8	\$9	\$10	\$17	\$17	\$20	\$24	$\cdots$

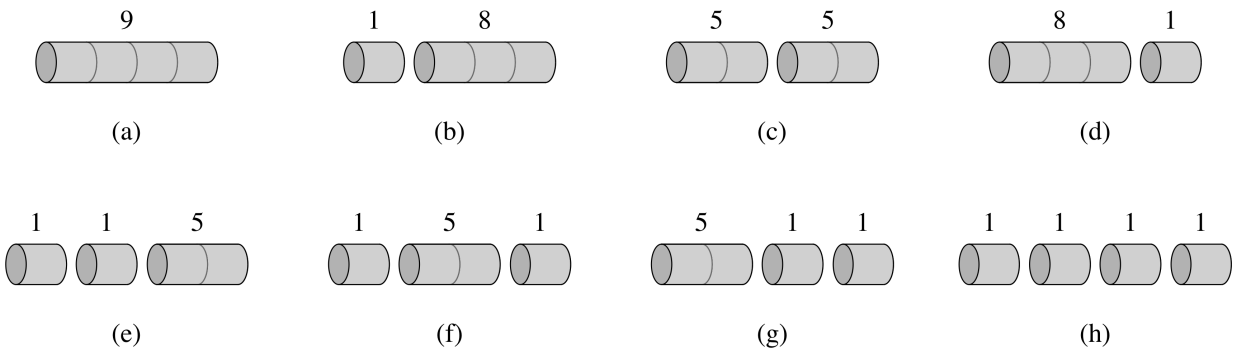
Hvordan skal du opdele din guldkæde for at optimere din salgspris?

Eksempel

Du har en guldkæde med n led. Den kan deles i mindre længder (med n led tilsammen, dvs. ingen led går tabt). Guldsmeden køber guldkæder af forskellige længder til forskellige priser:

længde i	1	2	3	4	5	6	7	8	9	$\cdots$
pris p_i	\$1	\$5	\$8	\$9	\$10	\$17	\$17	\$20	\$24	$\cdots$

Hvordan skal du opdele din guldkæde for at optimere din salgspris?



Optimale delprobleme

Observation: en opdeling af en kæde af længde n må bestå af:

- Et sidste stykke af længde $i \leq n$.
- En opdeling af resten, dvs. en opdeling af en kæde af længde $n - i$.

Den essentielle egenskab (optimale delprobleme):

> For en *optimal* opdelingen af kæden af længde n, må opdelingen af resten være optimal for en kæde af længde $n - i$. For hvis der fandtes en ægte bedre opdeling af den, kunne man bruge den til at forbedre opdelingen af kæden af længde n.

Kald *værdien* af en optimal opdeling af en kæde af længde n for $r(n)$.

Det er klart at $r(0) = 0$. Vi vil gerne finde $r(n)$ for $n > 0$.

Rekursiv formel for $r(n)$

En opdeling af en kæde af længde n består af:

- Et sidste stykke af længde $i \leq n$.
- En opdeling af resten, dvs. en opdeling af en kæde af længde $n - i$.

Lad T_i være opdelingen som består af et sidste stykke af længde $i \leq n$, samt en optimal opdeling af resten.

Pga. den essentielle egenskab: Én af T_1, T_2, $T_3, \ldots, T_n$ er en optimale opdeling for længde n.

Værdien af T_i er $p_i + r(n - i)$, og ingen af disse værdier kan være større end værdien af den optimale.

Heraf:

$$r(n) = \max_{1 \leq i \leq n} (p_i + r(n - i)), \qquad r(0) = 0$$

Beregne de optimale værdier

$$r(n) = \max_{1 \leq i \leq n} (p_i + r(n-i)), \qquad r(0) = 0$$

Dvs. $r(n)$ er (matematisk set) rekursivt defineret ud fra mindre instanser.

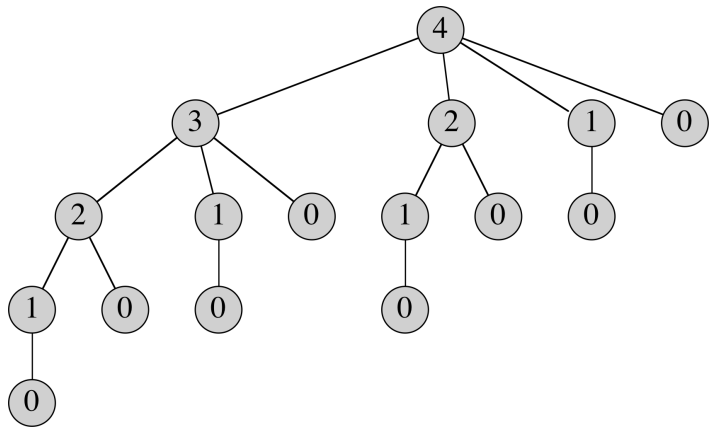
Er rekursion også en god løsning, algoritmisk set?

Beregne de optimale værdier

$$r(n) = \max_{1 \leq i \leq n} (p_i + r(n-i)), \qquad r(0) = 0$$

Dvs. $r(n)$ er (matematisk set) rekursivt defineret ud fra mindre instanser.

Er rekursion også en god løsning, algoritmisk set?



Man kan vise via induktion at der er 2^n knuder i rekursionstræet. Så køretiden vil blive $\Theta(2^n)$

Problemet er gentagelser blandt delproblemers delproblemer.

Beregne de optimale værdier

Problemet for en rekursiv algoritme er gentagelser blandt delproblemers delproblemer, dvs. gentagne beregninger af de samme løsninger.

Fokusér i stedet på en tabel over værdien af de optimale løsninger.

Start:

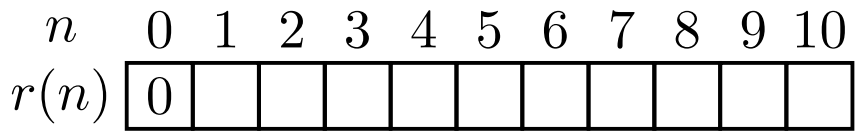
n	0	1	2	3	4	5	6	7	8	9	10
$r(n)$	0										

Beregne de optimale værdier

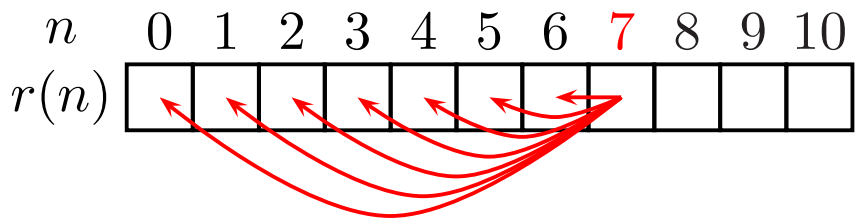
Problemet for en rekursiv algoritme er gentagelser blandt delproblemers delproblemer, dvs. gentagne beregninger af de samme løsninger.

Fokusér i stedet på en tabel over værdien af de optimale løsninger.

Start:



Beregn $r(n)$ for stigende n:



$$r(n) = \max_{1 \le i \le n} (p_i + r(n-i)), \qquad r(0) = 0$$

Tid: $O(1 + 2 + 3 + 4 + \cdots + n) = O(n^2)$

Eksempel

Brug

$$r(n) = \max_{1 \le i \le n} (p_i + r(n-i)), \qquad r(0) = 0$$

og tabellen over p_i:

længde i	1	2	3	4	5	6	7	8	9	10
pris p_i	1	5	8	9	10	17	17	20	24	30

til at udfylde tabellen over $r(n)$ fra højre mod venstre:

længde n	0	1	2	3	4	5	6	7	8	9	10
optimal værdi $r(n)$	0	1	5	8	10	13	17	18	22	...	

Find selve løsningen

Tallet $r(n)$ er kun *værdien* af den optimale løsning. Hvad hvis vi gerne vil have *selve løsningen* (de enkelte længder, guldkæden skal brydes op i)?

Gem længden $s(n)$ af det *sidste stykke* for en optimal løsning for længde n.

$$r(n) = \max_{1 \le i \le n} (p_i + r(n-i)), \qquad r(0) = 0$$

længde i	1	2	3	4	5	6	7	8	9	10
pris p_i	1	5	8	9	10	17	17	20	24	30

længde n	0	1	2	3	4	5	6	7	8	9	10
optimal værdi $r(n)$	0	1	5	8	10	13	17	18	22	25	30
sidste længde $s(n)$	0	1	2	3	2	2	6	1	2	3	10

while $n > 0$
  print $s[n]$
  $n = n - s[n]$

Memoization

Rekursion: $O(2^n)$. Struktureret tabeludfyldning: $O(n^2)$

Kan de to kombineres?

Memoization

Rekursion: $O(2^n)$. Struktureret tabeludfyldning: $O(n^2)$

Kan de to kombineres?Ja.

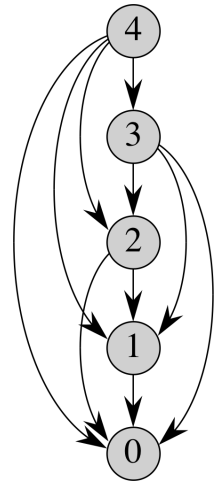
GuldKæde(n)
  if $n = 0$
    return 0
  else if $r(n)$ allerede udfyldt i tabel
    return $r(n)$
  else
   $x = \max_{1 \leq i \leq n}(p_i + \text{GuldKæde}(n - i))$
    $r(n) = x$
    return x

Memoization

Rekursion: $O(2^n)$. Struktureret tabeludfyldning: $O(n^2)$

Kan de to kombineres? Ja.

GuldKæde(n)
  if $n = 0$
    return 0
  else if $r(n)$ allerede udfyldt i tabel
    return $r(n)$
  else
   $x = \max_{1 \le i \le n}(p_i + \text{GuldKæde}(n - i))$
    $r(n) = x$
    return x



En kant i grafen, der viser delproblemers afhængighed af hinanden, vil blive en kant i rekursionstræet præcis én gang.

Så samme køretid $O(n^2)$ og pladsforbrug $O(n)$ som for bottom-up udfyldning af tabellen. Men nok dårligere konstanter i praksis.