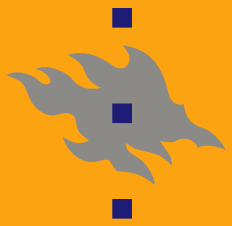
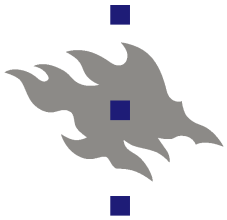


HELSINGIN YLIOPISTO
HELSINGFORS UNIVERSITET
UNIVERSITY OF HELSINKI

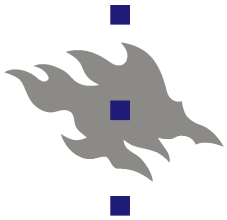
Dynaaminen ohjelmointi, osa 2

Algoritmit ongelmanratkaisussa, kevät 2018



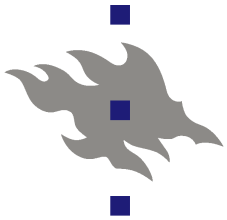
Repunpakkaus

- meillä on n tavaraa, joilla on tietyt painot
- minkä painoisia osajoukkoja voimme muodostaa tavaroista?
- esimerkki: painot ovat $[1, 3, 8]$
- vastaus: 0, 1, 3, 4, 8, 9, 11, 12



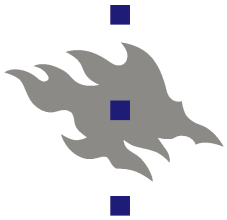
Repunpakkaus

- ongelma tunnetaan englanniksi nimellä ”knapsack”
- muunnelmia:
  - monenko osajoukon painona on x?
  - tavaroilla on arvot – mikä on arvokkain osajoukko, jonka paino on enintään x?
- dynaaminen ohjelmointi soveltuu näiden ongelmien ratkaisemiseen



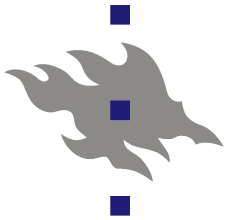
Rekursiivinen muotoilu

- n tavaraa, painot $p_1, p_2, \ldots, p_n$
- $f(a, b)$ tarkoittaa, voimmeko valita a ensimmäisestä tavarasta osajoukon, jonka painojen summa on b
- seuraus: $f(n, x)$ tarkoittaa, voimmeko valita kaikista tavaroista osajoukon, jonka painojen summa on x



Rekursiivinen muotoilu

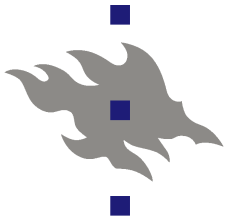
- $f(0,0) = \texttt{true}$, koska tyhjä joukko kelpaa
- $f(0,x) = \texttt{false}$, jos $x \neq 0$
- entä kuinka laskemme yleisen arvon $f(a,b)$?
- mahdollisuuksia on kaksi:
  - otamme mukaan painon p_a
  - emme ota mukaan painoa p_a
- $f(a,b) = f(a-1, b-p_a) \texttt{ or } f(a-1,b)$



Toteutus

- voimme muuntaa rekursiivisen funktion suoraan dynaamiseksi ohjelmoinniksi
- kuitenkin on kätevämpi toteutus:

```
f[0] = true;
for (int i = 1; i <= n; i++) {
   for (int j = M-p[i]; j >= 0; j--) {
      if (f[j]) f[j+p[i]] = true;
   }
}
```



Toteutus

- tämä koodi muodostaa taulukon, jossa $f[x] =$ `true` tarkalleen silloin, kun voimme muodostaa x-painoisen osajoukon
- huomaa, miten koodi päivittää taulukkoa lopusta alkuun, jotta toiminta on oikea
- samanlainen toteutus toimii muissakin repunpakkauksen muunnelmissa
- M on yläraja osajoukon painolle (eli käytännössä kaikkien tavaroiden painojen summa)