Dynamické programování

- Řešení vyjádříme kombinací řešení pro “podúlohy”
  - Části vstupu.
  - Menší parametry.
- Úlohy řešíme od nejmenší k největší a zaznamenáváme si výsledky.

Variace (rekurze s cachováním):

- Řešíme rekurzivně.
- Zaznamenáváme si výsledky, abychom se vyhnuli opakovaným voláním.

Příklad: Určete počet cest v acyklickém orientovaném grafu G z u do z.

Dynamické programování:

```
Topologicky setřid' vrcholy G.
Smaž ty před u či za z.
Zbylé vrcholy: v_1, ..., v_n, u=v_1, z=v_n.

cest[1] = 1;
for i = 2, ..., n:
  cest[i] = 0;
  for v_x -> v_i:
    cest[i] += cest[x];

return cest[n];
```

Příklad: Určete počet cest v acyklickém orientovaném grafu G z u do z.

Rekurze s cachováním:

```
for every v: cest[v] = -1;

počet_cest (do):
  if do == u: return 1;
  if cest[do] >= 0: return cest[do];
  cest[do] = 0;
  for x -> do:
    cest[do] += počet_cest (x);
  return cest[do];

return počet_cest (v);
```

Dynamické programování:

- má lokálnější přístupy do paměti
- nemá náklady na opakované (rekurzivní) volání funkce
- občas lze ušetřit paměť (pamatovat si jen relevantní část výsledků)

Rekurze s cachováním:

- nemusí nutně řešit všechny podúlohy
- není potřeba přemýšlet nad pořadím vyhodnocování

Jak poznat úlohu na dynamické programování:

- Velikost vstupů—často složitost $O(nm)$.
- Přirozené uspořádání.

Zadání

Máme mince hodnot $c_1, \ldots, c_m$. Kolik nejméně z těchto mincí potřebujeme k zaplacení hodnoty h?

$m \cdot h \approx 10^7$

- $p(a, b) =$ nejmenší počet z mincí s hodnotami $c_1, \ldots, c_a$, kterým lze zaplatit b.
- Výsledek $= p(m, h)$.

$$p(a,b) = \begin{cases} 0 & \text{když } a = b = 0 \\ \infty & \text{když } a = 0 \text{ a } b > 0 \\ p(a-1,b) & \text{když } c_a > b \\ \min \begin{pmatrix} p(a-1,b), \\ p(a-1,b-c_a)+1 \end{pmatrix} & \text{jinak.} \end{cases}$$

Zadání

Máme mince hodnot $c_1, \ldots, c_m$. Kolik nejméně z těchto mincí potřebujeme k zaplacení hodnoty h?

$m \cdot h \approx 10^7$

```
prev[0] = 0; for i=1,...,h: prev[i] = infty;
for a = 1, ..., m:
   for b = 0, ..., c[a]-1: act[b] = prev[b];
   for b = c[a], ..., h:
      act[b] = min (prev[b], prev[b-c[a]] + 1);
   prev = act;
return prev[h];
```

Čas $O(mh)$, paměť $O(h)$.

```
prev[0] = 0; for i=1,...,h: prev[i] = infty;
for a = 1, ..., m:
   for b = 0, ..., c[a]-1:
     act[b] = prev[b]; take[a][b] = false;
   for b = c[a], ..., h:
      if prev[b] <= prev[b-c[a]] + 1:
        act[b] = prev[b]; take[a][b] = false;
      else
        act[b] = prev[b-c[a]] + 1;
        take[a][b] = true;
   prev = act;

for a = m, ..., 1:
   if take[a][h]: print a; h -= c[a];
```

Zadání

Určete barevnost zadaného grafu s $n \approx 15$ vrcholy.

- Určíme $\chi(G[X])$ pro každé $X \subseteq V(G)$.
- $\chi(G[\emptyset]) = 0$
- Pro $X \neq \emptyset$:

$$\chi(G[X]) = 1 + \min\{\chi(G[X \setminus Y]) : Y \subseteq X \text{ nezávislá}, Y \neq \emptyset\}$$

Zadání

Určete barevnost zadaného grafu s $n \approx 15$ vrcholy.

```
chi[empty] = 0;
for X neprázdná podmn. V(G), v pořadí dle |X|:
  chi[X] = infinity;
  for Y neprázdná podmnožina X:
    if Y je nezávislá množina v G
      chi[X] = min (chi[X], 1 + chi[X-Y]);
return chi[V(G)];
```

Čas:

$$n^2 \sum_{X \subseteq V(G)} 2^{|X|} = n^2 \sum_{k=0}^{n} \binom{n}{k} 2^k = n^2 3^n$$