Posloupnosti - komentář a řešení

Rekurze v řešení této úlohy probíhá velmi elementárně. Předpokládáme, že již máme vygenerovánu nějakou začáteční část posloupnosti až před aktuální pozici, na aktuální pozici chceme vložit daný prvek a dále chceme posloupnost prodloužit v dalších úrovních rekurze.

Rekurzivní funkce obdrží na vstupu daný prvek a aktuální pozici a zkontroluje dvě věci. Zaprvé, pokud je daný prvek na aktuální pozici nepřípustný, funkce nic nedělá a ihned se vrací o úroveň výše. Zadruhé, pokud je prvek přípustný a přitom aktuální pozice je poslední v posloupnosti, pak je sestavená posloupnost kompletní, funkce tedy zvýší o jedničku počet úspěšně sestavených posloupností a opět se ihned vrací o úroveň výše.

Pokud není splněna ani jedna předchozí podmínka, zavolá funkce sama sebe s tím, že posloupnost byla právě prodloužena o jeden prvek a program se pokouší o další prodloužení dosud vygenerované částečné posloupnosti.

Při pokusu o prodloužení je nutno respektovat omezující podmínku úlohy, to jest další prvek se liší od aktuálně přidaného o některou z hodnot množiny M. Žádné další podmínky kontrolovat není třeba, o to se právě postará začátek příštího volání rekurze. Poté, co jsou všechny možné pokusy o prodloužení posloupnosti za aktuální pozicí vyčerpány, nastává návrat na předchozí úroveň rekurze.

Zjednodušujícím zjištěním je, že není nutno jakkoli fyzicky zaznamenávat generovanou posloupnost, ta je vlastně dána pořadím (na sytémovém zásobníku) zanořených rekurzivních volání s jejich parametry. Pouze je nutno si pamatovat, které prvky množiny {1, 2, ..., N} byly již v částečně vygenerované posloupnosti použity, aby si to rekurzivní funkce nemusela zdlouhavě ověřovat sama. Před zahájením pokusů o prodloužení posloupnosti jen zaregistrujeme uvnitř rekurzivní funkce, že daný prvek na aktuální pozici již je použitý, a tudíž nemůže být použit již dále v generované posloupnosti, to jest ani v dalších zanořených rekurzivních voláních. Po vyzkoušení všech pokusů o prodloužení a těsně před návratem na předchozí rekurzivní úroveň, zaregistrujeme, že daný prvek již je uvolněn a je k dispozici pro další pokusy na předchozích rekurzivních úrovních. K tomu postačí jediné logické pole indexované prvky množiny {1, 2, ..., N}.

Po tomto úvodu by již měl být kód dostatečně sám vypovídající.

```
static int N;
static boolean [] used;         // used values from set {1,2, ..., N}
static int [] M;                // set M defines possible extensions of a sequence
static int noOfSolutions = 0;

static void tryPut(int candidate, int position) {
  if((candidate < 1) || (candidate > N) || used[candidate])
    return;                                 // reject any non valid candidate

  if (position == N-1)                      // valid sequence is complete
    { noOfSolutions++; return; }
                                            // otherwise do:
  used[candidate] = true;                   // add candidate to the sequence
  for(int iM = 0; iM < M.length; iM++) {    // and try to expand the sequence
    tryPut(candidate-M[iM], position+1);
    tryPut(candidate+M[iM], position+1);
  }
  used[candidate] = false;                  // finally remove the candidate
}                                           // and return

public static void main(String[] args)  throws IOException {
  readAndInitAll ();
  // do the job:
  noOfSolutions = 0;
  for(int candidate = 1; candidate <= N; candidate++)
    tryPut(candidate, 0);                  // all valid candidates at position 0
  System.out.println(noOfSolutions);       // the result
} // and that's all.
```

```
static void readAndInitAll () throws IOException {
  // nearly longer than the solution itself :-)
  BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
  StringTokenizer st = new StringTokenizer(br.readLine());
  N = Integer.valueOf(st.nextToken());
  used = new boolean [N+1];
  st = new StringTokenizer(br.readLine());
  int Msize  = Integer.valueOf(st.nextToken());
  M = new int [Msize];
  for(int i = 0; i < Msize; i++)
  M[i] = Integer.valueOf(st.nextToken());
}
```

Poznámka

Kód je úmyslně prezentován v co nekompaktnější formě. Pro ladění lze doporučit např. ukládat generovanou posloupnost do dalšího globálního pole a ve vhodném okamžiku ji tisknout, apod.; tato pomocná/usnadňující zařízení zde neuvádíme kvůli zdůraznění myšlenky kódu. Stejně tak je možné přenést kontroly přípustnosti před rekurzivní volání, čas řešení se nepatrně zkrátí, kód neptrně prodlouží atd.