Praktisk kombinatorik

Niels Möller

2025-11-06

Kort CV

- Natur, Brännkyrka gymnasium
- Y-linjen, Linköping
- Roxen (Web-servrar), EHAND (handdatorer)
- Doktorsstudier reglerteknik
- Conemtech, Southpole (linux), Google (WebRTC, Meet)
- Glasklarteknik AB

Basic, 6502 assembly, Pascal, C, Common LISP, M68k assembly, Objective C, Python, Pike, Scheme, Emacs lisp, C++, tcl, awk, x86 and x86_64 assembly, CPU microcode, ARM assembly, Javascript, Java, OpenGL Shading Language, Verilog, go.

VIC 20, C64, Amiga, Xerox lisp, Sparc, PC, TKey.

Tema

Ett program som givet en lista av n saker och ett tal k, $0 \leq k \leq n$, skriver ut alla kombinationer av k saker.

```
$ ./all-k-of-n 2 foo bar baz quux
foo bar
foo baz
bar baz
foo quux
bar quux
baz quux
```

Idéer?

Sidospår 1: Binomialkoefficienter

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

Sätt att välja k av n

$$\binom{n}{0} = \binom{n}{n} = 1$$

$$\binom{n}{k} = \underbrace{\binom{n-1}{k-1}}_{\text{med första}} + \underbrace{\binom{n-1}{k}}_{\text{utan första}}$$

$n > k > 0$

Sidospår 1: Binomialkoefficienter

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

$$\binom{n}{0} = \binom{n}{n} = 1$$

$$\binom{n}{k} = \underbrace{\binom{n-1}{k-1}}_{\text{med första}} + \underbrace{\binom{n-1}{k}}_{\text{utan första}}$$

Sätt att välja k av n

$n > k > 0$

1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1
1 6 15 20 15 6 1

Pascals triangel

Sidospår 2: Rekursion

En funktion som anropar sig själv.

```
#! /bin/python3
import sys
def fac(n):
    if n == 0:
        return 1
    return n * fac(n-1)

print(fac(int(sys.argv[1])))
```

```
$ ./fac.py 23
25852016738884976640000
```

Python 1

```
#! /bin/python3
import itertools, sys

for list in itertools.combinations(sys.argv[2:], int(sys.argv[1])):
    print(' '.join(list))
```

Python 2

```
#! /bin/python3
import sys

def all_k_of_n(k, args):
    if k == 0 or len(args) == 0:
        return []
    if k == 1:
        return [[x] for x in args]
    first, rest = args[0], args[1:]
    return ([[first] + comb for comb in all_k_of_n(k-1, rest)]
            + all_k_of_n(k, rest))

for list in all_k_of_n(int(sys.argv[1]), sys.argv[2:]):
    print(' '.join(list))
```

Shell script

```
#! /bin/bash
self="$0"

[[ "$#" -ge 1 ]] || { echo "missing argument" >&2; exit 1; }
k="$1"; shift

[[ "$k" -gt 0 && "$k" -le "$#" ]] || exit 0

if [[ "$k" -eq 1 ]] ; then
    printf "%s\n" "$@"
else
    first="$1"; shift
    "${self}" "$((k - 1))" "$@" | while read -r line ; do
        printf "%s %s\n" "${first}" "${line}" ; done
    "${self}" "$k" "$@"
fi
```

Scheme

```
#! /usr/bin/guile \
-e main -s
!#
(define (main args)
  (for-each (lambda (list) (display (string-join list)) (newline))
            (all-k-of-n (string->number (cadr args)) (cddr args))))
(define (all-k-of-n k args)
  (cond
   ((= k 0) '())
   ((= k 1) (map list args))
   ((null? args) '())
   (else (let ((first (car args)) (rest (cdr args)))
           (append (map (lambda (comb) (cons first comb))
                        (all-k-of-n (- k 1) rest))
                   (all-k-of-n k rest))))))
```

Sidospår 3: Två-komplement

För `uint8_t` har bittarna vikt 1, 2, 4, 8, 16, 32, 64, 128.
För negativa ges översta bitten istället vikt -128.

$$100 = 64 + 32 + 4 = \texttt{0b01100100}$$

$$\tilde{}100 = 255 - 100 = \texttt{0b10011011}$$ flippa *alla* bittar

$$-100 = -128 + 16 + 8 + 4 = \texttt{0b10011100}$$ flippa bittar över den lägsta 1:an

Två-komplements aritmetik *definierar* negation som $-x = \tilde{}x + 1$.

HAKMEM 175 (Bill Gosper, MIT AI Laboratory 1972)

```
#include <stdint.h>

/* Return next larger number, with the same number of one bits. See
 * https://iamkate.com/code/hakmem-item-175/ */
static uint64_t
next (uint64_t x) {
  uint64_t low, left, changed, right; // x = xxx0 1111 0000
  low = x & -x;                       //     0000 0001 0000
  left = x + low;                     //     xxx1 0000 0000
  changed = x ^ left;                 //     0001 1111 0000
  right = (changed / low) >> 2;       //     0000 0000 0111
  return left | right;                //     xxx1 0000 0111
}
```

```
#include <stdio.h>
#include <stdlib.h>
#include "next.c"
int main (int argc, char **argv) {
  if (argc < 2 || argc > 65) return 1;
  int k = atoi (argv[1]);
  if (k <= 0) return 0;
  argc -= 2; argv += 2;
  if (k > argc) return 0;

  for (uint64_t mask = ((uint64_t) 1 << k) - 1;
       !(mask >> argc); mask = next (mask)) {
    for (unsigned i = 0; i < argc; i++)
      if ((mask >> i) & 1) printf("%s ", argv[i]);
    printf ("\n");
  }
}
```