

(Downloaded from <https://cs.stanford.edu/~knuth/programs.html> and typeset on September 17, 2017)

**1. Intro.** Sequence M1805, posets with linear order  $1 \dots n$ . Same as upper triangular  $n \times n$  Boolean matrices  $B$  with zeros on diagonal and  $B^2 \subseteq B$ .

The unique(?) thing here is that I use  $2^k$  as an index, not  $k$ ; therefore I can also use  $2^k + 2^l$  as an index into a triangular matrix!

```
#define maxn 9
#define maxnn (1 << (maxn - 1))
#include <stdio.h>
int row[maxnn + 1];    /* row[1 << (n - j)] is jth row of B */
int mask[maxnn + (maxnn >> 1) + 1];
/* mask[1 << (n - j)] shows bits that must be zero in jth row */
int sols;
main()
{
    register int l, x, y, z;
    int n, nn;
    for (n = 3; n ≤ maxn; n++) {
        sols = 0;
        l = nn = 1 << (n - 1);
        for (x = 2; x ≤ l; x <= 1) mask[x] = 0;
newlev: if (l ≡ 2) {
            sols += 2 - (mask[2] & 1);
            goto backtrack;
        }
        mask[l] &= l - 1;
        row[l] = 0;
        l >>= 1;
        goto newlev;
backtrack: l <= 1, x = row[l];
        for (y = x & (x + 1); y; y -= z) z = y & -y, mask[z] = mask[l + z];
        x = (x | mask[l]) + 1;
        if (x ≥ l) {
            if (l ≡ nn) goto done;
            goto backtrack;
        }
        row[l] = x = x & ~mask[l];
        for (y = x & (x + 1), x = x ⊕ -1; y; y -= z) z = y & -y, mask[l + z] = mask[z], mask[z] |= x;
        l >>= 1;
        goto newlev;
done: printf("%d solutions for %d.\n", sols, n);
    }
}
```

**2. Index.**

*backtrack*: [1](#).

*done*: [1](#).

*l*: [1](#).

*main*: [1](#).

*mask*: [1](#).

*maxn*: [1](#).

*maxnn*: [1](#).

*n*: [1](#).

*newlev*: [1](#).

*nn*: [1](#).

*printf*: [1](#).

*row*: [1](#).

*sols*: [1](#).

*x*: [1](#).

*y*: [1](#).

*z*: [1](#).

# POSETS0

	Section	Page
Intro .....	<a href="#">1</a>	1
Index .....	<a href="#">2</a>	2