

(Downloaded from <https://cs.stanford.edu/~knuth/programs.html> and typeset on May 28, 2023)

1. Intro. An experimental forward method for topswops, using genlex permutation generation.

```
#define n 16      /* degree of perms; should be at most 16 */
#include <stdio.h>
unsigned char score[] = {0, 0, 1, 2, 4, 7, 10, 16, 22, 30, 38, 51, 65, 80, 101, 113, 114};
/* should have n + 1 entries */
typedef char perm [16];
perm p;          /* current permutation for card choices */
perm v;          /* inversion control for perm generation */
perm h;          /* index of largest element not yet in final position */
perm b, bb;      /* buffers used to print solutions */
struct {
    perm q;
} s[16], a;
int d[16];        /* distances so far */
int profile[16];

main()
{
    register int j, k, l, t, c;
    for (k = 1; k < n; k++) p[k - 1] = k + 1, a.q[k - 1] = -k;
    p[n - 1] = 1;
    v[0] = 1;
    h[1] = n;
    profile[0] = 1;
    l = 1;
    s[l] = a;
    j = n - 1;
advance: j--;
tryit: ⟨ Go up a level, except goto infeas if j is infeasible 2 ⟩;
infeas:
    if (j) goto advance;
backup: l--;
nextv:
    if (v[l] ≡ 0) {
        t = p[l - 1], p[l - 1] = p[n - 2], p[n - 2] = t;
        goto backup;
    }
    if (l) {
        j = v[l] - 1;
        t = p[l - 1], p[l - 1] = p[n - 3 - j], p[n - 3 - j] = t;
        a = s[l];
        goto tryit;
    }
    ⟨ Print the stats 4 ⟩;
}
```

2. $\langle \text{Go up a level, except } \mathbf{goto} \text{ } \mathit{infeas} \text{ if } j \text{ is infeasible } 2 \rangle \equiv$

```

k = p[n - 2 - j];
if (k  $\equiv$  -a.q[0]) goto infeas;
t = h[l];
c = d[l] + 1;
if (k  $\equiv$  t) {
    for (t = 1; a.q[t]  $\equiv$  k - t; t++) ;
    if (c + score[k - t] < score[n]) goto infeas;
} else if (c + score[t] < score[n]) goto infeas;
v[l] = j, p[n - 2 - j] = p[l - 1], p[l - 1] = k;
while (1) {
    a.q[0] = a.q[k - 1], a.q[k - 1] = k;
    for (j = 1, k -= 2; j < k; j++, k--) t = a.q[j], a.q[j] = a.q[k], a.q[k] = t;
    k = a.q[0];
    if (k  $\leq$  0) break;
    c++;
}
profile[l]++;
if (l  $\equiv$  n - 1) {
    if (c  $\geq$  score[n])  $\langle$  Record and print the solution 3  $\rangle$ ;
    goto nextv;
}
for (t = h[l]; a.q[t - 1]  $\equiv$  t; t--) ;
l++;
s[l] = a, d[l] = c, h[l] = t;
j = n - l;
goto advance;

```

This code is used in section 1.

3. $\langle \text{Record and print the solution } 3 \rangle \equiv$

```

{
    score[n] = c;
    printf("%d:", c);
    for (k = 1; k  $\leq$  n; k++) b[k - 1] = -k;
    for (k = 1; k  $\leq$  n; k++) {
        while (b[0] > 0)
            for (j = b[0], b[0] = b[j - 1], b[j - 1] = j, c = 1, j -= 2; c < j; c++, j--)
                t = b[c], b[c] = b[j], b[j] = t;
        bb[-b[0] - 1] = p[k - 1];
        b[0] = p[k - 1];
    }
    for (k = 0; k < n; k++) printf("_%d", bb[k]);
    printf("_->_1");
    for (k = 1; k < n; k++) printf("_%d", a.q[k]);
    printf("\n"); fflush(stdout);
}

```

This code is used in section 2.

4. $\langle \text{Print the stats } 4 \rangle \equiv$

```

for (k = 0; k < n; k++) printf("%9d_nodes_ at_ level_ %d. \n", profile[k], k);

```

This code is used in section 1.

5. Index.

a: [1](#).
advance: [1](#), [2](#).
b: [1](#).
backup: [1](#).
bb: [1](#), [3](#).
c: [1](#).
d: [1](#).
fflush: [3](#).
h: [1](#).
infeas: [1](#), [2](#).
j: [1](#).
k: [1](#).
l: [1](#).
main: [1](#).
n: [1](#).
nextv: [1](#), [2](#).
p: [1](#).
perm: [1](#).
printf: [3](#), [4](#).
profile: [1](#), [2](#), [4](#).
q: [1](#).
s: [1](#).
score: [1](#), [2](#), [3](#).
stdout: [3](#).
t: [1](#).
tryit: [1](#).
v: [1](#).

⟨ Go up a level, except **goto *infeas*** if j is infeasible 2 ⟩ Used in section 1.
⟨ Print the stats 4 ⟩ Used in section 1.
⟨ Record and print the solution 3 ⟩ Used in section 2.

TOPSWOPS-FWD

	Section	Page
Intro	1	1
Index	5	3