§1 TOPSWOPS-FWD INTRO 1

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

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 \*/ /\* inversion control for perm generation \*/ **perm** v; /\* index of largest element not yet in final position \*/**perm** h; **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:  $\langle Go \text{ up a level, except goto } infeas \text{ if } j \text{ is infeasible } 2 \rangle$ ; infeas:if (j) goto advance; backup: l--;nextv:if  $(v[l] \equiv 0)$  { 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;  $\langle \text{ Print the stats 4} \rangle;$ 

2 Intro topswops-fwd §2

```
2. \langle Go up a level, except goto infeas if j 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 \le 0) break;
    c++;
  }
  profile[l]++;
  if (l \equiv n-1) {
    if (c \geq score[n]) (Record and print the solution 3);
     goto nextv;
  for (t = h[l]; a.q[t-1] \equiv t; t--);
  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 \le n; k++) b[k-1] = -k;
     for (k = 1; k \le 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 \leftrightarrow) printf("%9d_nodes_at_level_kd.\n", profile[k], k);
This code is used in section 1.
```

 $\S 5$  Topswops-fwd index 3

## 5. Index.

```
a: \underline{1}.
advance: \underline{1}, \underline{2}.
b: \underline{1}.
backup: \underline{1}.
bb: \underline{1}, \underline{3}.
c: <u>1</u>.
d: \underline{\mathbf{1}}.
fflush: 3.
h: \underline{1}.
infeas: \underline{1}, \underline{2}.
j: \underline{1}. k: \underline{1}.
l: <u>1</u>.
main: \underline{1}.
n: \underline{1}.
nextv: \underline{1}, \underline{2}.
p: \underline{1}.
perm: \underline{1}.
\begin{array}{cccc} printf: & \overline{3}, & 4. \\ profile: & \underline{1}, & 2, & 4. \end{array}
q: \underline{1}.
s: \underline{1}.
score: \underline{1}, 2, 3.
stdout: 3.
t: \underline{\mathbf{1}}.
tryit: \underline{1}.
v: \underline{1}.
```

4 NAMES OF THE SECTIONS

TOPSWOPS-FWD

```
 \begin{array}{ll} \langle \mbox{ Go up a level, except $\it goto infeas$ if $j$ is infeasible $\it 2$} \rangle & \mbox{Used in section 1.} \\ \langle \mbox{ Print the stats 4} \rangle & \mbox{ Used in section 1.} \\ \langle \mbox{ Record and print the solution 3} \rangle & \mbox{ Used in section 2.} \\ \end{array}
```

## TOPSWOPS-FWD

	Section	Page
Intro		1
Indev	5	•