

**1\*** **Introduction.** This is a hastily written implementation of the daghull algorithm.

```

format Graph int /* gb_graph defines the Graph type and a few others */
format Vertex int
format Arc int
format Area int
#include "gb_graph.h"
#include "gb_rand.h"
#include "gb_miles.h"
int n = 128;
int mapping[10000];
⟨Global variables 2⟩
⟨Procedures 13*⟩
main(argc, argv)
    int argc;
    char **argv;
{
    ⟨Local variables 7⟩
    Graph *g;
    int kk, kkk, xrnd, yrnd;
    char str[10];
    if (argc ≠ 2) n = 100;
    else if (sscanf(argv[1], "%d", &n) ≠ 1) {
        printf("Usage: %s [n]\n", argv[0]);
        exit(1);
    }
    else if (n < 20 ∨ n > 10000) {
        printf("n should be at least 20 and at most 10000!\n");
        exit(1);
    }
    g = gb_new_graph(n);
    gb_init_rand(0);
    for (kk = 0; kk < n; kk++) mapping[kk] = kk;
    for (kk = 0, v = g-vertices; kk < n; kk++, v++) {
        kkk = gb_next_rand() % (n - kk);
        v-x.I = mapping[kkk];
        mapping[kkk] = mapping[n - kk - 1];
        sprintf(str, "%d", v-x.I);
        v-name = gb_save_string(str);
    }
    mems = ccs = 0;
    ⟨Find convex hull of g 8⟩;
    printf("Total of %d mems and %d calls on ccw.\n", mems, ccs);
}

```

**13\*** **Determinants.** I need code for the primitive function *ccw*. Floating-point arithmetic suffices for my purposes.

We want to evaluate the determinant

$$ccw(u, v, w) = \begin{vmatrix} u(x) & u(y) & 1 \\ v(x) & v(y) & 1 \\ w(x) & w(y) & 1 \end{vmatrix} = \begin{vmatrix} u(x) - w(x) & u(y) - w(y) \\ v(x) - w(x) & v(y) - w(y) \end{vmatrix}.$$

⟨Procedures 13\*⟩ ≡

```

int ccw(u, v, w)
    Vertex *u, *v, *w;
    { register det = 1, ux = u-x.I, vx = v-x.I, wx = w-x.I, t;
      if (ux > vx) {
        t = ux; ux = vx; vx = t; det = -det;
      }
      if (vx > wx) {
        t = vx; vx = wx; wx = t; det = -det;
      }
      if (ux > vx) {
        det = -det;
      }
      if (n < 150)
        printf("cc(%s; %s; %s) is %s\n", u-name, v-name, w-name, det > 0 ? "true" : "false");
        ccs++;
      return (det > 0);
    }

```

This code is used in section 1\*.

**14\* Index.**

The following sections were changed by the change file: 1, 13, 14.

**Arc:** 4, 5, 7.

**Area:** 5.

*argc:* 1\*

*argv:* 1\*

*ccs:* 1\*, 2, 13\*

*ccw:* 2, 10, 11, 13\*

*det:* 13\*

*exit:* 1\*

*first\_inst:* 4, 5, 6, 10, 12.

*g:* 1\*

*gb\_alloc:* 4.

*gb\_graph:* 1\*

*gb\_init\_rand:* 1\*

*gb\_new\_graph:* 1\*

*gb\_next\_rand:* 1\*

*gb\_save\_string:* 1\*

**Graph:** 1\*

*init\_area:* 6.

*inst:* 3, 6, 11, 12.

*kk:* 1\*

*kkk:* 1\*

*main:* 1\*

*mapping:* 1\*

*mems:* 1\*, 2.

*n:* 1\*

*name:* 1\*, 6, 9, 12, 13\*

*next:* 3, 6, 10, 11, 12.

*next\_inst:* 4, 5, 6, 11, 12.

*o:* 2.

*oo:* 2, 6, 8, 10, 11.

*p:* 7.

*pred:* 3, 6, 10, 11.

*printf:* 1\*, 6, 9, 12, 13\*

*q:* 7.

*r:* 7.

*rover:* 5, 6, 9, 11.

*s:* 7.

*serial\_no:* 5, 8.

*sprintf:* 1\*

*sscanf:* 1\*

*str:* 1\*

*succ:* 3, 6, 9, 11.

*t:* 13\*

*tip:* 3, 6, 10, 12.

*u:* 7, 13\*

*ux:* 13\*

*v:* 7, 13\*

**Vertex:** 5, 7, 13\*

*vertices:* 1\*, 6, 8.

*vv:* 7, 8, 10, 11, 12.

*vx:* 13\*

*w:* 7, 13\*

*working\_storage:* 4, 5, 6.

*wx:* 13\*

*xrnd:* 1\*

*yrnd:* 1\*

- ⟨ Compile two new instructions, for  $(u, vv)$  and  $(vv, v)$  12 ⟩ Used in section 11.
- ⟨ Find convex hull of  $g$  8 ⟩ Used in section 1\*.
- ⟨ Follow the instructions; **continue** if  $vv$  is inside the current hull 10 ⟩ Used in section 8.
- ⟨ Global variables 2, 5 ⟩ Used in section 1\*.
- ⟨ Initialize the array of instructions 4 ⟩ Used in section 6.
- ⟨ Initialize the data structures 6 ⟩ Used in section 8.
- ⟨ Local variables 7 ⟩ Used in section 1\*.
- ⟨ Print the convex hull 9 ⟩ Used in section 8.
- ⟨ Procedures 13\* ⟩ Used in section 1\*.
- ⟨ Update the convex hull, knowing that  $vv$  lies outside the consecutive hull vertices  $u$  and  $v$  11 ⟩ Used in section 8.