

1. Intro. Given a graph g with m edges, make data from which DLX2 should tell us all ways to label the vertices, using distinct labels in $\{0, 1, \dots, m\}$, so that the edges have distinct difference. (Those differences will be $\{1, \dots, m\}$).

Each label could be complemented with respect to m . I avoid this by “orienting” the edge labeled m .

```
#define encode(x) ((x) < 10 ? (x) + '0' : (x) < 36 ? (x) - 10 + 'a' : (x) < 62 ? (x) - 36 + 'A' : (x) + 99)
#define maxm 156 /* based on that encoding, but I could go higher in a pinch! */
```

```
#include <stdio.h>
#include <stdlib.h>
#include "gb_graph.h"
#include "gb_save.h"
int c;
main(int argc, char *argv[])
{
    register int i, j, k, m, n;
    register Arc *a;
    register Graph *g;
    register Vertex *v;
    < Process the command line 2 >;
    < Output the item-name line 3 >;
    for (k = 1; k ≤ m; k++) < Output the options for edge k 4 >;
}
```

2. < Process the command line 2 > \equiv

```
if (argc ≠ 2) {
    fprintf(stderr, "Usage: %s %s\n", argv[0]);
    exit(-1);
}
g = restore_graph(argv[1]);
if (!g) {
    fprintf(stderr, "I couldn't reconstruct graph %s!\n", argv[1]);
    exit(-2);
}
m = g-m/2, n = g-n;
if (m ≥ maxm) {
    fprintf(stderr, "Sorry, at present I require %d!\n", maxm);
    exit(-3);
}
printf("%s\n", argv[0], argv[1]);
```

This code is used in section 1.

3. There's a primary item k for each edge label, and a primary item uv for each edge. This enforces a permutation between edges and labels.

There's a secondary item $.v$ for each vertex; its color will be its label.

There's a secondary item $+k$ for each vertex label; its color will be the vertex so labeled.

⟨ Output the item-name line 3 ⟩ \equiv

```

for ( $k = 1$ ;  $k \leq m$ ;  $k++$ ) printf("%c□", encode( $k$ ));
for ( $v = g\text{-vertices}$ ;  $v < g\text{-vertices} + n$ ;  $v++$ )
  for ( $a = v\text{-arcs}$ ;  $a$ ;  $a = a\text{-next}$ )
    if ( $a\text{-tip} > v$ ) printf("%s-%s□",  $v\text{-name}$ ,  $a\text{-tip-name}$ );
printf("|");
for ( $v = g\text{-vertices}$ ;  $v < g\text{-vertices} + n$ ;  $v++$ ) printf("□.%s",  $v\text{-name}$ );
for ( $k = 0$ ;  $k \leq m$ ;  $k++$ ) printf("□+%c", encode( $k$ ));
printf("\n");

```

This code is used in section 1.

4. `#define vrt(v) ((int)((v) - g-vertices))`

⟨ Output the options for edge k 4 ⟩ \equiv

```

{
  for ( $i = 0, j = k$ ;  $j \leq m$ ;  $i++, j++$ ) {
    for ( $v = g\text{-vertices}$ ;  $v < g\text{-vertices} + n$ ;  $v++$ )
      for ( $a = v\text{-arcs}$ ;  $a$ ;  $a = a\text{-next}$ )
        if ( $a\text{-tip} > v$ ) {
          printf("%c□%s-%s□.%s:%c□.%s:%c□+%c:%c□+%c:%c\%n", encode( $k$ ),  $v\text{-name}$ ,  $a\text{-tip-name}$ ,
             $v\text{-name}$ , encode( $i$ ),  $a\text{-tip-name}$ , encode( $j$ ), encode( $i$ ), encode( $vrt(v)$ ), encode( $j$ ),
            encode( $vrt(a\text{-tip})$ ));
          if ( $i \neq 0 \vee j \neq m$ ) /* prevent complementation symmetry */
            printf("%c□%s-%s□.%s:%c□.%s:%c□+%c:%c□+%c:%c\%n", encode( $k$ ),  $v\text{-name}$ ,  $a\text{-tip-name}$ ,
               $v\text{-name}$ , encode( $j$ ),  $a\text{-tip-name}$ , encode( $i$ ), encode( $j$ ), encode( $vrt(v)$ ), encode( $i$ ),
              encode( $vrt(a\text{-tip})$ ));
        }
      }
    }
}

```

This code is used in section 1.

5. Index.*a*: [1](#).**Arc**: [1](#).*arcs*: [3](#), [4](#).*argc*: [1](#), [2](#).*argv*: [1](#), [2](#).*c*: [1](#).*encode*: [1](#), [3](#), [4](#).*exit*: [2](#).*fprintf*: [2](#).*g*: [1](#).**Graph**: [1](#).*i*: [1](#).*j*: [1](#).*k*: [1](#).*m*: [1](#).*main*: [1](#).*maxm*: [1](#), [2](#).*n*: [1](#).*name*: [3](#), [4](#).*next*: [3](#), [4](#).*printf*: [2](#), [3](#), [4](#).*restore_graph*: [2](#).*stderr*: [2](#).*tip*: [3](#), [4](#).*v*: [1](#).**Vertex**: [1](#).*vertices*: [3](#), [4](#).*vrt*: [4](#).

- ⟨Output the item-name line 3⟩ Used in section 1.
- ⟨Output the options for edge k 4⟩ Used in section 1.
- ⟨Process the command line 2⟩ Used in section 1.

GRACEFUL-DLX

	Section	Page
Intro	1	1
Index	5	3