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 $\{0, 1, \ldots, m\}$, so that the edges have distinct difference. (Those differences will be $\{1, \ldots, m\}$.

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⟩ ≡

```
  if (argc ≠ 2) {
    fprintf(stderr, "Usage:␣%s␣foo.gb\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␣m<%d!\n", maxm);
    exit(−3);
  }
  printf("|␣%s␣%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 ⟩ ≡

```
for (k = 1; k ≤ m; k++) printf("%c␣", encode(k));
for (v = g→vertices; v < g→vertices + n; v++)
  for (a = v→arcs; a; a = a→next)
    if (a→tip > v) printf("%s-%s␣", v→name, a→tip→name);
printf("|");
for (v = g→vertices; v < g→vertices + n; v++) printf("␣.%s", v→name);
for (k = 0; k ≤ 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 ⟩ ≡

```
{
  for (i = 0, j = k; j ≤ m; i++, j++) {
    for (v = g→vertices; v < g→vertices + n; v++)
      for (a = v→arcs; a; a = a→next)
        if (a→tip > v) {
          printf("%c␣%s-%s␣.%s:%c␣.%s:%c␣+%c:%c␣+%c:%c\n", encode(k), v→name, a→tip→name,
              v→name, encode(i), a→tip→name, encode(j), encode(i), encode(vrt(v)), encode(j),
              encode(vrt(a→tip)));
          if (i ≠ 0 ∨ j ≠ m)    /* prevent complementation symmetry */
            printf("%c␣%s-%s␣.%s:%c␣.%s:%c␣+%c:%c␣+%c:%c\n", encode(k), v→name, a→tip→name,
                v→name, encode(j), a→tip→name, encode(i), encode(j), encode(vrt(v)), encode(i),
                encode(vrt(a→tip)));
        }
  }
}
```

This code is used in section 1.

5. Index.

GRACEFUL-DLX