19 Undirected graphs

A *Graph* is a pair (V, E) of *vertices* and *undirected edges.*

There is a notation which we don't use much:

$$V^{(2)} = \{\{u, v\} : \ u, v \in V \wedge u \neq v\}$$

the set of all possible undirected edges with vertices in V.

So:

(19.1) Definition *An (undirected)* graph *is a pair* (V, E) *of vertices and (undirected) edges where* V *is any finite set and* $E \subseteq V^{(2)}$.

In fact, we shall represent graphs as 'bidirected graphs,' which are identical to directed graphs, except that whenever there is an edge $e = (u, v)$, there is another edge $e' = (v, u)$, and every edge carries an extra 'inverse' component; e' is the inverse of e and e is the inverse of e'.

For the purposes of this module all graphs will be input as bidirected graphs, as, for example

```
6 14
 0  3  4  2  5
 1  2  3  5
 2  2  4  0
 3  2  1  5
 4  2  0  2
 5  3  3  1  0
```

After a graph has been read in, it is necessary to install the 'inverse' links in the edges.

The code samples include a program `bcc.c` whose purpose is to calculate the biconnected components of G, see below. It has a routine `link_inverse_edges` which installs these links:

```
void link_inverse_edges ( GRAPH * graph )
{
  int i;
  for (i=0; i<graph->n; ++i)
  {
    int j;
    EDGE * e = graph->an_edge[i];
    for (j=0; j<graph->out_deg[i]; ++j)
    {
      if ( e->inverse == NULL )
      {
        int k;
        EDGE * ee = graph->an_edge[e->to];
        int found = 0;
        for ( k=0; k<graph->out_deg[e->to] && (!found); ++k )
        {
          if ( ee->to == i )
```

```
          { e->inverse = ee; ee->inverse = e; found = 1; }
          ee = ee->next;
        }
      }
      e = e->next;
    }
  }
}
```

This code is 'inefficient' because in order to set up the inverse link from an edge, it inspects all the edges in the graph. In other words, the runtime is $\Omega(m^2)$ where m is the number of edges.

It is possible to accomplish this in linear time, $O(m)$, using a variant of lexical sort, and the third programming assignment is to replace the above piece of code by a more efficient one (of course any improvement in the runtime will not be noticeable).