cgi_decode

```
/**
  * @title cgi_decode
  * @desc
  * Translate a string from the CGI encoding to plain ascii text
  * '+' becomes space, %xx becomes byte with hex value xx,
  * other alphanumeric characters map to themselves
  *
  * returns 0 for success, positive for erroneous input
  * 1 = bad hexadecimal digit
  */

int cgi_decode(char *encoded, char *decoded)
{
    char *eptr = encoded;
    char *dptr = decoded;
    int ok = 0;
```

A

1

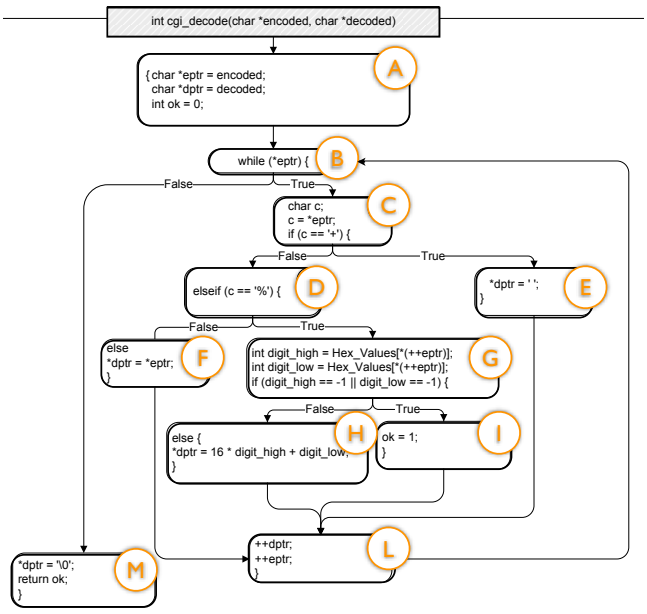
Here's an ongoing example. The function cgi_decode translates a CGI-encoded string (i.e., from a Web form) to a plain ASCII string, reversing the encoding applied by the common gateway interface (CGI) on common Web servers.
(from Pezze + Young, "Software Testing and Analysis", Chapter 12)

```
    while (*eptr)  /* loop to end of string ('\0' character) */
    {
        char c;
        c = *eptr;
        if (c == '+') {  /* '+' maps to blank */
            *dptr = ' ';
        } else if (c == '%') { /* '%xx' is hex for char xx */
            int digit_high = Hex_Values[*(++eptr)];
            int digit_low  = Hex_Values[*(++eptr)];
            if (digit_high == -1 || digit_low == -1)
                ok = 1; /* Bad return code */
            else
                *dptr = 16 * digit_high + digit_low;
        } else { /* All other characters map to themselves */
           *dptr = *eptr;
        }
        ++dptr; ++eptr;
    }

    *dptr = '\0';   /* Null terminator for string */
    return ok;
}
```

B C E D G I H F L M

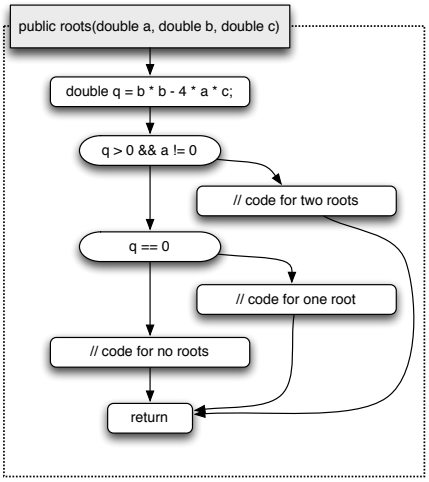
2



3

This is what cgi_decode looks as a CFG.
(from Pezze + Young, "Software Testing and Analysis", Chapter 12)

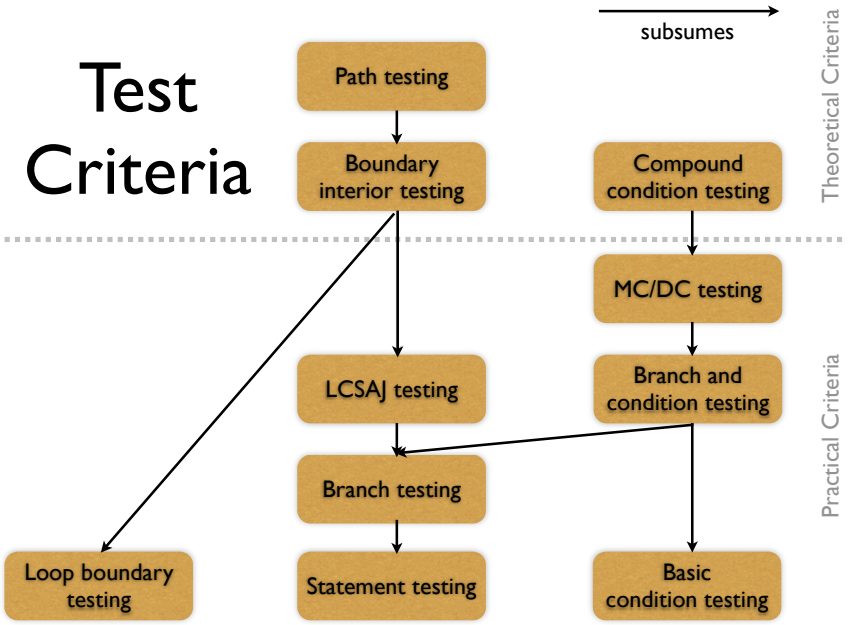
Control Flow Graph



- A *control flow graph* expresses paths of program execution
- *Nodes* are *basic blocks* – sequences of statements with one entry and one exit point
- *Edges* represent *control flow* – the possibility that the program execution proceeds from the end of one basic block to the beginning of another

To talk about structure, we turn the program into a *control flow graph*, where statements are represented as nodes, and edges show the possible control flow between statements.

Test Criteria



And this is the summary of structural testing techniques.

Weyuker's Hypothesis

The adequacy of a coverage criterion can only be intuitively defined.

Established by a number of studies done by E. Weyuker at AT&T. "Any explicit relationship between coverage and error detection would mean that we have a fixed distribution of errors over all statements and paths, which is clearly not the case".