NAME

Algorithm::DiffOld - Compute 'intelligent' differences between two files / lists but use the old (<=0.59) interface.

NOTE

This has been provided as part of the Algorithm::Diff package by Ned Konz. This particular module is **ONLY** for people who **HAVE** to have the old interface, which uses a comparison function rather than a key generating function.

Because each of the lines in one array have to be compared with each of the lines in the other array, this does M*N comparisions. This can be very slow. I clocked it at taking 18 times as long as the stock version of Algorithm::Diff for a 4000-line file. It will get worse quadratically as array sizes increase.

SYNOPSIS

```
use Algorithm::DiffOld qw(diff LCS traverse_sequences);

@lcs = LCS( \@seq1, \@seq2, $comparison_function );

$lcsref = LCS( \@seq1, \@seq2, $comparison_function );

@diffs = diff( \@seq1, \@seq2, $comparison_function );

traverse_sequences( \@seq1, \@seq2,
{ MATCH => $callback,
DISCARD_A => $callback,
DISCARD_B => $callback,
},
$comparison_function );
```

COMPARISON FUNCTIONS

Each of the main routines should be passed a comparison function. If you aren't passing one in, **use Algorithm::Diff instead**.

These functions should return a true value when two items should compare as equal.

For instance,

```
@lcs = LCS( \@seq1, \@seq2, sub { my ($a, $b) = @_; $a eq $b } );
```

but if that is all you're doing with your comparison function, just use Algorithm::Diff and let it do this (this is its default).

Or:

```
sub someFunkyComparisonFunction
{
my ($a, $b) = @_;
$a =~ m{$b};
}

@diffs = diff( \@lines, \@patterns, \&someFunkyComparisonFunction );
```

which would allow you to diff an array `@lines` which consists of text lines with an array `@patterns` which consists of regular expressions.

This is actually the reason I wrote this version — there is no way to do this with a key generation function as in the stock Algorithm::Diff.