Алгоритм Левенштейна

```php
function distance($source, $dest) {
    if ($source == $dest) {
        return 0;
    }

    list($slen, $dlen) = [strlen($source), strlen($dest)];

    if ($slen == 0 || $dlen == 0) {
        return $dlen ? $dlen : $slen;
    }

    $dist = range(0, $dlen);
//Array ( [0] => 0 [1] => 1 [2] => 2 [3] => 3 [4] => 4 [5] => 5 [6]
=> 6 [7] => 7 [8] => 8 [9] => 9 [10] => 10 [11] => 11 [12] => 12 )

    for ($i = 0; $i < $slen; $i++) {
        $ dist = [$i + 1];
        for ($j = 0; $j < $dlen; $j++) {
            $cost = ($source[$i] == $dest[$j]) ? 0 : 1;
            $ dist[$j + 1] = min(
                $dist[$j + 1] + 1,     // deletion
                $ dist[$j] + 1,        // insertion
                $dist[$j] + $cost      // substitution
            );
        }
        $dist = $_dist;
    }

    return $dist[$dlen];
}

$source = 'PHP';
$dest = 'new PHP test';

echo distance($source, $dest) . "\n";
  echo levenshtein($source, $dest); // built-in PHP function
```

```
for(int i=1;i<=n;i++)
{
   for(int j=1;j<=A;j++)
   {
      if(j-w[i]>=0)
      {
         a[i][j]=max(a[i-1][j],a[i-1][j-w[i]]+v[i]);
         if(a[i-1][j]>a[i-1][j-w[i]]+v[i])
         {
            (from[i][j]).x=i-1;
            (from[i][j]).y=j;
            (from[i][j]).bullion=0;
            (from[i][j]).value=0;
         }
         else
         {
            (from[i][j]).x=i-1;
            (from[i][j]).y=j-w[i];
            (from[i][j]).bullion=i;
            (from[i][j]).value=v[i];
         }
      }
      else
      {
         a[i][j]=a[i-1][j];
         (from[i][j]).x=i-1;
         (from[i][j]).y=j;
         (from[i][j]).bullion=0;
         (from[i][j]).value=0;
      }
   }
}
cout<<"a[n][A]= "<<a[n][A]<<endl;
```

S1,S2 - строки длиной M,N , d(S1,S2) - расстояние Левенштейна, тогда рекуррентное соотношение

удаление
вставка
замена

$d(S_1, S_2) = D(M, N)$, где

$$D(i,j) = \begin{cases} 0, & i=0,\ j=0 \\ i, & j=0,\ i>0 \\ j, & i=0,\ j>0 \\ \min\{ & \\ \quad D(i,j-1)+1, & \text{вставка} \\ \quad D(i-1,j)+1, & \text{удаление} \\ \quad D(i-1,j-1)+\mathrm{m}(S_1[i], S_2[j]) & j>0,\ i>0 \\ \} & \end{cases},$$

m(a,b)=0, если a==b, иначе 1

D(M,N)

Пусть S1 кончается на символ «a»,S2 кончается на символ «b». Есть три варианта:
1)Символ «a», на который кончается S1, в какой-то момент был стёрт. Сделаем это стирание первой операцией. Тогда мы стёрли символ «a», после чего превратили первые i-1 символов S1 в S2 (на что потребовалось D(i-1,j) операций), значит, всего потребовалось D(i-1,j)+1 операций
2)Символ «b», на который кончается S2, в какой-то момент был добавлен. Сделаем это добавление последней операцией. Мы превратили S1 в первые j-1 символов S2, после чего добавили «b». Аналогично предыдущему случаю, потребовалось D(i,j-1)+1 операци
3)Оба предыдущих утверждения неверны. Если мы добавляли символы справа от финального «a», то, чтобы сделать последним символом «b», мы должны были или в какой-то момент добавить его (но тогда утверждение 2 было бы верно), либо заменить на него один из этих добавленных символов (что тоже невозможно, потому что добавление символа с его последующей заменой неоптимально). Значит, символов справа от финального «a» мы не добавляли. Самого финального «a» мы не стирали, поскольку утверждение 1 неверно. Значит, единственный способ изменения последнего символа — его замена. Заменять его 2 или больше раз неоптимально. Значит,
1)Если a=b , мы последний символ не меняли. Поскольку мы его также не стирали и не приписывали ничего справа от него, он не влиял на наши действия, и, значит, мы выполнили D(i-1,j-1) операций.
2)Если a!=b мы последний символ меняли один раз. Сделаем эту замену первой. В дальнейшем, аналогично предыдущему случаю, мы должны выполнить D(i-1,j-1) операций, значит, всего потребуется D(i-1,j-1) +1

```
function lev($source, $dest)
{
        if ($source == $dest) {
    return 0;
}

list($slen, $dlen) = [strlen($source), strlen($dest)];

if ($slen == 0 || $dlen == 0) {
    return $dlen ? $dlen : $slen;
}
        $dist[0][0]=0;
        for($i=0;$i<$slen;$i++)
        {
                $dist[$i][0]=$i;
        }
        for($j=0;$j<$dlen;$j++)
        {
                $dist[0][$j]=$j;
        }
        for($i=1;$i<$slen;$i++)
        {
                for($j=1;$j<$dlen;$j++)
                {

                        $cost = ($source[$i] == $dest[$j]) ? 0 : 1;
                        $dist[$i][$j]=min(
                                $dist[$i][$j-1]+1,
                                $dist[$i-1][$j]+1,
                                $dist[$i-1][$j-1]+$cost
                        );
                        if($dist[$i][$j]==$dist[$i][$j-1]+1)
                        {
                                $from[$i][$j][0]=$i;
                                $from[$i][$j][1]=$j-1;
                                $from[$i][$j][2]=1;
                        }
                        if($dist[$i][$j]==$dist[$i-1][$j]+1)
                        {
                                $from[$i][$j][0]=$i-1;
                                $from[$i][$j][1]=$j;
                                $from[$i][$j][2]=2;
                        }
                        if($dist[$i][$j]==$dist[$i-1][$j-1]+$cost)
                        {
                                $from[$i][$j][0]=$i-1;
                                $from[$i][$j][1]=$j-1;
                                $from[$i][$j][2]=3;
                        }
                }
        }
        print_from($from, $slen-1, $dlen-1, $source, $dest);
        return $dist[$slen-1][$dlen-1];
}


function print_from($from, $x, $y, $source, $dest)
{
        if($from[$x][$y][2]==1)
        {
                echo substr($source,0,$x+1).'      '.substr($dest,0,$y).' op=1 '.'<br>';
        }
        else if($from[$x][$y][2]==2)
        {
                echo substr($source,0,$x).'      '.substr($dest,0,$y+1).' op=2 '.'<br>';
        }
        else if($from[$x][$y][2]==3)
        {
                echo substr($source,0,$x).'      '.substr($dest,0,$y).' op=3 '.'<br>';
        }
        if($x>0 || $y>0)
        {
                print_from($from, $from[$x][$y][0], $from[$x][$y][1], $source, $dest);
        }
}
```