Dynamické programovanie

1. **Určíme podproblém.**
   - aké sú rozmery matice, ktorú budeme vypĺňať?
   - aký je presný význam každého políčka matice?
   - kde v matici nájdeme riešenie pôvodnej úlohy?
2. **Vyriešime podproblém za pomoci iných podproblémov.** Ako vypočítame jedno políčko matice z iných políčiek matice?
3. **Bázové podproblémy.** Ktoré políčka nemožno vypočítať pomocou vzťahov z predchádzajúceho kroku? Aké hodnoty by mali obsahovať?
4. **Vyberieme poradie vypĺňania.** V akom poradí musíme maticu vypĺňať tak, aby sme v každom kroku mali vypočítané všetky políčka, ktoré potrebujeme na výpočet daného políčka?

Najdlhšia spoločná podpostupnosť

```
// base cases
for i:=0 to m do C[i,0]:=0;
for j:=0 to n do C[0,j]:=0;

// filling the matrix
for i:=1 to m do
  for j:=1 to n do
    if X[i]=Y[j] then C[i,j]:=C[i-1,j-1]+1;
    else C[i,j]:=max(C[i-1,j],C[i,j-1]);

return C[m,n];
```

		A	L	G	O	R	I	T	H	M
	0	0	0	0	0	0	0	0	0	0
L	0	0	1	1	1	1	1	1	1	1
O	0	0	1	1	2	2	2	2	2	2
G	0	0	1	2	2	2	2	2	2	2
A	0	1	1	2	2	2	2	2	2	2
R	0	1	1	2	2	3	3	3	3	3
I	0	1	1	2	2	3	4	4	4	4
T	0	1	1	2	2	3	4	5	5	5
H	0	1	1	2	2	3	4	5	6	6
M	0	1	1	2	2	3	4	5	6	7

```
row:=m; col:=n;
lcs:="";

while (row>0 and col>0) do
  if (D[row,col]=upleft) then
    // X[row]=Y[col]
    lcs:=lcs.X[row];
    row:=row-1; col:=col-1;
  else if (D[row,col]=up) then
    row:=row-1;
  else if (D[row,col]=left) then
    col:=col-1;

reverse lcs;
return lcs;
```

Najkratšia triangulácia

```
// base case - j=i+1
for i:=1 to n-1 do
  T[i,i+1]:=D[i,i+1];

for delta:=2 to n-1 do
  // cases where j-i=delta
  for i:=1 to n-delta do
    j:=i+delta;
    T[i,j]:=infinity;
    // try all possible triangles v_i,v_j,v_m
    for m:=i+1 to j-1 do
      cost:=D[i,j]+T[i,m]+T[m,j];
      if cost<T[i,j] then T[i,j]:=cost;

return T[1,n];
```

$M[i,j] : m$, ktoré sme použili na výpočet $T[i,j]$

```
function give_solution(i,j)
  output edge (i,j);
  if j>i+1 then
    give_solution(i,M[i,j]);
    give_solution(M[i,j],j);
```