"Vzor" dôkazu správnosti greedy algoritmu

Lema: Predpokladajme, že greedy algoritmus vráti riešenie G. Potom existuje optimálne riešenie, ktoré sa s riešením G zhoduje na prvých k voľbách.

Dôkaz: Matematickou indukciou podľa k.

Báza indukcie. Pre $k = 0$ – ľubovoľné optimálne riešenie.

Indukčný krok. (Prepokladajme, že sme neurobili chybu pri prvých k voľbách, potom aj $(k+1)$-vá voľba je OK.)

- Predpokladajme, že existuje optimálne riešenie OPT, ktoré sa zhoduje s G na prvých k voľbách.
- Vyrobíme riešenie OPT':
  - OPT' má rovnakú hodnotu ako OPT (a preto je tiež optimálne)
  - OPT' súhlasí s G na jednej ďalšej $(k+1)$-vej voľbe.

Rozmieňanie peňazí

```
coins[0]:=0;
for i:=1 to S do
  min:=infinity;
  for j:=1 to m do
    if d[j]<=i and coins[i-d[j]]<min then
      min:=coins[i-d[j]];
  coins[i]:=1+min;

return coins[S];
```

Time: $\Theta(mS)$

Rozmieňanie peňazí s vypísaním riešenia

```
coins[0]:=0;
for i:=1 to S do
  min:=infinity;
  for j:=1 to m do
    if d[j]<=i and coins[i-d[j]]<min then
      min:=coins[i-d[j]];
*     minchoice:=j;
  coins[i]:=1+min;
* choice[i]:=minchoice;

if coins[S]=infinity then write 'No solution!';
else
  while S>0 do
    write d[choice[S]];
    S:=S-d[choice[S]];
```

Dynamické programovanie—zhrnutie

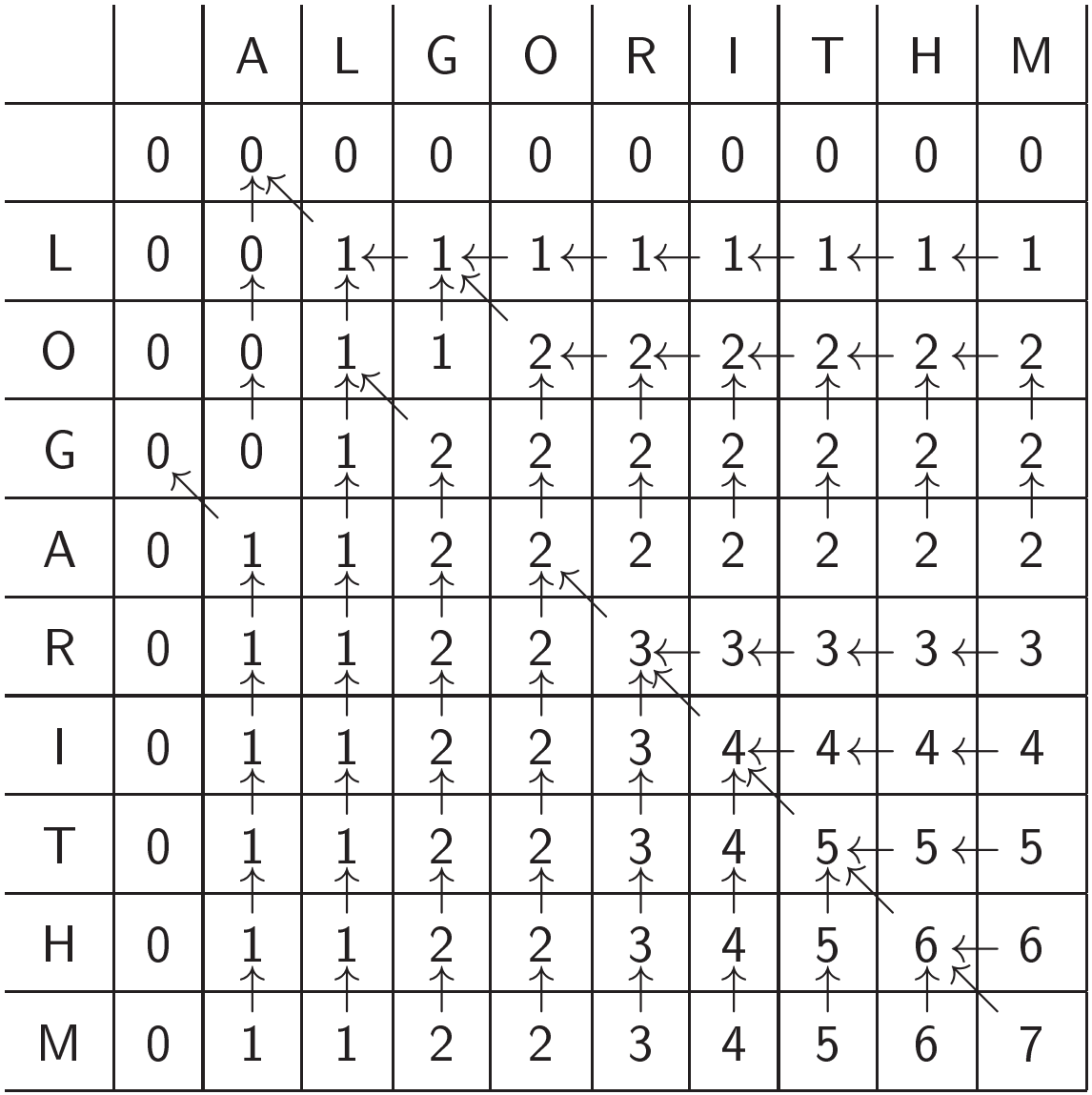
1. **Určíme podproblém.**
   - aké sú rozmery matice, ktorú budeme vypĺňať?
   - aký je presný význam každého políčka matice?
   - kde v matici nájdeme riešenie pôvodnej úlohy?
2. **Vyriešime podproblém za pomoci iných podproblémov.**
   Ako vypočítame jedno políčko matice z iných políčiek matice?
3. **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

Najdlhšia spoločná podpostupnosť—vypísanie riešenia

```
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;
```