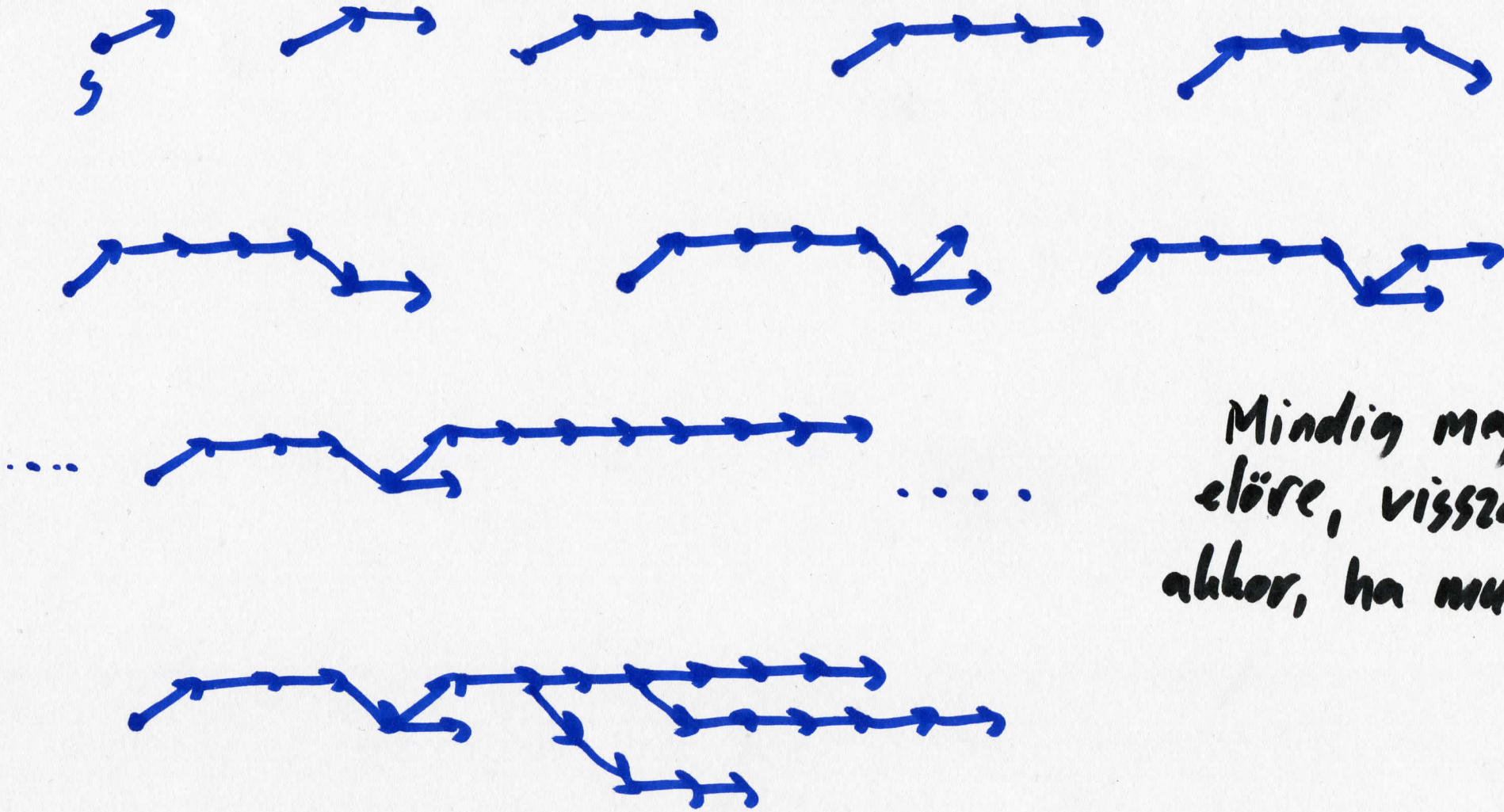
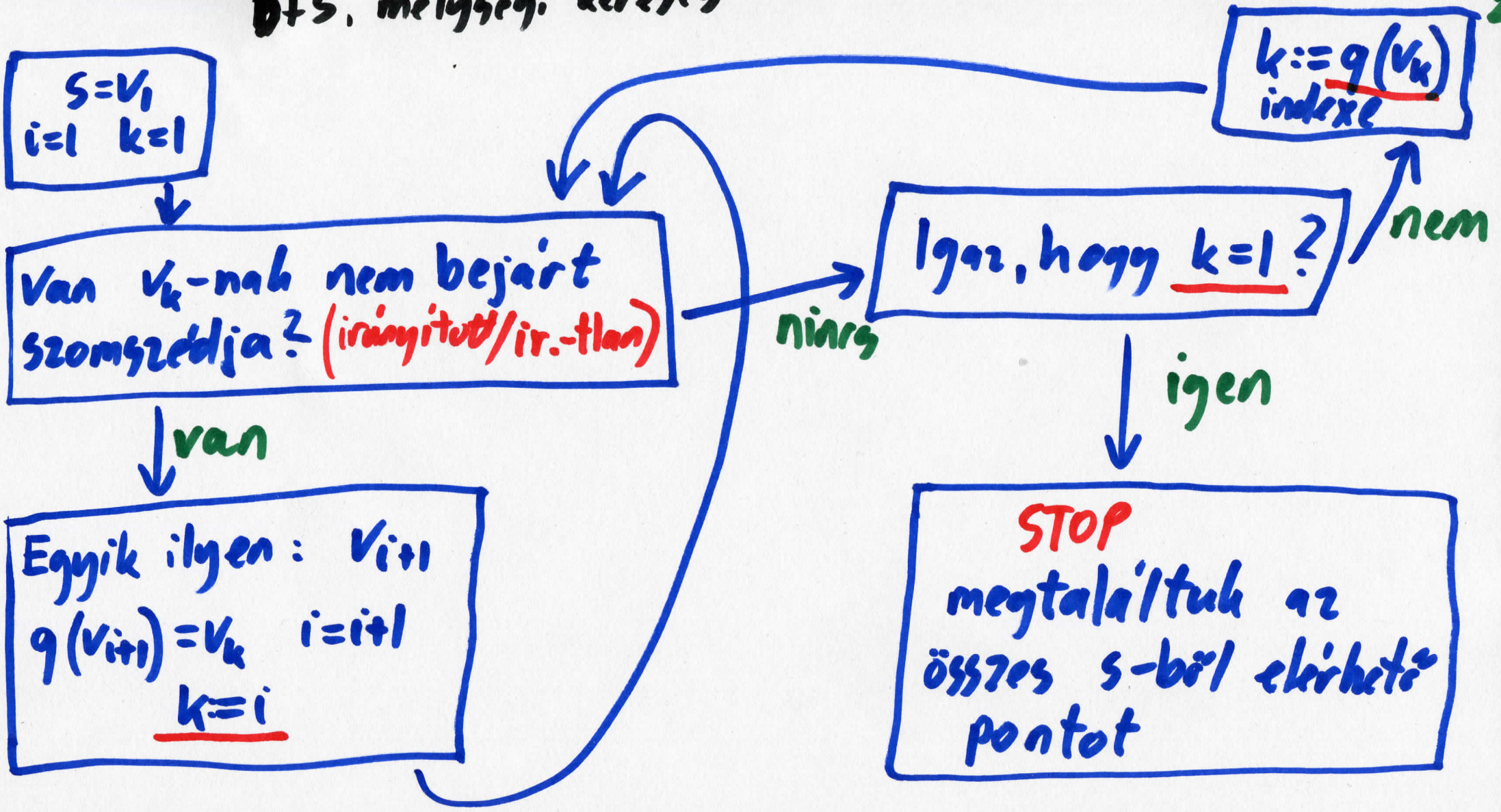
DFS, mélységi bejárás/keresés s-ből

Lehet irányított vagy irányítatlan gráfban.



Mindig megyünk előre, vissza csak akkor, ha muszáj.

DFS, mélységi keresés



q: elől

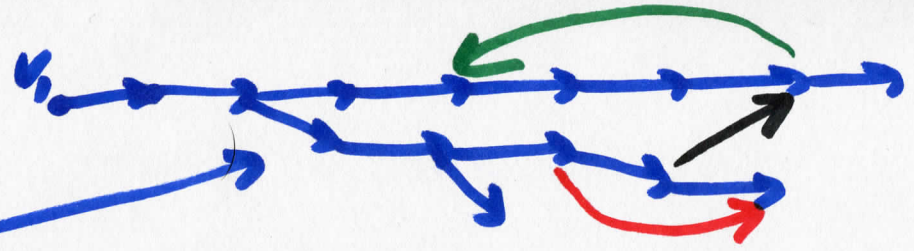
i: utolsó felderített pont

k: ahonnan éppen keresünk

Lépésszám: $c(n+m)$ optimális

Élek osztályozása.

v_1 - Döl mélységi fa



Fa él

Előre él: uv, v u utódja

Vissza él: uv, u v utódja

Keresztél: uv, egyik se a másik utódja

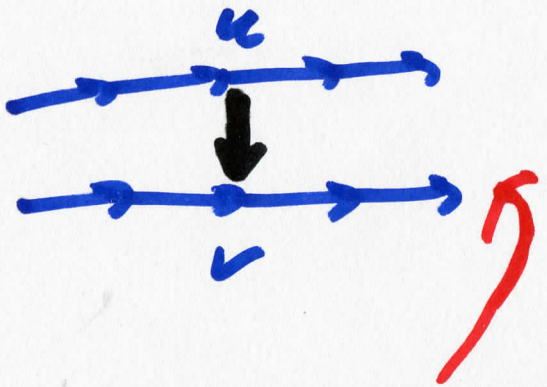
Mélységi szám: amilyen sorrendben megtalálták a csúcsokat
(i) vagy m(v)

Befejezési szám: amilyen sorrendben befejeztük a csúcsokat
b(v)

Ha uv előre él: $m(v) > m(u)$

vissza él: $m(v) < m(u)$

keresztél: $m(v) < m(u)$



Tehát: ha $m(v) > m(u)$

$\Rightarrow$ uv <u>fael</u> vagy előre él.

Ez az ág volt először, különben uv fael lenne.

$m(v) < m(u)$

$b(v) < b(u)$

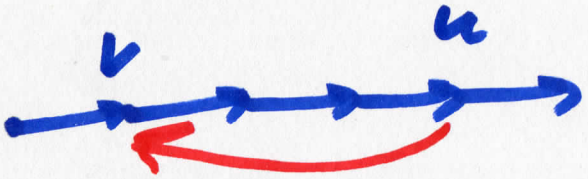
Ha uv vissza él: $b(v) > b(u)$

ha keresztél: $b(v) < b(u)$

Tehát: ha $m(v) < m(u)$, $b(v) < b(u)$:

<u>keresztél</u>

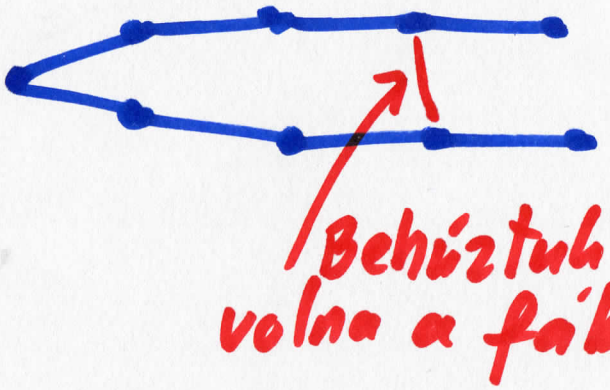
$m(v) < m(u)$, $b(v) > b(u)$:



<u>visszaél</u>.

Irányítatlan gráf mélységi fájában nincs keresztél!

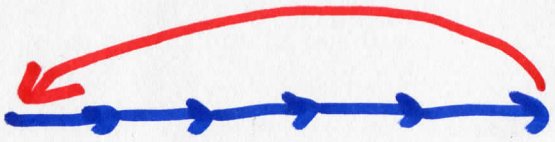
Előre él = visszaél



Directed acyclic graph DAG:
Irányított gráf, nincs irányított kör.

G irányított gráf DAG $\Longleftrightarrow$ mélységi fában (erdőben) nincs visszaél

$\Rightarrow$ trivi: visszaél $\rightarrow$ irányított kör:



G DAG $\Leftrightarrow$ mélységi fában nincs visszaél. $\Rightarrow$ ✓

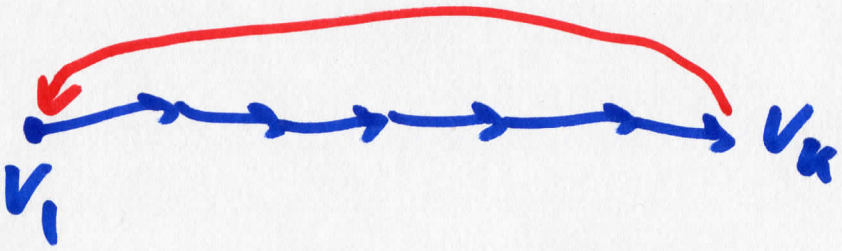
$\Leftarrow$ Biz: Tfh $C = v_1 \dots v_k$ irányított kör.
DFS fa v-ből, $m(v_i)$: a legkisebb C-n. (őt írjuk el először)

Áll: $\forall i$ v_i v_1 leszármazottja.
Biz: indukció i-re. v_1 elérése után minden olyan v_1 leszármazottja, v_1 befejezéséig.

v_2: ha behúztuk a v_1v_2 élt: ✓
ha nem: v_2-t már megtaláltuk: ✓

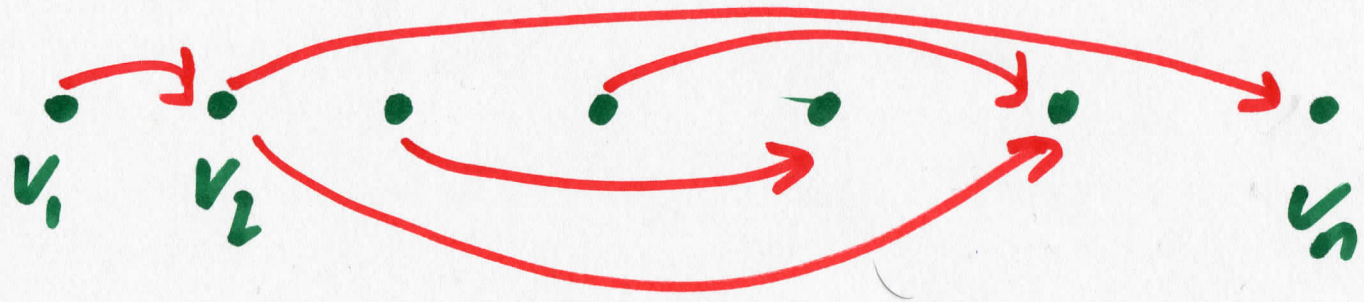
Tfh v_i v_1 leszármazottja.
ha behúztuk a v_iv_{i+1} élt: ✓
ha nem: v_{i+1}-et már megtaláltuk: ✓

Tehát: v_k v_1 leszármazottja! **v_kv_1: visszaél!**



G irányított gráf.

Topologikus rendezés: csúcsok sorrendje, $v_1 \dots v_n$ úgy, hogy csak előre mutató élek vannak:
minden $v_i v_j$ élre $i < j$



G DAG $\Leftrightarrow$ van topologikus rendezése.

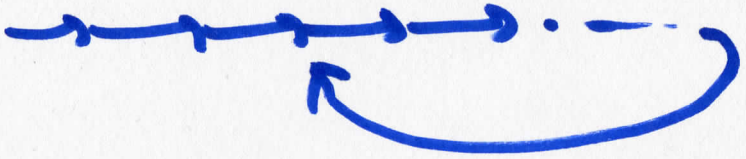
$\Leftarrow$ van top. rendezés $\rightarrow$ minden él előre mutat $\rightarrow$
$\rightarrow$ nincs irányított kör $\rightarrow$ G DAG.

G DAG $\Leftrightarrow$ van topologikus rendezése $\Leftarrow$ ✓

$\Rightarrow$ Biz: G DAG : van nyelő (nyelő: nincs ki-él)

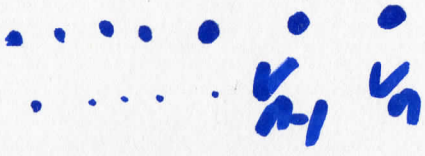
Ha nem lenne:

mindig tovább tudunk menni egy ki-élen .... valahova visszatérünk.



← irányított kör!

Nyelő: tegyük a végére. Maradék: újabb nyelő stb



→ topologikus rendezés.

PERT módszer (Program Evaluation and Review Technique)

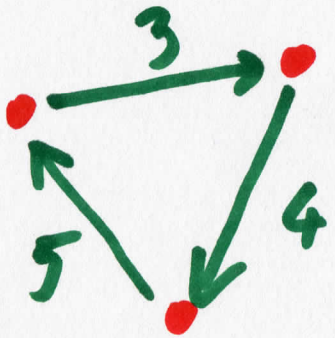
G: DAG Élekben: $c(e) \geq 0$.

G: komplex munkafolyamat. csúcsok: munkafázisok.

$c(e) = c(uv)$: u kezdése után c idővel (vagy később) lehet elkezdeni v-t.

Kérdés: optimális ütemezés: min. teljes idő.

Trivi: ha G-ben van irányított kör: nincs jó ütemezés.



Feltehetjük: Egy forrás, A, egy nyelő, B.
Különben: extra pontok, A, B, $\forall v$ Av él, $c=0$
vB él, $c=0$.

G DAG → van top. sorrend:

$v_0=A\ v_1 \dots v_n\ B=v_{n+1}$

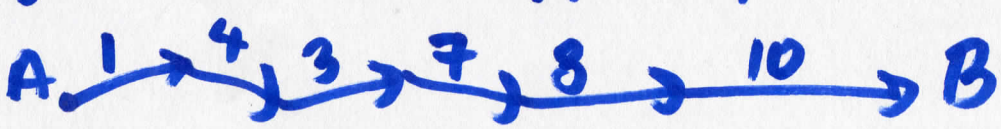
Algoritmus: $k(A)=0$ (kezdési idő)

$i=1,2 \dots n+1$: v_i ütemezése:

$$k(v_i) = \max_{j<i} \left(k(v_j) + c(v_j, v_i) \right).$$

közben megjegyezzük: melyik j adta a maximumot.
→ kritikus út (bizonyítek)

$u_0=A\ u_1 \dots u_m=B$, $k(B) = c(u_0,u_1) + c(u_1,u_2) + \dots c(u_{m-1},u_m)$



$k(B) = 33$