Miskolci Egyetem
Gépészmérnöki és Informatikai Kar
Informatikai Intézet
Alkalmazott Informatikai Intézeti
Tanszék

ET Erőforrás tervezés Resource Planning

2019/20 2. félév

10. rész

Dr. Kulcsár Gyula
egyetemi docens

Projektütemezés

Tartalom

1. A projektütemezés alapjai.
2. Erőforrás-korlát nélküli projektütemezési feladatok megoldása CPM-módszerrel.
3. Erőforrás-korlátos projektütemezési feladatok modellezése és heurisztikus megoldása.

Felhasznált irodalom

- Michael L. Pinedo:
  Planning and Scheduling in Manufacturing and Services.
  Springer, (2nd ed.), 2009

A projektütemezés alapjai

Projektütemezés

- Projekt:
  - Egy nagy, összetett, általában egyedi igény alapján előállítandó termék vagy nyújtandó szolgáltatás előállítására/teljesítésére irányú törekvés, amely általában nagyszámú komponens feladat/aktivitás végrehajtását igényli.
- Projektütemezés:
  - Projekt(ek) időbeli végrehajtásának megtervezése úgy, hogy a megfogalmazott célok teljesüljenek figyelembe véve az előírt korlátozásokat.

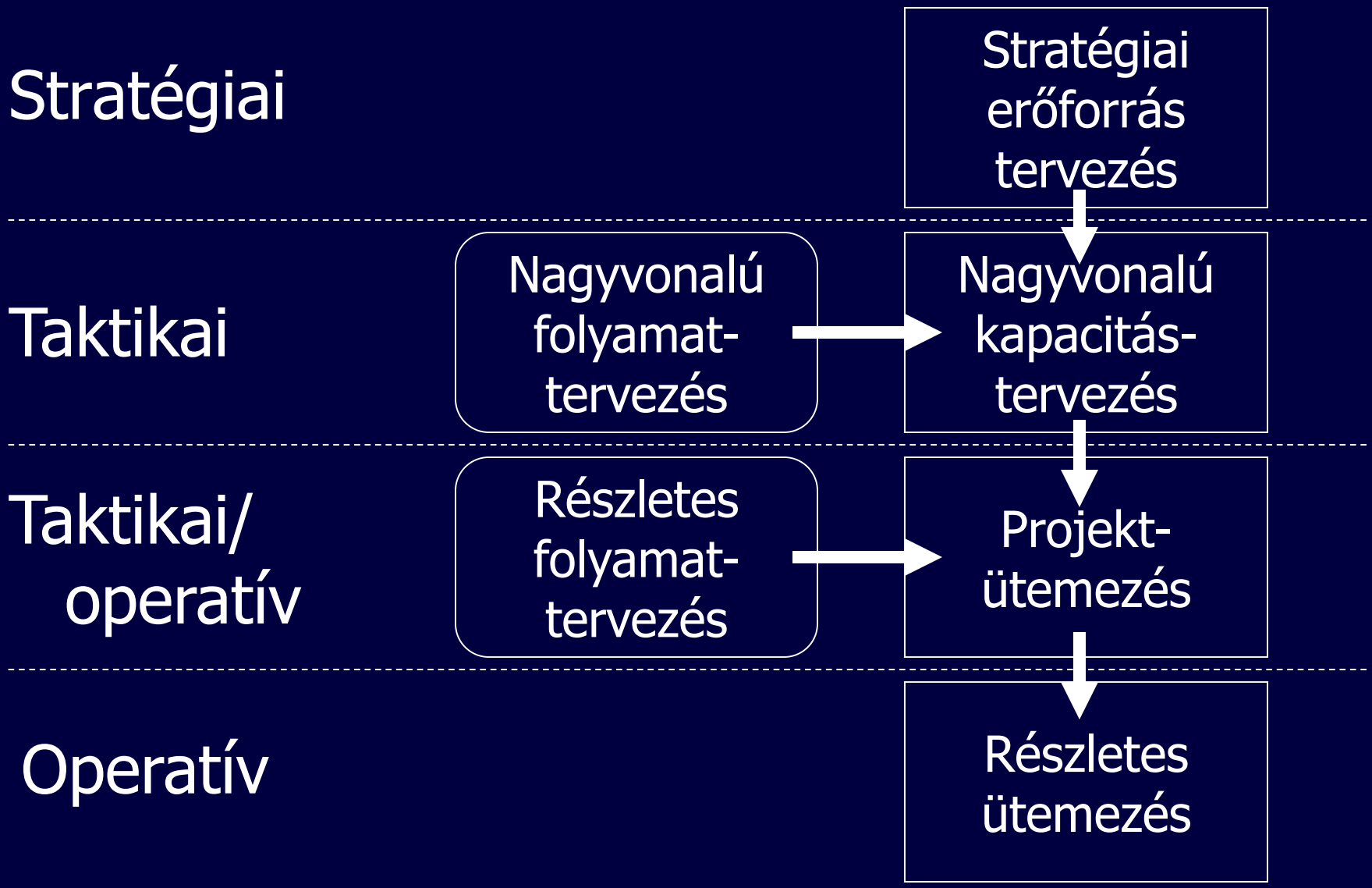
Projektütemezés jellemzői

- Cél:
  - Egy- vagy többcélú optimalizálás,
  - amelyben sokféle szempont szerepelhet (pl. minőség, idő, költség, felhasználói elégedettség stb.).
- Feladatok/aktivitások hálózata alakul ki (pl. megelőzési relációk alapján).
- Korlátozottan/korlátlanul rendelkezésre álló erőforrásokat kell figyelembe venni.

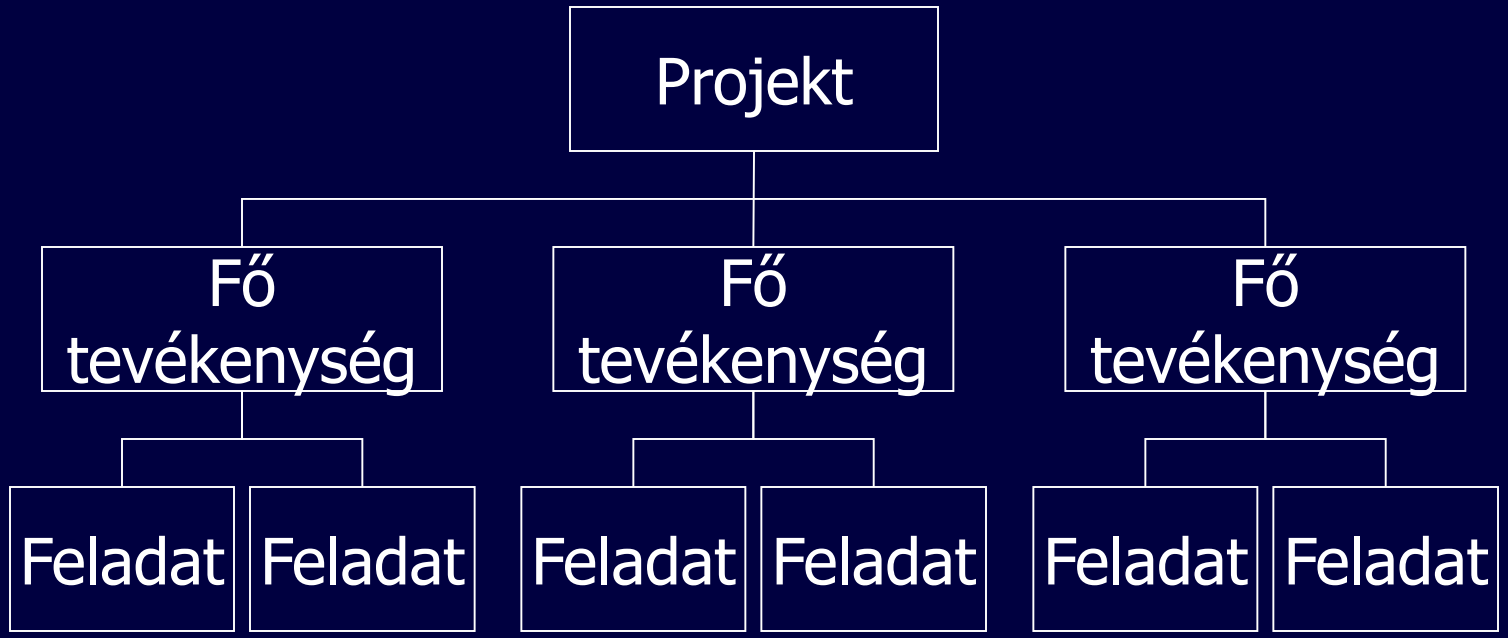
Projekt példák

- Termelés
- Tervezés
- Kutatás/fejlesztés
- Menedzsment
- Építés
- Karbantartás, fenntartás
- Implementálás, telepítés
- stb.

Hierarchikus tervezés



Egy projekt struktúrája



RCCP

Projekt-ütemezés

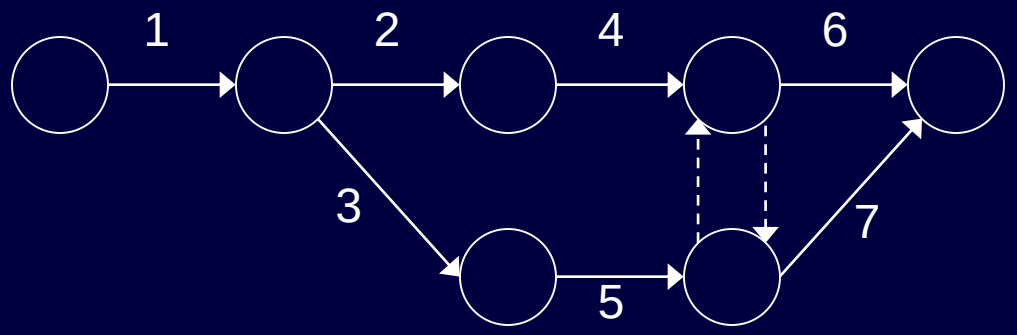
A projektütemezés alapjai

- Projekt/projektek reprezentálása (precedencia gráfok)
- Modellek és megoldási módszerek
  - Kritikus útvonal módszer (egyszerű) (Critical Path Method, CPM)
  - Erőforrás-korlátos projektütemezés (bonyolult) (Resource-Constrained Project Scheduling, RCPS)
    - Prioritás/szabályalapú megoldási módszerek
    - Tudás-intenzív megoldási módszerek
  - Kiterjesztett modellek és módszerek (összetett)

Projekt ábrázolása

Feladat	Végrehajtási idő [időegység]	Megelőző feladat(ok)
1	2	-
2	3	1
3	1	1
4	4	2
5	2	3
6	1	4, 5
7	3	4, 5

„job on arc” reprezentáció:



Projekt reprezentálása precedencia gráffal

Feladat	Végrehajtási idő [időegység]	Megelőző feladat(ok)
1	2	-
2	3	1
3	1	1
4	4	2
5	2	3
6	1	4, 5
7	3	4, 5

„job on node” ábrázolás

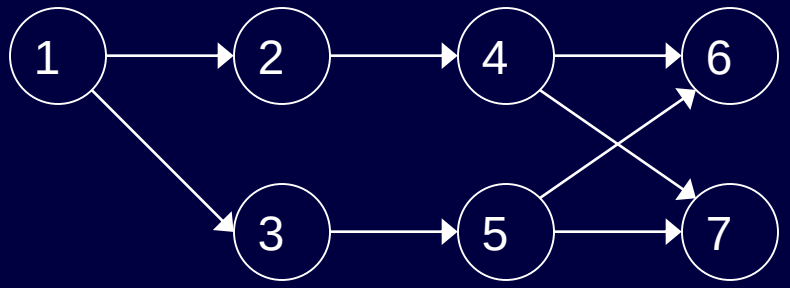
Csomópont: feladat

A csomópontok számozottak.

Irányított él: kötelező sorrendiség

Nincs irányított körút.

Nincs redundáns él.



Erőforrás-korlát nélküli projektütemezési feladatok megoldása CPM-módszerrel

Projektütemezési feladat erőforráskorlátok nélkül

- Feltételezzük, hogy:
  - korlátlan erőforrások állnak rendelkezésre párhuzamosan,
  - adott *n* feladat megelőzési relációkkal,
  - minden egyes feladat p_j végrehajtási idejét ismertjük.
- Az ütemezés célja:
  - a projekt befejezési időpontjának (makespan) minimalizálása.

Jelölések

- A j feladat:
  - végrehajtási ideje: p_j
  - legkorábbi lehetséges kezdési időpontja: S_j'
  - Legkésőbbi megengedett kezdési időpontja: S_j''
  - legkorábbi lehetséges befejezési időpontja: C_j'
  - legkésőbbi megengedett befejezési időpontja: C_j''
  - időtartaléka: $slack_j = C_j'' - p_j - S_j'$
- Kritikus feladat: nincs tartaléka $slack_j = 0$
- Kritikus útvonal: kritikus feladatok láncolata.

Kritikus útvonal módszer (Critical Path Method, CPM)

- A CPM módszer két algoritmusból áll:
  - Előre haladó eljárás („Forward procedure”)
  - Visszafelé haladó eljárás („Backward procedure”)

Kritikus útvonal módszer (Critical Path Method, CPM)

- Előre haladó eljárás (Forward procedure):
  - Kezdeti időpontból indul, a precedencia gráfon végighaladva az irányított élek mentén kiszámítja minden feladat esetében a legkorábbi megengedett indítási és befejezési időpontot.
  - A legkésőbb befejeződő feladat adja meg a projekt befejezési időpontját (C_{max}).

Előre haladó eljárás (Forward procedure)

1. lépés:

Legyen $t = t_s$ (pl. $t_s = 0$ az indítás referencia időpontja).

A megelőző feladattal nem rendelkező minden egyes j feladat esetében legyen $S_j' = t$ és $C_j' = t + p_j$.

2. lépés:

A megelőző feladattal rendelkező minden egyes j feladat esetében legyen induktív módon:

$$S_j' = \max_{all\ k \to j} C_k' \quad \text{és } C_j' = S_j' + p_j.$$

3. lépés:

Az elérhető legkorábbi projekt-befejezési időpont:

$$C_{max} = max\left\{C_1', C_2', ..., C_n'\right\}$$

Kritikus útvonal módszer (Critical Path Method, CPM)

- Visszafelé haladó eljárás (Backward procedure):
  - A projekt befejezési időpontjából indul, a precedencia gráfon az irányított élek mentén visszafelé haladva kiszámítja minden feladat esetében a legkésőbbi megengedett befejezési és indítási időpontot tekintettel arra, hogy a projektbefejezési határidő még tartható legyen.

Visszafelé haladó eljárás (Backward procedure)

1. lépés:

Legyen $t = C_{max}$

A rákövetkező feladattal nem rendelkező minden egyes j feladat esetében legyen S_k

$C_j'' = C_{max}$ és $S_j'' = C_{max} - p_j$.

2. lépés:

A rákövetkező feladattal rendelkező minden egyes j feladat esetében legyen

$C_j'' = \min_{j \to all\ k} S_k''$ és $S_j'' = C_j'' - p_j$.

3. lépés:

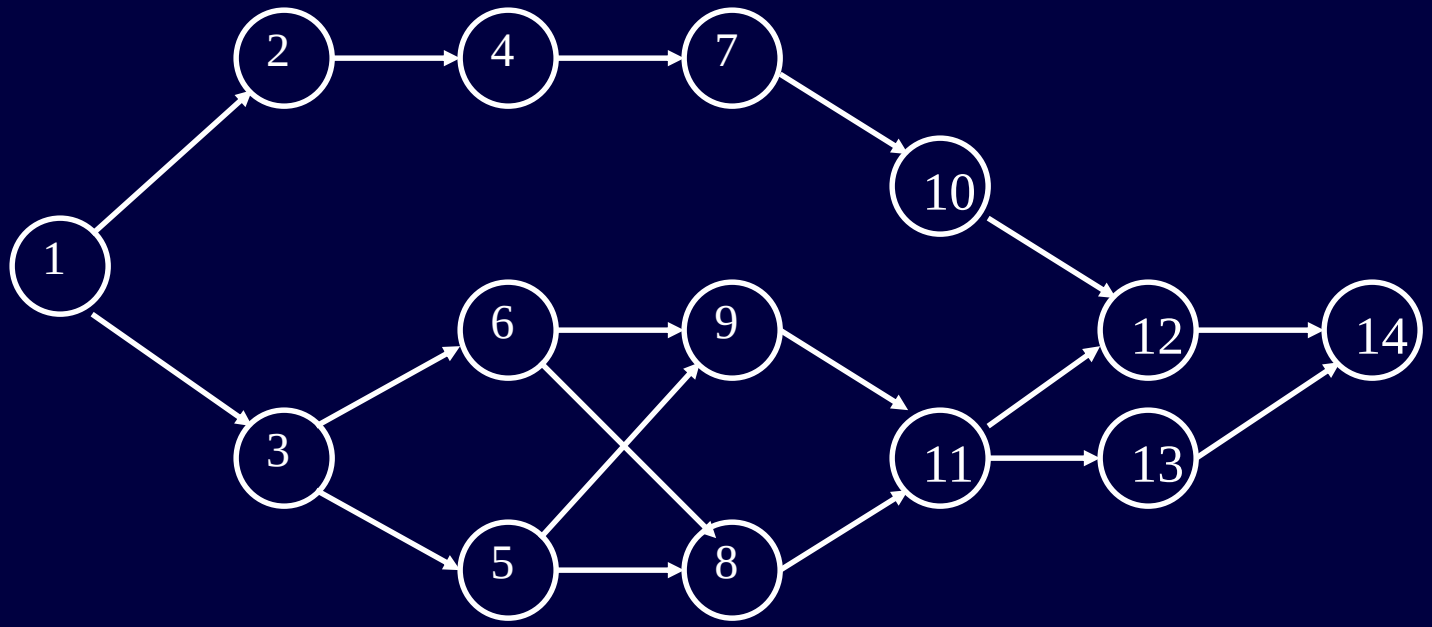
Ellenőrizzük, hogy $t_s = min\{ S_1'', ..., S_n'' \}$.

Magyarázat

- A „*forward procedure”* megadja az S_j' megengedett legkorábbi indítási időpontját minden feladatnak.
- A „*backward procedure”* megadja az S_j'' megengedett legkésőbbi indítási időpontját minden feladatnak.
- Ha ezek azonosak, akkor a feladat kritikus.
- Ha ezek különbözőek, akkor a feladatnak van időtartaléka (**slack)**.
- Kritikus útvonal (**critical path):**
  kritikus feladatok láncolata, amely a t_s kezdési időponttól a C_{max} befejezési időpontig vezet.
- Kritikus útvonalból egyszerre több is lehet, ezek akár részben fedhetik is egymást.

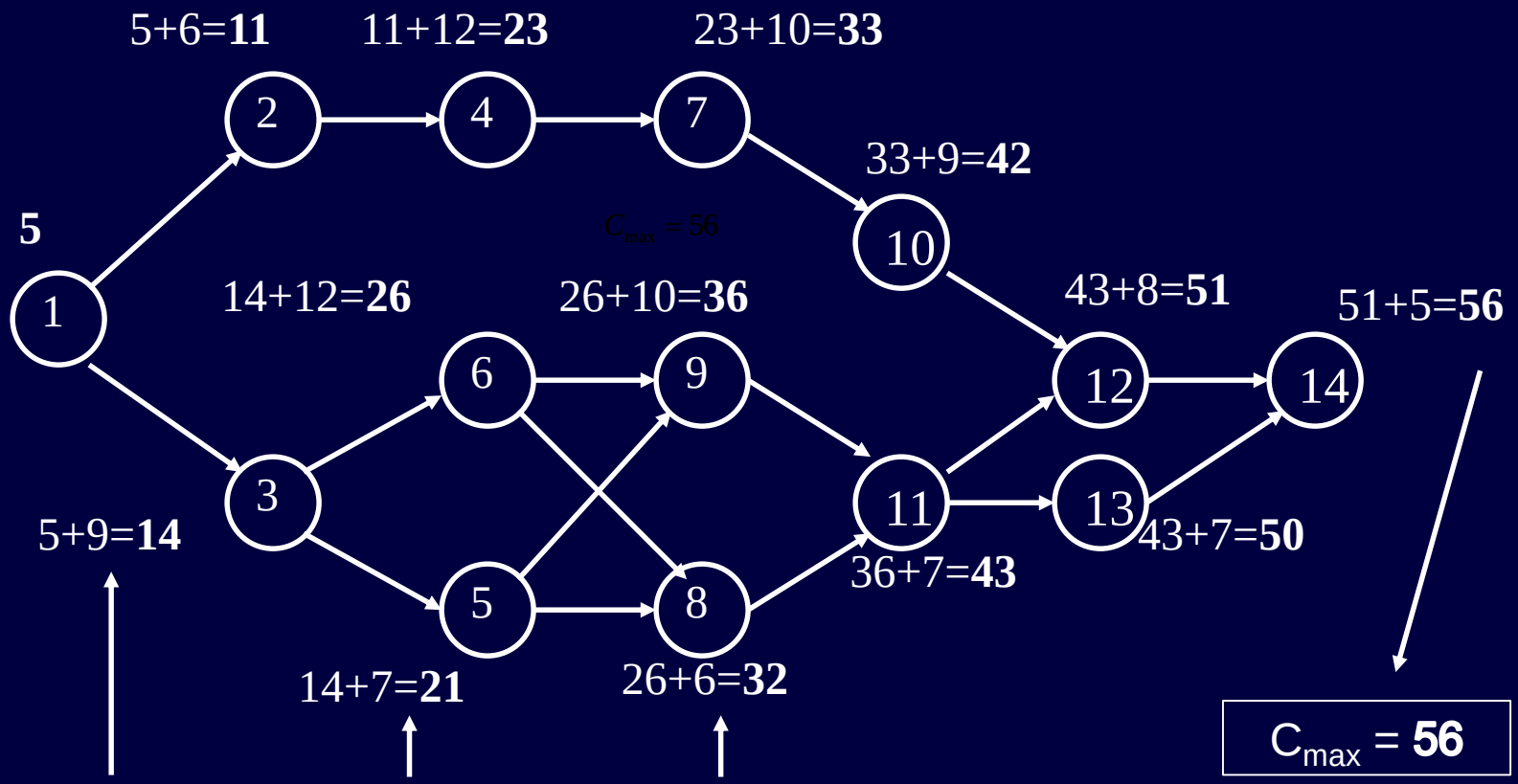
CPM példa 1

j	1	2	3	4	5	6	7	8	9	10	11	12	13	14
p_j	5	6	9	12	7	12	10	6	10	9	7	8	7	5



Előre haladó eljárás

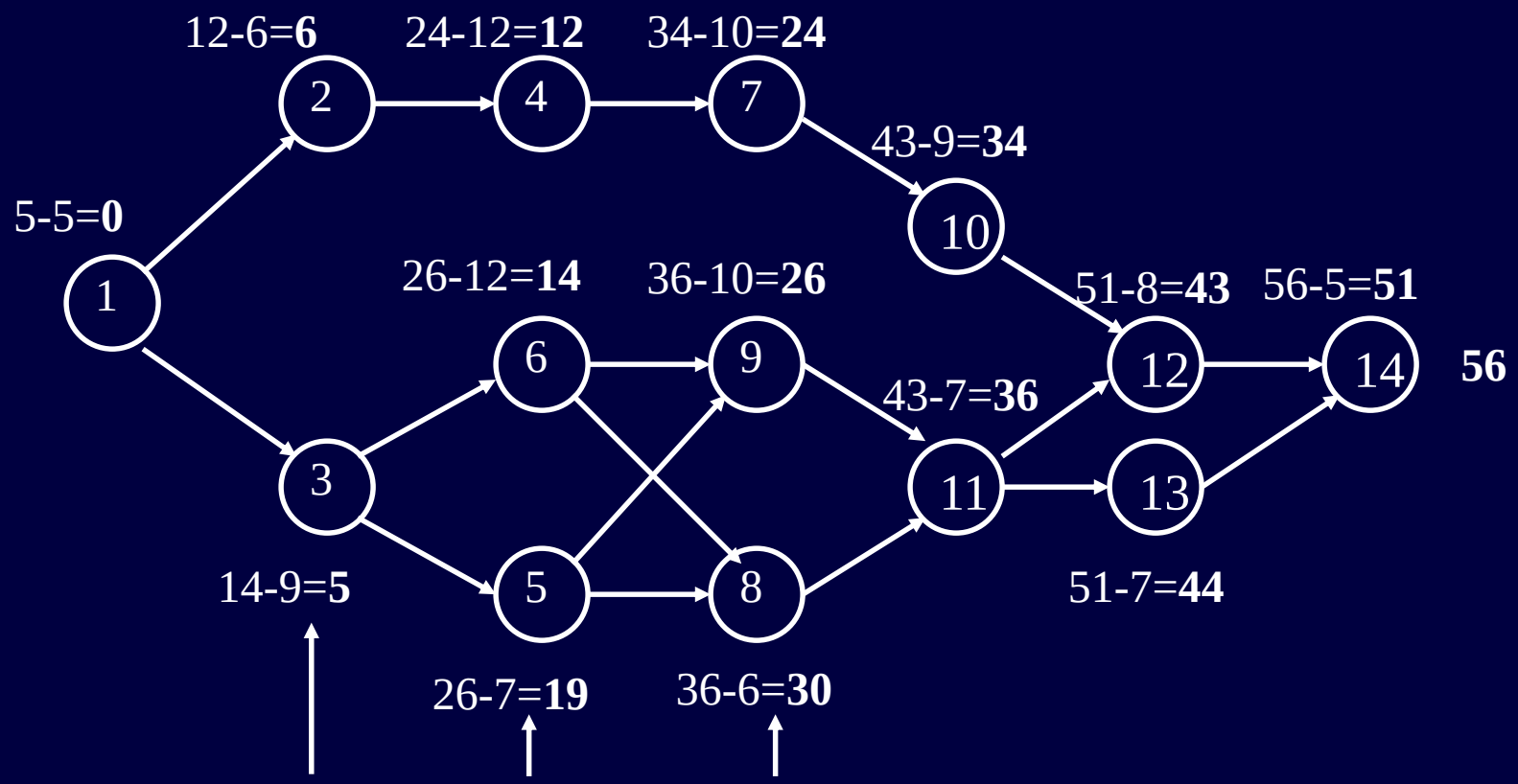
j	1	2	3	4	5	6	7	8	9	10	11	12	13	14
p_j	5	6	9	12	7	12	10	6	10	9	7	8	7	5



A feladatok legkorábbi befejezési időpontjainak számítása

Visszafelé haladó eljárás

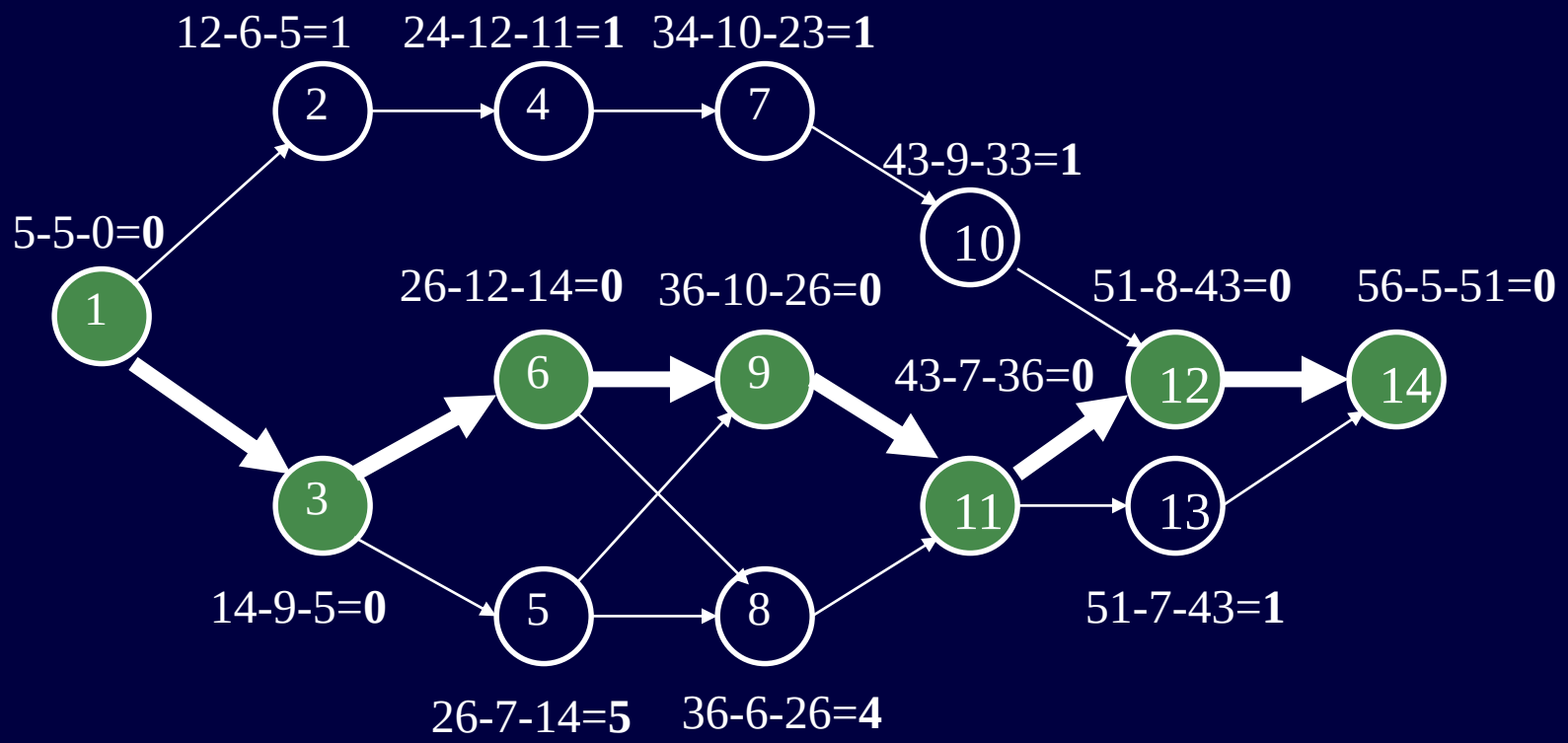
j	1	2	3	4	5	6	7	8	9	10	11	12	13	14
p_j	5	6	9	12	7	12	10	6	10	9	7	8	7	5



A feladatok legkésőbbi indítási időpontjainak számítása

Kritikus útvonal

j	1	2	3	4	5	6	7	8	9	10	11	12	13	14
p_j	5	6	9	12	7	12	10	6	10	9	7	8	7	5

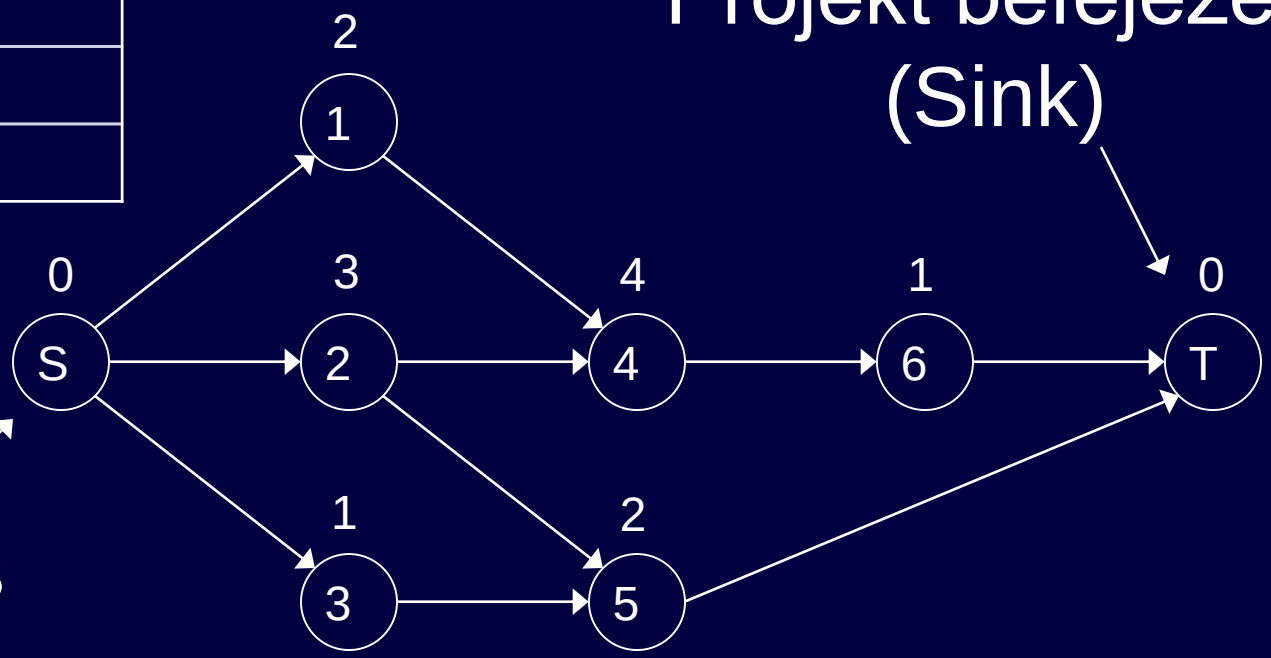


$$slack_j = C_j'' - p_j - S_j'$$

CPM példa 2

Feladat Műveleti idő Megelőző feladat(ok)

Job	p(j)	Predecessors
1	2	-
2	3	-
3	1	-
4	4	1,2
5	2	2,3
6	1	4

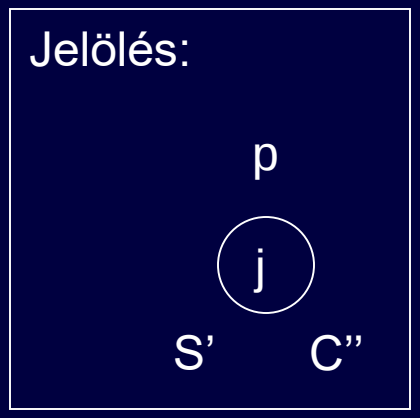


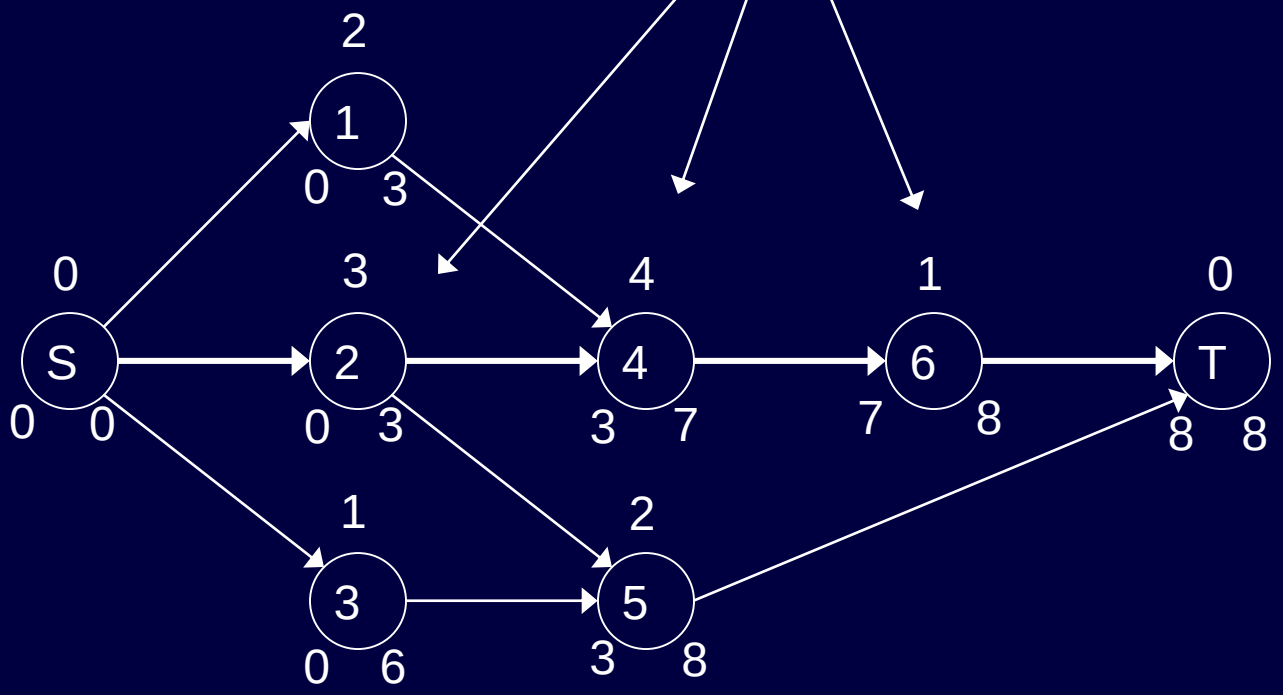
CPM példa 2 (folyt.)

Job	p(j)	Predecessors	S'	C''
1	2	-	0	3
2	3	-	0	3
3	1	-	0	6
4	4	1,2	3	7
5	2	2,3	3	8
6	1	4	7	8

Kritikus feladat (Critical job):

S'+ p = C' = C'' = S''+ p





Erőforrás-korlátos projektütemezési feladatok modellezése és heurisztikus megoldása

Erőforrás-korlátos projektütemezés

Resource Constrained Project Scheduling (RCPSP)

RCPSP

- n munka (job) $j=1,\ldots,n$
- N erőforrás $k=1,\ldots,N$
- R_k: a k erőforrás korlátja (rendelkezésre állás)
- p_j: a j munka (job) végrehajtási ideje
- R_{kj}: a j munka (job) igénye az k erőforrásból
- P_j: a j munkát (job-ot) közvetlenül megelőző munkák halmaza (predecessors).

RCPSP

- Cél:
  - a projekt befejezési időpontjának (C_{max}) minimalizálása

$$C_{max} = \max_j C_j'$$

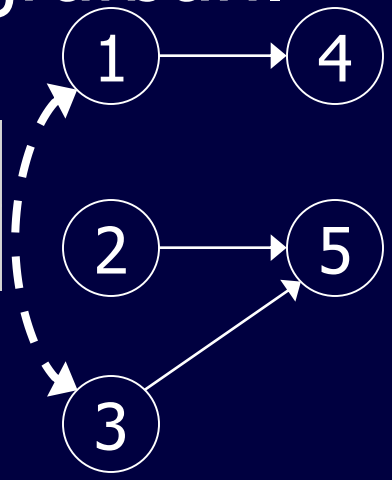
- Korlátozások:
  - a T=0 időpont előtt egyetlen munka sem indíthtó
  - a precedencia korlátozásokat be kell tartani
  - az erőforrások kapacitása véges

Projektütemezés

- Erőforrás-korlátok nélkül viszonylag egyszerű.
- Erőforrás-korlátokkal nagyon bonyolult: amikor a korlátozottan rendelkezésre álló erőforrások miatt bizonyos munkák (jobs) nem hajthatók végre párhuzamosan $\Rightarrow$ ***diszjunktív élek*** jelennek meg a gráfban.

Jobs	**1**	**2**	**3**	**4**	**5**
p(j)	8	4	6	4	4
R(1,j)	2	1	3	1	2
R(2,j)	3	0	4	0	3

Erőforrás	R1	R2
Korlát	4	8



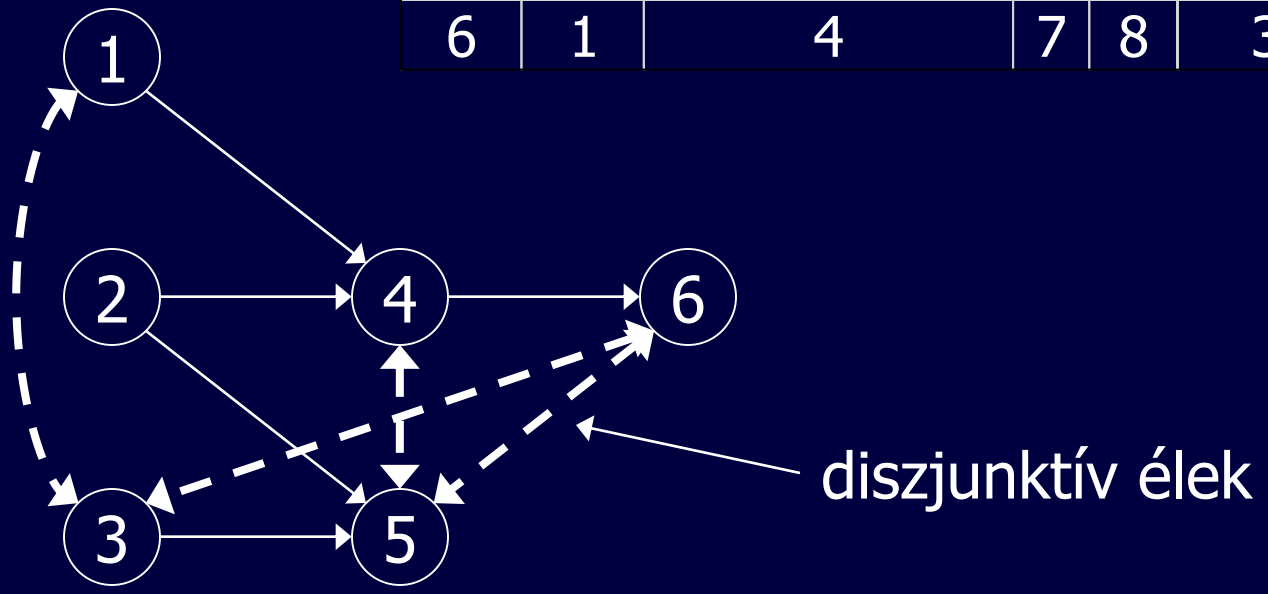
Diszjunktív élek

Tegyük fel, hogy $R_1=4$.

Job	p(j)	Predecessors	S'	C''	R(1,j)
1	2	-	0	3	3
2	3	-	0	3	1
3	1	-	0	6	2
4	4	1,2	3	7	2
5	2	2,3	3	8	3
6	1	4	7	8	3

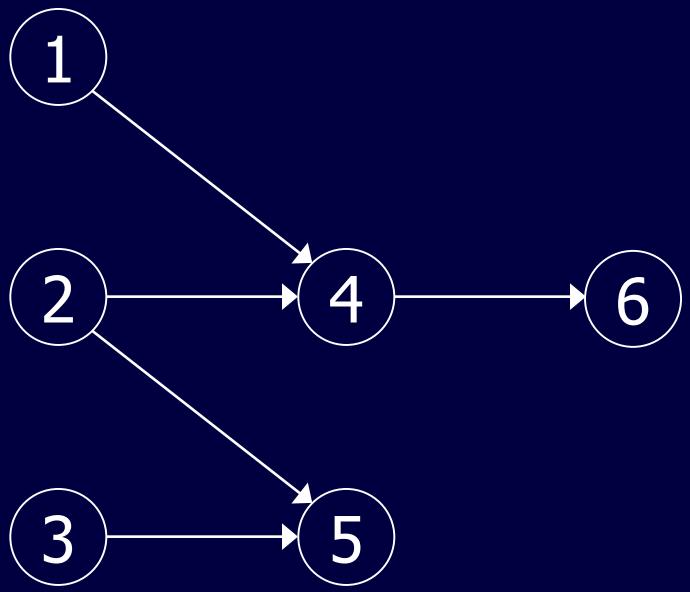
A következő munkák nem hajtók végre párhuzamosan:

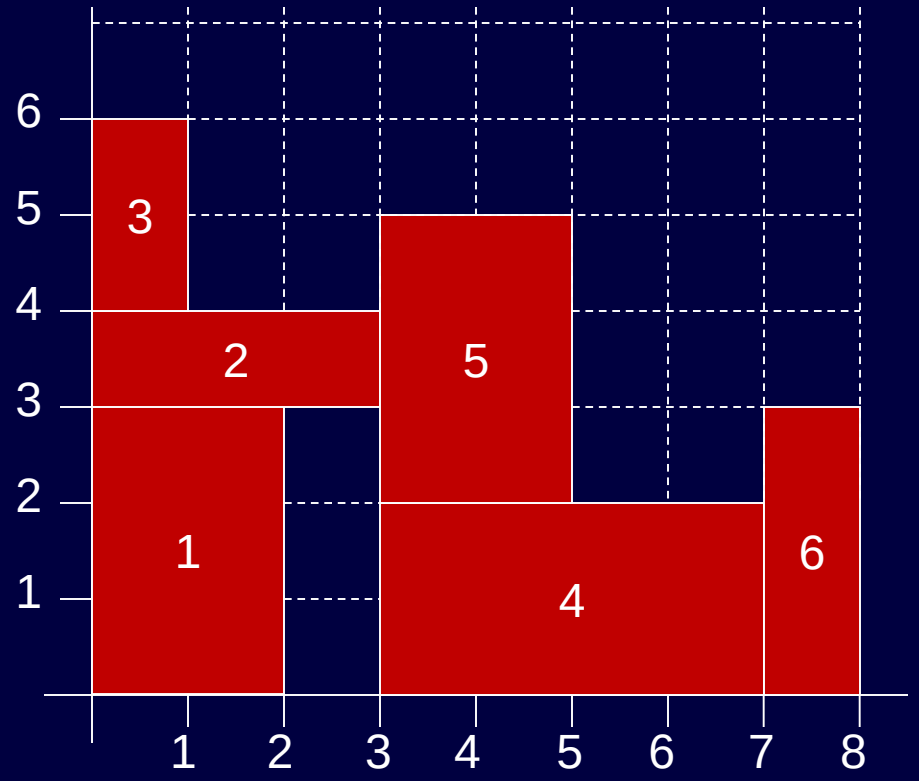
- 1 & 3
- 3 & 6
- 4 & 5
- 5 & 6



RCPSP (példa)

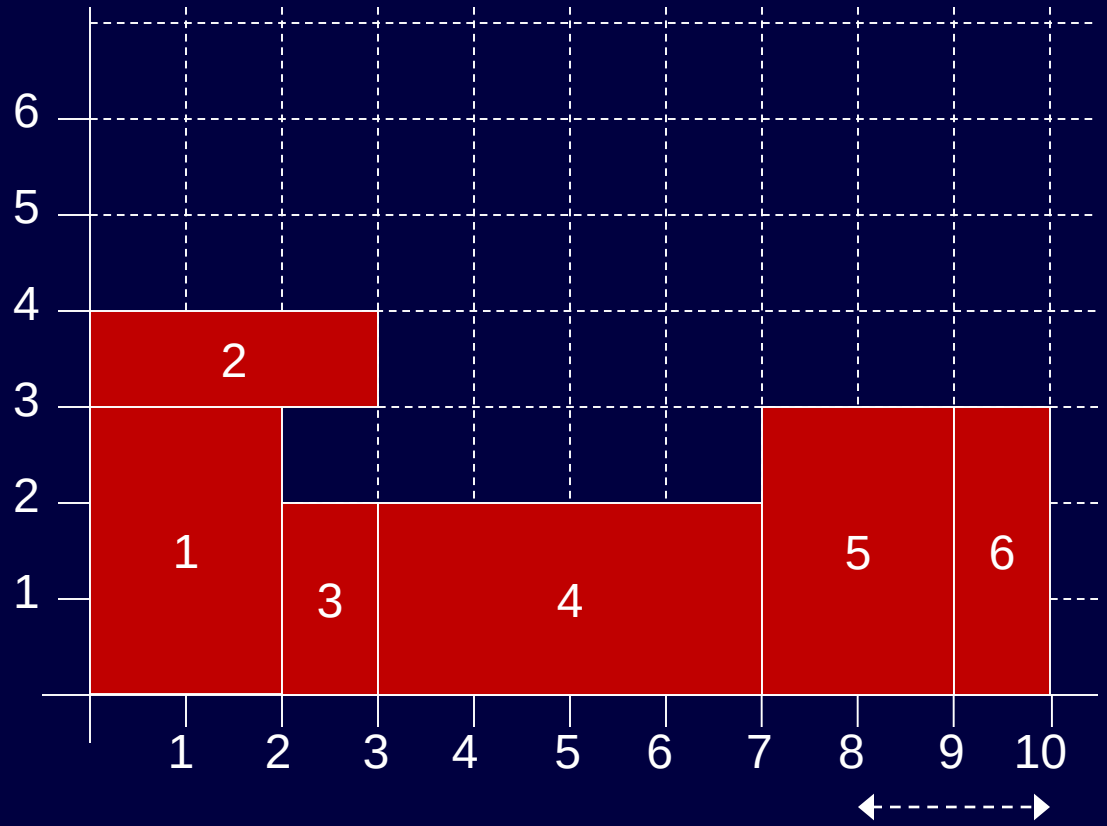
Job	p(j)	Predecessors	S'	C''	R(1,j)
1	2	-	0	3	3
2	3	-	0	3	1
3	1	-	0	6	2
4	4	1,2	3	7	2
5	2	2,3	3	8	3
6	1	4	7	8	3





RCPSP (példa folyt.)

- Tételezzük fel, hogy $R_1 = 4$, ekkor:

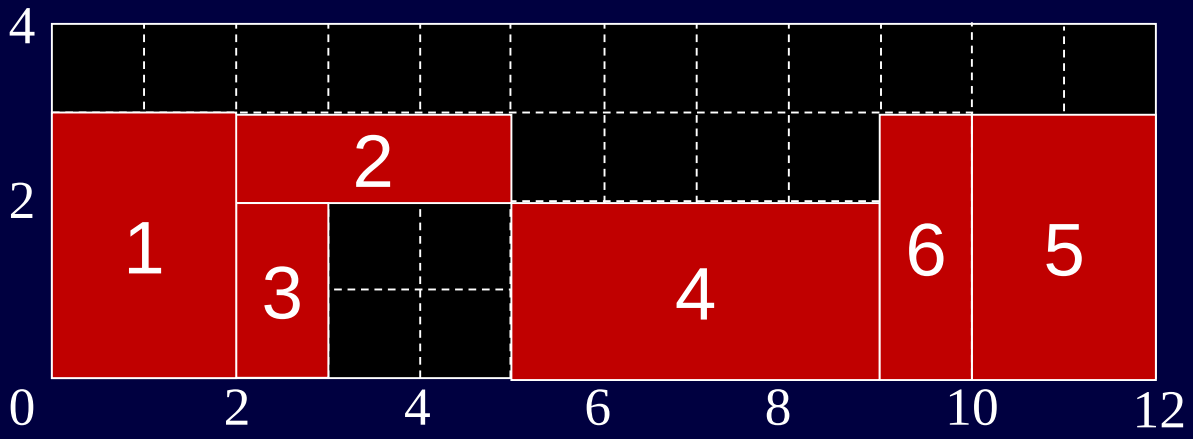


C_{max} nő 2 időegységgel!

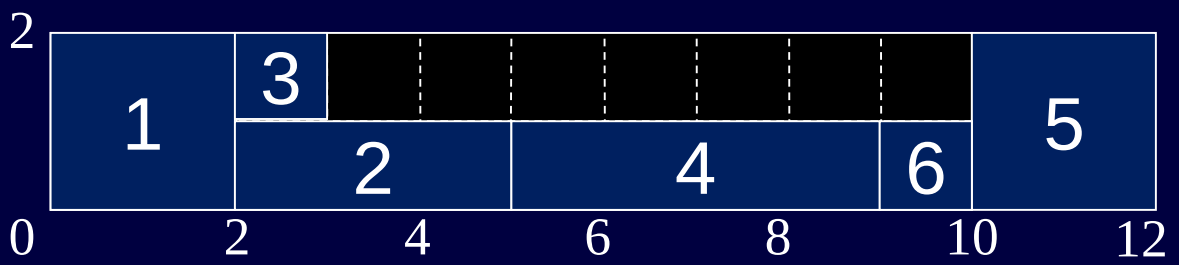
RCPSP (Példa 2)

Job	p(j)	P(j)	S'	C''	R(1,j)	R(2,j)
1	2	-	0	3	3	2
2	3	-	0	3	1	1
3	1	-	0	6	2	1
4	4	1,2	3	7	2	1
5	2	2,3	3	8	3	2
6	1	4	7	8	3	1

$R_1 = 4$



$R_2 = 2$



Prioritási szabály alapú ütemezés (Priority-rule-based scheduling)

- Generálási sémák (Generation Scheme) ←
  - Soros (serial)
  - Párhuzamos (parallel)
- Prioritási szabályok (Priority rule)
  - Legkésőbbi befejezési időpont (latest finish time)
  - Minimális időtartalék (minimum slack)

Soros ütemezési módszer (Serial Scheduling Method)

Adott *n* számú munka (job).

Az algoritmus közben a következő halmazokat használjuk:

- a kész munkák halmaza: beütemezett munkák
- a döntési halmaz: indítható munkák (az „előfeltételeik” be vannak ütemezve)
- a fennmaradó munkák halmaza: a többi munka

Az eljárás:

1. Készítsünk egy üres ütemtervet és határozzuk meg a döntési halmazt.
2. A döntési halmazból vegyük ki a legnagyobb prioritású munkát, és ütemezzük a lehető legkorábbi kezdéssel.
3. Frissítsük a halmazokat.
4. Ha a döntési halmaz nem üres, akkor folytassuk a 2. lépéssel, egyébként vége.

Soros ütemezési módszer
Példa (#1)

Job	p(j)	P(j)	R(1,j)	v(j) (priority)
1	2	-	1	2
2	3	-	1	1
3	3	1	2	2

Döntési halmaz

Soros ütemezési módszer
Példa (#2)

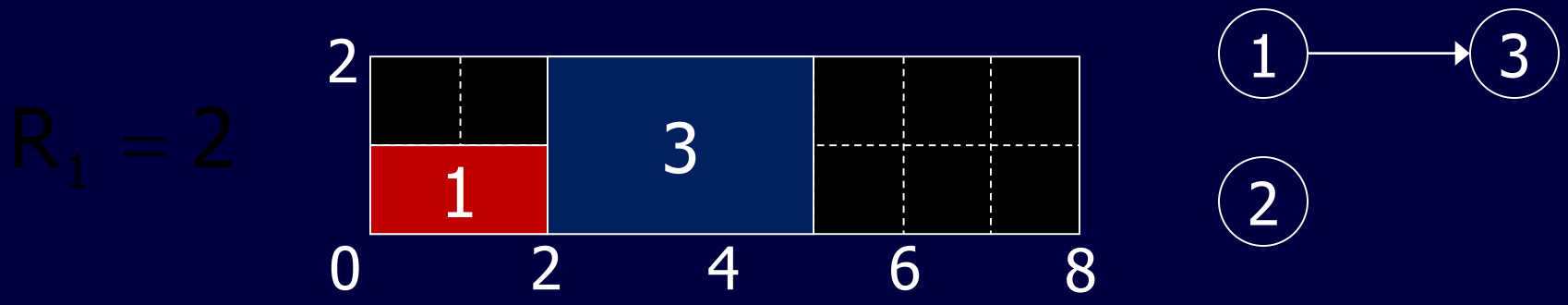
Job	p(j)	P(j)	R(1,j)	v(j) (priority)
1	2	-	1	2
2	3	-	1	1
3	3	1	2	2

Döntési halmaz

Soros ütemezési módszer
Példa (#3)

Job	p(j)	P(j)	R(1,j)	v(j) (priority)
1	2	-	1	2
2	3	-	1	1
3	3	1	2	2

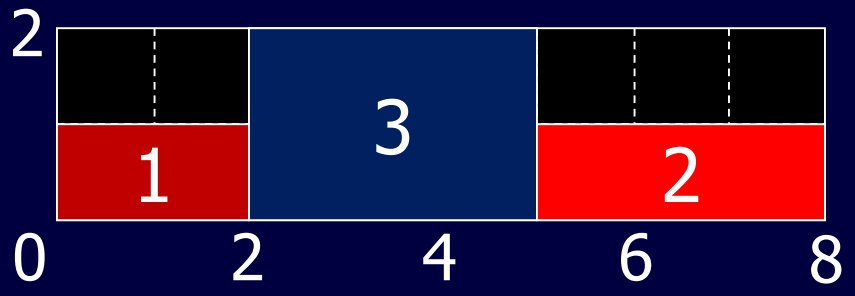
Döntési halmaz

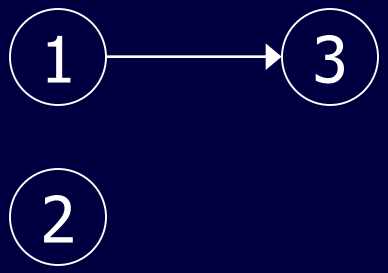


Soros ütemezési módszer
Példa (#4)

Job	p(j)	P(j)	R(1,j)	v(j) (priority)
1	2	-	1	2
2	3	-	1	1
3	3	1	2	2

$R_1 = 2$





Párhuzamos ütemezési módszer (Parallel Scheduling Method)

1. Készítsünk egy üres ütemtervet.
2. Legyen *T* az a legkorábbi időpont, amikor egy ütemezetlen munka indítható (az elő-feltételei teljesültek). Válogassuk ki azokat a munkákat, melyek a *T* időpontban indíthatók. Jelölje ezen munkák halmazát *D*.
3. Ha a *D* halmaz nem üres, akkor válasszuk ki belőle a legnagyobb prioritású munkát. Ütemezzük a kiválasztott munkát a *T* indítási időpontra. Folytassuk a 2. lépéssel.
4. Ha a *D* halmaz üres, akkor vége.

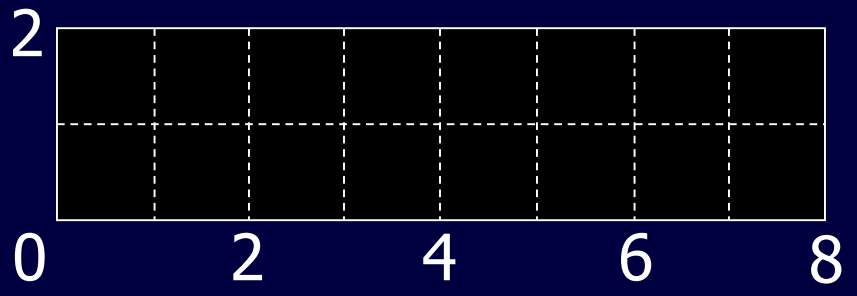
Párhuzamos ütemezési módszer Példa (#1)

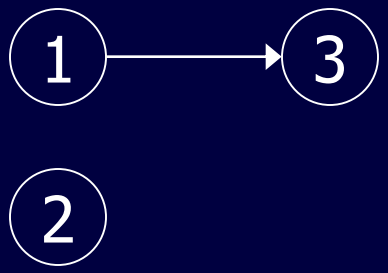
Job	p(j)	P(j)	R(1,j)	v(j) (priority)
1	2	-	1	2
2	3	-	1	1
3	3	1	2	2

D

$T = 0$

$R_1 = 2$





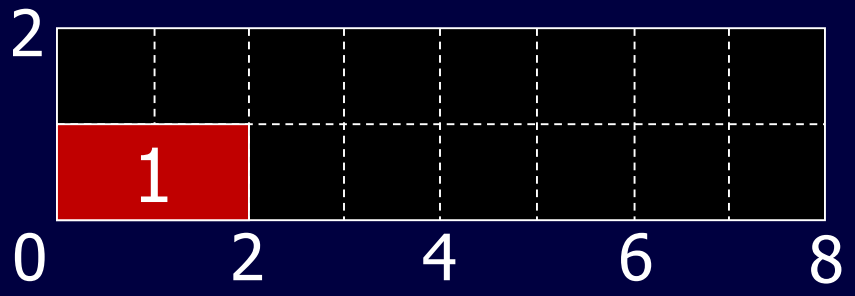
Párhuzamos ütemezési módszer Példa (#2)

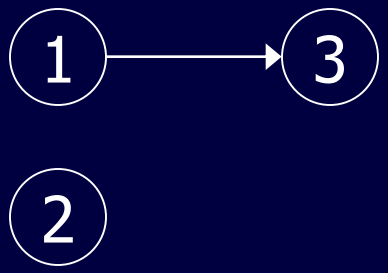
Job	p(j)	P(j)	R(1,j)	v(j) (priority)
1	2	-	1	2
2	3	-	1	1
3	3	1	2	2

D

$T = 0$

$R_1 = 2$





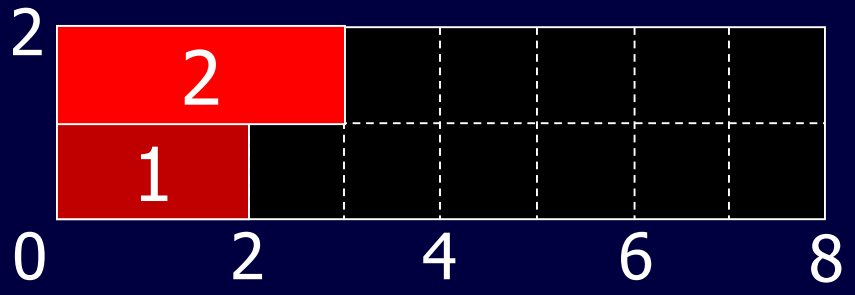
Párhuzamos ütemezési módszer Példa (#3)

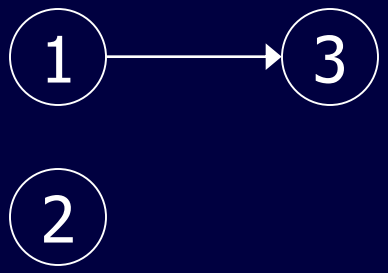
Job	p(j)	P(j)	R(1,j)	v(j) (priority)
1	2	-	1	2
2	3	-	1	1
3	3	1	2	2

D

T = 3

$R_1 = 2$

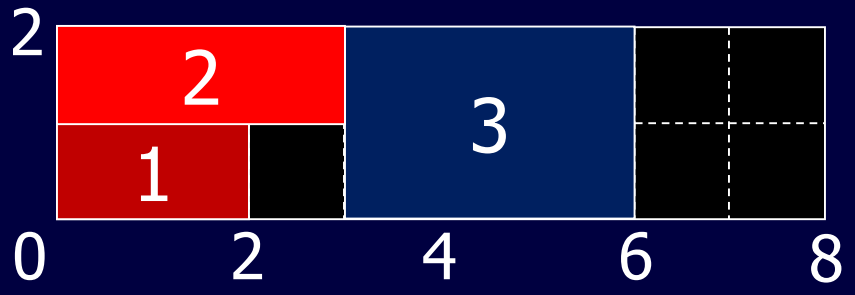


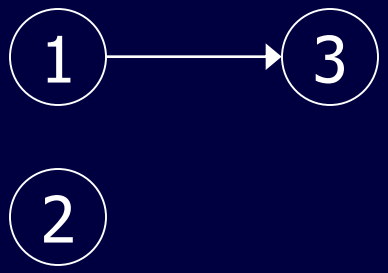


Párhuzamos ütemezési módszer Példa (#4)

Job	p(j)	P(j)	R(1,j)	v(j) (priority)
1	2	-	1	2
2	3	-	1	1
3	3	1	2	2

$R_1 = 2$





Prioritási szabály alapú ütemezés (Priority-rule-based scheduling)

- Generálási sémák (Generation scheme)
  - Soros (serial)
  - Párhuzamos (parallel)
- Prioritási szabályok (Priority rule) ←
  - Legkésőbbi befejezési időpont (latest finish time)
  - Minimális időtartalék (minimum slack)

Prioritási szabályok

- Legkésőbbi befejezési időpont
  Latest finish time (LFT): $v_j = - C''_j$

- Minimális időtartalék
  Minimum slack (MS): $v_j = - (C''_j - p_j - t^*)$

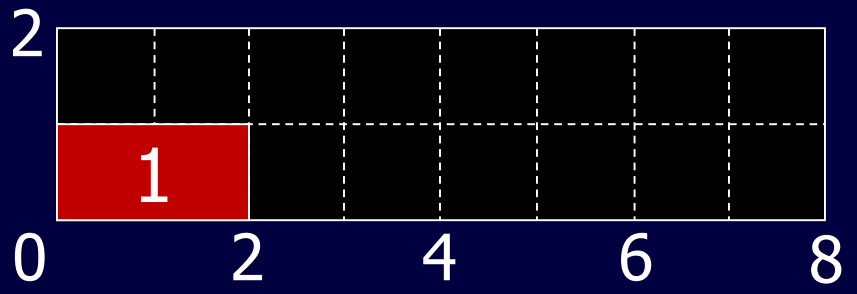
az aktuális legkorábbi indítási időpont

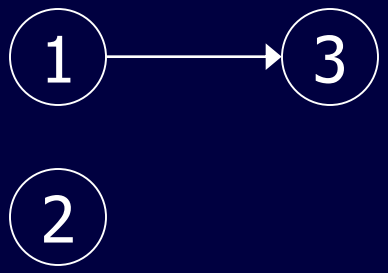
MS prioritási szabály
soros ütemezési sémával (#1)

Job	p(j)	P(j)	R(1,j)	S'(j)	C''(j)	v(j) (priority)
1	2	-	1	0	2	0
2	3	-	1	0	5	-2
3	3	1	2	2	5	0

$$v_j = -(C''_j - p_j - t^*)$$

$R_1 = 2$



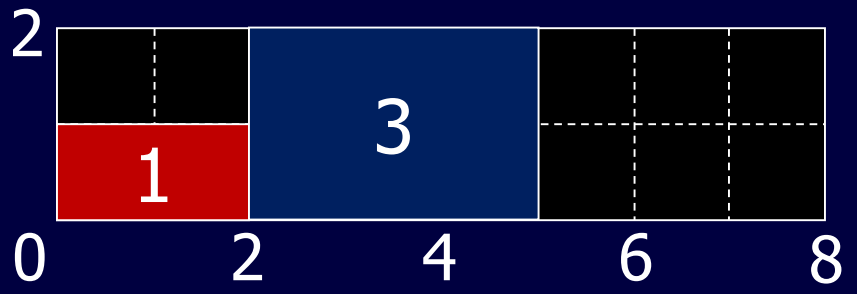


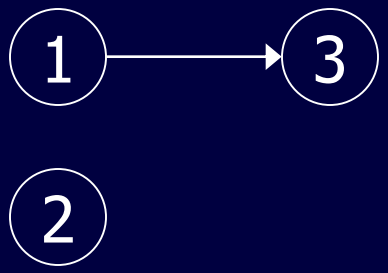
MS prioritási szabály
soros ütemezési sémával (#2)

Job	p(j)	P(j)	R(1,j)	S'(j)	C''(j)	v(j) (priority)
1	2	-	1	0	2	
2	3	-	1	0	5	-2
3	3	1	2	2	5	0

$$v_j = -(C''_j - p_j - t^*)$$

$R_1 = 2$



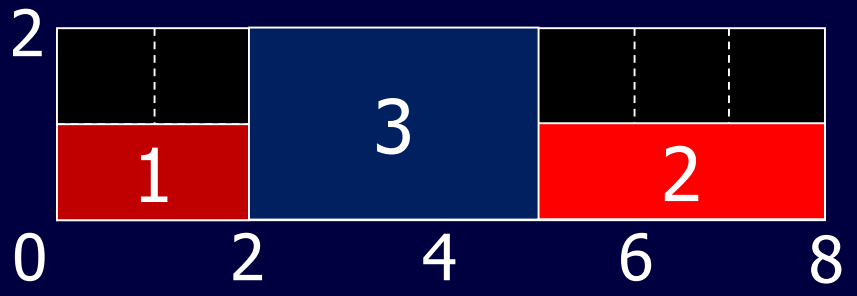


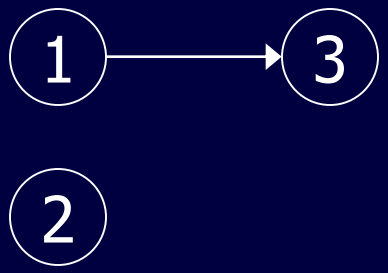
MS prioritási szabály
soros ütemezési sémával (#3)

Job	p(j)	P(j)	R(1,j)	S'(j)	C''(j)	v(j) (priority)
1	2	-	1	0	2	
2	3	-	1	5	5	3
3	3	1	2	2	5	

$$v_j = -(C''_j - p_j - t^*)$$

$R_1 = 2$





Összefoglalás

- A projektütemezés alapjai
- Erőforrás korlát nélküli probléma
  - CPM módszer
- RCPS problema
  - Generálási sémák és prioritási szabályok
  - Keresési algoritmusok (következő előadás)

Köszönöm a figyelmet!

Az előadásvázlat elérhető az alábbi webcímen:

http://ait.iit.uni-miskolc.hu/~kulcsar/serv07.htm