Ütemezési Gyakorlatok

Szakdolgozat

Irta: Naszádos Áron

Matematika Bsc, Elemző matematikus szakirány

Témavezető:

Jordán Tibor

egyetemi tanár

Operációkutatási tanszék

Eötvös Loránd Tudományegyetem

Természettudományi Kar

2010.

Tartalomjegyzék

I. Bevezetés

Szakdolgozatomban R. Gary Parker: Deterministic Scheduling Theory című könyvében szereplő harminc különféle feladat részletes megoldását tárgyalom, ábrákkal és gráfokkal kiegészítve. Ezenkívül a dolgozatom első felében felsorolom a felhasznált algoritmusokat rövid leírásukkal együtt. A továbbiakban az i-edik munka megmunkálási idejét τ_i -vel; befejezési idejét c_i -vel; súlyát ω_i -vel; határidejét d_i -vel; büntetésfüggvényét f_i -vel; késését L_i -vel; a késés meglétét vagy nem létét u_i -vel (így $\sum u_i$ a késések száma); T_i -vel pedig a késedelmességét, vagyis hogyha késik akkor mennyit: $T_i := max(0, c_i - d_i)$. A feladatok kitűzésében használom még az *intree*-befenyő, *tree*-fa kifejezéseket, valamint a C_{max} -legnagyobb befejezési idő; L_{max} -legnagyobb késés; T_{max} -legnagyobb késedelem célfüggvény jelöléseket. A láda pakolási feladatban pedig a munkák tárgyaknak, a gépek pedig ládáknak felelnek, valamint a ládák rendelkeznek maximális kapacitással.

II. Egygépes ütemezési algoritmusok

SPT sorrend

Rendezzük a munkákat a megmunkálási idők szerint nem csökkenő sorrendbe.

WSPT sorrend

Rendezzük a munkákat $\frac{\tau}{\omega}$ arány szerint nem csökkenő sorrendbe.

EDD sorrend

Rendezzük a munkákat a határidők szerint nem csökkenő sorrendbe.

Greedy algoritmus

1.lépés:

Rendezzük a munkákat súlyuk szerint csökkenő sorrendbe.

2.lépés:

Ebből a listából válasszuk be a munkákat az ütemezésbe. Csak akkor nem választunk be egy munkát, ha az előző vagy ez a munka késik. Ezt a lépést addig folytatjuk, amíg el nem fogynak a munkák.

Maximális párosítás keresés speciális páros gráfban

Itt a speciálison azt értjük, hogy a pontok növekvő sorba vannak rendezve a pozíciók oldalán és csökkenő sorrendbe a munkák oldalán és ha egy munka össze van kötve egy pozícióval akkor minden előtte lévő pozícióval is össze van kötve.

1. lépés:

Rendezzük a munkákat súlyuk szerint nem növekvő sorrendbe.

2. lépés:

Építsünk páros gráfot, amiben az egyik ponthalmaz a munkák sorrendjét a másik pedig magukat a munkákat jelentik, mivel itt célszerű az első ponthalmazra úgy gondolni, mintha a munkák elvégzésének egységnyi helyére, hiszen sorrendet akkor is értelmezhetnénk, ha a munkák nem lennének egyforma hosszúak így viszont a gép munkaideje feloszthatő egységnyi adagokra, amikor az adott munkák megmunkálási ideje történik. (ezzel még csak a gráf pontjait kaptuk meg)

3. lépés:

Vegyük fel az éleket minden munkához úgy, hogy csak azzal a pozíciókkal kötjük össze, ahol nem késik.

4. lépés:

Ütemezzük a legnagyobb súlyú munkát a legkésőbbi pozícióba, ahol még nem késik, majd a következő munkát (vagyis a második legnagyobb súlyút) a lehető legtávolabbi pozícióba, ahol nem késik és ami még nem foglalt. Ezt addig folytassuk, amíg tudunk munkákat nem késő helyekre ütemezni. Ha ilyet találunk azt a munkát tegyük a későkhöz és folytassuk a következő munkával.

5. lépés:

Az algoritmusunk leáll, ha minden munkát ütemeztünk vagy próbáltunk ütemezni vagyis, ha végig mentünk az összes munkán. Ekkor a késő munkák számának súlyozott értéke minimális lesz.

Moore algoritmusa $1||\sum u_i$ **problémára**

1. lépés:

Rendezzük a munkákat EDD-be és nevezzük el ezt az ütemezést S-nek.

2. lépés:

Vegyük sorra S-ben a munkákat és nevezzük el az első késő munkát x-nek. Ekkor vegyük a leghosszabb munkát y-t és vegyük ki S-ből majd $S \leftarrow S \setminus y$, nézzük újra a munkákat S-ben már y nélkül és ha újra találunk késő munkát akkor megint kivesszük a leghosszabb munkát, és ezt ismételgetjük amíg az S-ben maradó munkák egyike sem késik. Ekkor az optimális ütemezés az S-ben maradó munkák utána pedig a kiesett munkák valamilyen sorrendben.

Lawler algoritmusa az $1||f_{max}$ **problémára**

1. lépés: legyen $S=(1,2,\ldots,n)$ és legyen $k=n$

2. lépés: legyen $\lambda \leftarrow \sum_{j \in S} \tau_j$ és határozzuk meg x-et úgy hogy $f_x(\lambda) \leqslant f_i(\lambda), \forall i \in S$

3. lépés: Ekkor rakjuk x-et k-adik pozícióba, továbbá vegyük S ből x-et $S \leftarrow S \setminus x$ és k-t csökkentsük le egyel $k \leftarrow k-1$ majd kezdjük újra a második lépéstől amíg s elemszáma 0 lesz vagyis $|S|=0$

III. Többgépes ütemezési algoritmusok

Listás ütemezés

Csináljunk egy listát a rendelkezésre álló munkákból, ebből a listából tegyük fel az első munkát az első elérhető gépre, majd a másodikat munkát a következő elérhető gépre és így tovább, amíg el nem fogynak a munkák.

Knuth-Kleitman algoritmus

Adott $k \in \mathbb{Z}$ válasszuk ki az első k leghosszabb munkát, helyezzük el optimálisan őket az m gépen, ehhez használható a listás ütemezés., majd a maradék n-k munkát szintén listás ütemezés szerint ütemezzük az m gépen.

LPT szerinti ütemezés

Rendezzük a munkákat megmunkálási idők szerint nem növekvő sorrendbe, majd erre a listára alkalmazzuk a listás ütemezést.

FFD algoritmus

Adottak a gépek és a munkák, továbbá adott a gépek maximális kapacitása is, ami legalább akkora, mint a leghosszabb munka. Rendezzük a munkákat megmunkálási idők szerint nem növekvő sorrendbe. Vegyük az első munkát az előbbi listából és helyezzük fel az első gépre, majd a következő munkánál, ha az első gép kapacitása még engedi, akkor az első gépre rakjuk fel, ha már nem fér fel, akkor a második gépre, a harmadik munkánál úgyszintén először az első gépre, ha oda nem fér fel, akkor a második gépre, ha oda se fér fel akkor a 3-dik gépre rakjuk fel.

MULTIFIT algoritmus

Lényege: Az FFD algoritmust akarjuk javítani úgy, hogy adunk egy iterációs eljárást a gépek kapacitásának csökkentése érdekében így közelítve az optimális megoldást.

0. lépés: Inicializálás

Legyen T a munkák egy halmaza és m a gépek száma, valamint m-től és T-től függő alsó és felső korlátok $\beta_L[T,m]$ és $\beta_U[T,m]$. Legyen $\beta_1(0)\leftarrow\beta_U$ valamint $\beta_2(0)\leftarrow\beta_L$ továbbá legyen t az elvégezendő iterációs lépések száma, i pedig jelölje azt, hogy hányadik iterációs lépésnél járunk Az első iterációs lépéshez kellő kezdeti alsó és felső korlátot a következő képletekkel határozzuk meg:

$$\beta_L[T,m]=max\{\sum\tau_j/m\,,\,\max_j(\tau_j)\}$$
$$\beta_U[T,m]=max\{2\sum\tau_j/m\,,\,\max_j(\tau_j)\}$$

1. lépés: Kapacitás megváltoztatása

Ha $i>t$ akkor megállunk, minden más esetben viszont $C\leftarrow(\beta_2(i-1)+\beta_1(i-1))/2$

2. lépés: Felső korlát

Jelöljük FFD[T,C] azt a gépmennyiséget amire az FFD algoritmusnak szüksége van C kapacitás mellett T munkák ütemezésére. Ha FFD[T,C] nem nagyobb mint m, akkor

$\beta_1(i)\leftarrow C\,,\beta_2(i)\leftarrow\beta_2(i-1)$, valamint növeljük meg i értékét 1-gyel.

3. lépés: Alsó korlát

Ha FFD[T,C]>m, akkor legyen $\beta_2(i) \leftarrow C$, $\beta_1(i) \leftarrow \beta_1(i-1)$ és növeljük meg i értékét 1-gyel.

4. lépés:

A végén kapott C kapacitású m gépen végezzük el az FFD algoritmust.

Bruno algoritmus

1.lépés:

Készítsünk párosgráfot, amelynek egyik csúcs halmazában a munkák vannak a másik csúcshalmazában pedig pozíció-gép kombinációk vagyis, az első pozíció az első gépen, majd az első pozíció a második gépen és így tovább egészen az utolsó munka az utolsó helyen -ig. Egy munkát egy pozíció-gép párral összekötő élen pedig a munka megmunkálási idejének a pozíció sorszámával súlyozott értéke.

2.lépés:

Keressünk maximális párosítást ebben a gráfban.

Hu algoritmusa

Hu algoritmusa tétel szerint csak olyan irányított gráfok ad optimális megoldást, amelyekben a pontok kifoka maximum 1, vagyis gráfelméleti kifejezéssel befenyő.

1. lépés:

Számoljuk ki minden ponthoz a belőlük kivezető leghosszabb irányított út hosszát. Ezt a legegyszerűbben úgy tudjuk kiszámolni, hogy:

Vegyük a nulla kifokú pontokat a befenyőben., Adjunk ezeknek egy értéket, ami a többi pontnál is a belőle kivezető leghosszabb irányított út hosszát mutatja vagyis ezeknél a pontoknál 0 lesz.

2. lépés:

Vegyük azokat a pontokat, amikből mutat él ezekbe a nulla kifokú pontokba. Ezekhez rendelünk 1-es értéket vagyis ezekből a pontokból egyhosszú úton eljuthatunk a nulla kifokúakhoz. Ezt a lépést ismételgetve minden pontoz hozzá rendeljük a nulla kifokúaktól való távolságát.

3. lépés:

Készítsünk listát, amibe elsőnek a legnagyobb fokszámú pontot vagy pontokat majd az eggyel kisebbeket, majd a kettővel kisebbeket vesszük be és így tovább, amíg el nem fogynak a pontok. Majd ezen lista alapján listás ütemezéssel helyezzük el a munkákat a gépeken.

Leghosszabb út módszere (Longest Path Heuristic)

1.lépés: Számoljuk ki mindegyik csúcsra a leghosszabb belőle kivezető irányított út hosszát l_i -t.
2.lépés: Tegyük be a munkákat egy listába l_i szerint csökkenő sorrendbe, majd végezzük el erre a listára a listás ütemezést.

Fujii, Kasami és Ninomiya algoritmusa (FKN algoritmus)

Ez az algoritmus csak két gépes feladatokra alkalmazható, ha adott a munkák egy irányított gráfban tárolt precedencia halmaza.

1. lépés:
Adott egy irányított gráf, amiben a megelőzési feltételeket jelennek meg. Ebből készítsünk egy módosított gráfot, amiben két pontot azaz két munkát akkor köt össze él, ha az eredeti gráfunkban nincs köztük irányított út.
2. lépés:
Keressünk maximális párosítást ebben a módosított gráfban.
3. lépés:
Hasonlítsuk össze a párosításunkat az eredeti gráfunkban lévő forrásokkal, hogy vezet-e köztük a párosításban szereplő él. Ha nincs két forrás, akkor az azt jelenti, hogy az egyik gépen az a munka a másik gépen pedig ugyanolyan hosszú állás idő lesz, ha van két forrás, de a párosításban nem megy köztük él akkor tétel szerint átalakítható a párosítás úgy, hogy a források valamint a párjaik legyenek összekötve.
4. lépés:
Vegyük be az ütemezésbe azt a párosításbeli élhez kapcsolódó munkapárt (egyiket az egyik gépre másikat a másikra). Ezt és a 3. lépést ismételgessük, amíg el nem fogynak a munkák.

Coffman és Graham algoritmusa (CG algoritmus)

Ez az algoritmus is csak kétgépes esetekben működik és csak irányított gráfokra.

1 lépés:

Vegyük az összes nyelőt és sorszámozzuk meg őket tetszőleges sorrendben.

2 lépés:

Vegyük az előbb említett nyelők szüleit vagyis azokat a pontokat, amikből csak ezekbe a nyelőkbe mutat él és címkézzük meg őket a velük összekötött nyelők sorszámaival (csökkenő sorrendben), majd a címkék növekvő lexikografikus sorrendje szerint sorszámozzuk meg ezeket a csúcsokat is. Ezt ismételgessük, amíg el nem fogynak a csúcsok.

3 lépés:

Az így kapott sorrend szerint felállított listával listás ütemezéssel ütemezzük a munkákat a gépre.

IV. Egygépes feladatok

1. feladat

Adjunk optimális megoldást a $1\|\sum \omega_i c_i$ *problémára a következő munkák halmazára:*

$\tau = (7,4,6,3,2,8,7,5,3,4,1)$
$\omega = (2,8,1,4,7,9,2,1,8,3,6)$

Megoldás:

Első lépésként számoljuk ki $\frac{\tau}{\omega}$ értékét minden munkára:

$$\frac{\tau}{\omega} = (\frac{7}{2}, \frac{1}{2}, 6, \frac{3}{4}, \frac{2}{7}, \frac{8}{9}, \frac{7}{2}, 5, \frac{3}{8}, \frac{4}{3}, \frac{1}{6})$$

Második lépésként rendezzük WSPT sorrendbe vagyis $\frac{\tau}{\omega}$ szerint nem csökkenő sorrendbe:

$$sorrend = (11,5,9,2,4,6,10,7,1,8,3)$$
$$\frac{\tau}{\omega} = (\frac{1}{6}, \frac{2}{7}, \frac{3}{8}, \frac{4}{8}, \frac{3}{4}, \frac{8}{9}, \frac{4}{3}, \frac{7}{2}, \frac{7}{2}, \frac{5}{1}, \frac{6}{1})$$

Tétel szerint erre a feladatra a WSPT sorrend optimális ütemezést ad. Tehát az optimális ütemezés: (11,5,9,2,4,6,10,7,1,8,3)

Ez után már csak a $\sum \omega_i c_i$ értékét kell kiszámolni:

$c_i : (1,3,6,10,13,21,25,32,39,44,50)$
$\omega_i : (6,7,8,8,4,9,3,2,2,1,1)$
$c_i \cdot \omega_i : (6,21,42,80,52,189,75,64,78,44,50)$
$\sum \omega_i c_i = 701$

2. feladat

Adjunk optimális megoldást a $1 \| \sum \omega_i c_i$ *problémára a következő munkák halmazára, ahol nem a* $\sum \omega_i c_i$ *minimumára, hanem maximumára vagyunk kíváncsiak:*

$\tau = (7,4,6,3,2,8,7,5,3,4,1)$
$\omega = (2,8,1,4,7,9,2,1,8,3,6)$

Megoldás:

Mivel tétel szerint minden olyan munkák cseréje, amire $\frac{\tau_i}{\omega_i} < \frac{\tau_j}{\omega_j}$ és i<j rontja az optimális értéket vagyis növeli $\sum \omega_i c_i$ értékét, ezért a fordított sorrend $\frac{\tau_i}{\omega_i} > \frac{\tau_j}{\omega_j}$ i<j lesz a legrosszabb vagyis maximális azaz pont a 2. feladat célkitűzésének megfelelő.

$\tau = (6,5,7,7,4,8,3,4,3,2,1)$
$\omega = (1,1,2,2,3,9,4,8,8,7,6)$
$c_i = (6,11,18,25,29,37,40,44,47,49,50)$
$c_i * \omega_i = (6,11,36,50,87,333,160,352,376,343,300)$
$\sum c_i \omega_i = 2054$

3. feladat

Adott munkaidők és súlyok mellett arra vagyunk kíváncsiak, hogy melyik munkát utolsónak választva lesz a legkisebb a munka súlyozott kezdési ideje.

A munkák az előző két feladatban már megismertek:

$\tau = (7,4,6,3,2,8,7,5,3,4,1)$
$\omega = (2,8,1,4,7,9,2,1,8,3,6)$

Megoldás:

Itt lényegében csak ki kell számolni, minden munkára a hozzátartozó súlyozott kezdési időt vagyis az előző munka súlyozott befejezési idejét.

$\tau = (6,5,7,7,4,8,3,4,3,2,1)$
$\omega = (1,1,2,2,3,9,4,8,8,7,6)$
$\tau_i * \omega_i = (6,5,14,14,12,72,12,32,24,14,6)$
$\omega_i * \sum \tau_i = (50,50,100,100,150,450,200,400,400,350,300)$
$\omega_i * \sum \tau_i - \tau_i * \omega_i = (44,45,86,86,138,378,196,368,376,336,294)$

Ezek közül már csak ki kell választani a legkisebbhez tartozó munkát. Az kerül az utolsó helyre, ami jelen esetben a feladat kitűzésében szereplő 3.-dik munka lesz ($\tau = 6, \omega = 1$)

4. feladat

Általánosítsuk a 3. feladatot bármilyen k munkára.

Megoldás:

Ekkor elég felhasználni az előző feladat eredményeit, majd sorba rendezni. Ez ugyanis annak felel meg, mintha először kiszámolnák az utolsó munkára aztán az utolsó előtti munkára és így tovább egészen k-ig. Könnyen látható, hogy ez optimális, mivel a módszer csak a használt konstansokban tér el attól, hogy lépésenként számoljuk ki, melyik munka legyen az utolsó majd az utolsó előtti. Hiszen ha másik sorrend jobb lenne akkor az egyik munka, amit későbbre raktunk az új sorrendben már korábbi lépésben is jobbnak kellett volna lennie. Szóval csak sorba kell raknunk az előző feladat $\omega_i * \sum \tau_i - \tau_i * \omega_i$ értékeit növekvő sorrendben, amivel meg is kapjuk az utolsó k munkát úgy, hogy mindegyiknél a legminimálisabb súlyozott kezdési idő lesz.

5. feladat

Találjunk algoritmust, amely megoldja a $1|\tau_i = 1|\sum \omega_i T_i$ *a következő súlyú és határidejű munkák halmazára:*

$\omega = (7,4,6,3,2,1,8,5,4,9)$
$d = (4,3,7,8,5,7,8,4,9,6)$

Megoldás:

Mivel minden munka egységnyi hosszú, ezért a munkák befejezési ideje könnyen felírható:

$c_i = (1,2,3,4,5,6,7,8,9,10)$

Ekkor a hasonló határidejű munkák közül a legnagyobb súlyú munka az utolsó nem késő helyen van optimális pozícióban, mivel későbbi helyen egyértelműen csak a későket növelné korábbi helyen pedig nem lenne hatékony.

$c_i = (1,2,3,4,5,6,7,8,9,10)$
$d_i = (?,?,3,4,5,6,7,8,9,?)$
$\omega_i = (?,?,4,7,2,9,6,8,4,?)$

Ezután viszont kénytelenek vagyunk a megmaradt munkákat végig számolva kiszámolni, hogy melyik munka késné a legkevesebbet az adott pozícióban, a legutolsó üressel kezdve és hátulról előre haladva:

10 pozíciónál:

$d = (8,7,4)$
$\omega = (3,1,5)$
$10 - d = (2,3,6)$
$\omega(10-d) = (6,3,30)$

Itt a másodiknál a legkisebb súlyozott késés, vagyis ez kerül a 10 pozícióra.

2-es pozíció

$d = (8,4)$
$\omega = (3,5)$
$2 - d = (-6,-2)$
$\omega(10-d) = (0,0)$

Ha két optimálisat kapunk, akkor a korábbi határidejűt tegyük a későbbi pozícióra, hogy a többi minél kevesebb késsen, de jelen esetben mindegy, mivel egyik sem késik, ezért bármelyik kerülhet korábbi pozícióba is. Így az optimális megoldás a következő sorrend:

$c_i = (1,2,3,4,5,6,7,8,9,10)$
$d_i = (8,4,3,4,5,6,7,8,9,7)$
$\omega_i = (3,1,4,7,2,9,6,8,4,1)$

A minimális súlyozott késés: 3

6. feladat

Adott n munka és egy gép és k egész szám, valamint ezek egy szétválasztása {E,T} ahol E nem késő T pedig késő munkák halmaza, létezik-e ekkor olyan ütemezés, amire a teljes késési idő kevesebb mint k?

Megoldás:

Két eshetőség lehetséges:

- ha már az adott szétválasztás késési ideje kisebb mint k ekkor nincs tenni való
- ha viszont több akkor első lépésként meg kellene határozni, hogy megoldható-e a feladat egyáltalán. Ehhez viszont meg kellene oldani a $1||\sum T_i$ feladatot, ami NP-teljes probléma így ez a feladat is az.

7. feladat

Adjunk olyan példát, amire $1||T_{max}$ probléma megoldásához nem szükséges az EDD sorrendet használni.

Megoldás:

Ha minden munkáknak ugyanaz a határideje, akkor minden sorrend optimális megoldás, mivel nincs állás idő, ezért az utolsó munka befejezési ideje minden sorrendnél ugyanannyi. Így a maximális késés is mindig ugyanannyi. Tulajdonképpen minden sorrend EDD-ben van anélkül, hogy abba rendeztük volna és a feladat triviálisan megoldható anélkül, hogy ezt felhasználnánk.

8. feladat

Bizonyítsuk be:

a.) Ha az EDD egyetlen késő munkát ad eredményül valamilyen egy gépes feladatra, akkor ez a $1||\sum T_i$ feladatra is megoldás lesz.

Igaz, mivel EDD-ben a legutolsó munkának lesz a legkésőbbi határideje. Így bármilyen más munka kerül a helyére azzal biztosan nő az összes késés értéke, vagyis az EDD által adott sorrend nem csak a legnagyobb késést minimalizálja hanem az összes kését is.

b.) Ha minden munkának ugyanaz a határideje akkor az SPT sorrend minimalizálja a $\sum T_i$ értékét is

Igaz, mivel ekkor érdemes az azonos határidőkre úgy tekinteni, mint egy általános határidőre amíg végezni kellene a munkákkal. Ekkor értelemszerűen az a legjobb, ha minél több munkával végzünk vagyis a legrövidebbekkel kezdjük és a leghosszabbak kerülnek a végére. Ezzel elérjük, hogy az előtte lévők minél kevesebbet késsenek vagyis az SPT sorrend lesz optimális. Könnyű látni, hogyha felcseréljük bármelyik két munkát ebben a sorrendben akkor a második munka, ami az első helyére kerül hosszabb, ezért később ér véget. Vagyis ha késő akkor növeli a késést, ha nem késő akkor pedig változatlan marad, hiszen az első munka új befejezési ideje ugyanakkor lesz, mint az eredeti SPT sorrendben a régi munkának.

Egy másik érvelés hogy az SPT sorrend minimalizálja $\sum c_i$ a $\sum T_i = \sum (c_i - d_i)$ és mivel d_i állandó így vissza vezethető az SPT sorrendre.

c) Legyen n munka, aminek bármilyen ütemezésében mind késő munka lesz ekkor az SPT sorrend szintén az optimális megoldást adja $1||\sum T_i$ *feladatra*

Ezt a feladatot könnyen vissza vezethetjük az előzőre. Mindössze csak ki kell választanunk a legkisebb határidejűt a többi határidejét lecsökkentjük ugyanennyire, ezzel mindössze csak egy konstans számot adtunk hozzá a megoldáshoz, amit levonva visszakapjuk az eredeti optimális értéket. Így elértük, hogy minden határidő egyforma legyen azaz visszavezettük az előző példára.

d) Legyen n adott munka amik SPT sorrendben vannak ütemezve és, ekkor minden munka késik, bizonyítsuk be, hogy ez az ütemezés optimális lesz $1||\sum T_i$ *feladatra.*

Igaz, mivel ekkor ugyanazt a meggondolást kell végig követni mint, amit a b) feladatban alkalmaztunk: Az SPT sorrendben, ha felcserélünk két munkát, akkor a hosszabb munka befejezési ideje nőni fog. Valamint, ha nem szomszédos munkákat cseréltünk fel, akkor az összes köztes munka befejezési ideje nőni fog. Így késés mértéke is nőni fog, mivel minden sorrend elő állítható az SPT-sorrendből ilyen cserékkel és mindnél csak nőhet a késés, ha későbbi munkát cserélünk korábbira. Ezért az SPT sorrend adja a legkisebb kését vagyis optimalizálja az $1||\sum T_i$ feladatot.

9. feladat

Oldjuk meg a $1|\tau_i=1|\sum \omega_i u_i$ *feladatot a greedy algoritmussal a következő munkák halmazán:*

$\omega=(7,6,4,5,6,3,8)$
$d=(4,2,3,5,6,5,3)$

Megoldás:

A greedy algoritmus szerint rendezzük a munkákat a súlyok szerint nem növekvő sorrendbe:

$\omega=(8,7,6,6,5,4,3)$
$d=(3,4,2,6,5,3,5)$

Az első munkát beválasztjuk a másodikat is, a harmadikat is. Ez már késő lesz így a következőt elutasítjuk az ötödiket, beválasztjuk a hatodikat is a hetedik újra elutasítjuk mivel a hatodik késő volt. Így az új sorrend:

$\omega=(8,7,6,5,4|6,3)$
$d=(3,4,2,5,3|6,5)$

Rendezzük a beválasztott munkákat EDD-be:

$\omega=(6,8,4,7,5|6,3)$
$d=(2,3,3,4,5|6,5)$

Így viszont már csak az utolsó munka fog késni hármas súllyal, mivel ez a legkisebb súlyú munka és mivel nincs hetes határidejű munka ezért biztos, hogy ez az optimális megoldás.

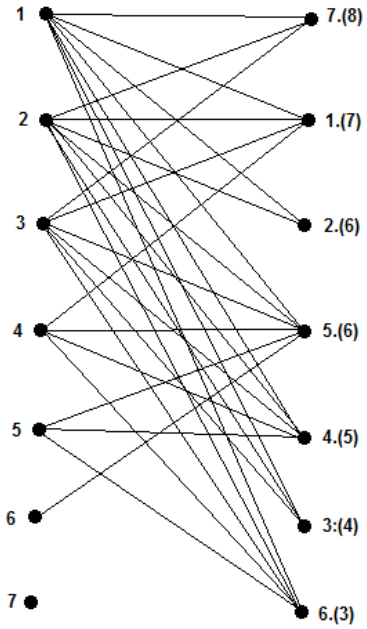
10. feladat

Alkalmazzuk a korábban említett speciális maximális párosítás kereső módszert a 9-es feladatban szereplő munkákra:

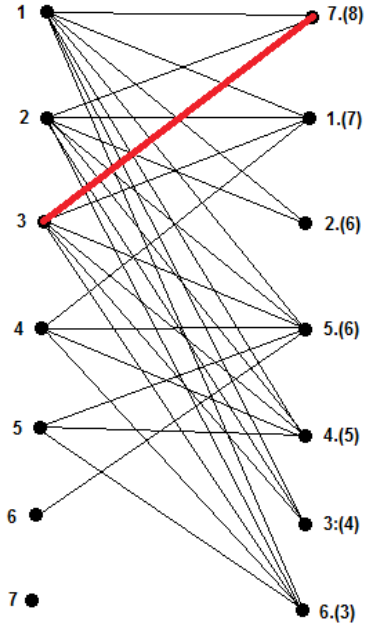
Megoldás:

Építsük fel a gráfját:

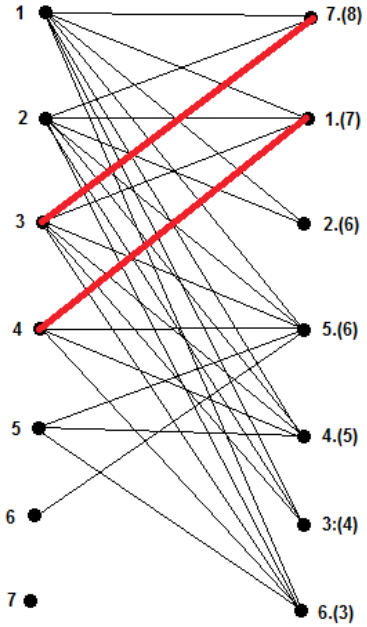
10.1.ábra: Felvettük a gráf pontjait úgy, hogy a baloldali pontok a munkák sorrendjét mutatja a jobboldali pontok pedig maguk a munkák a feladatban megadott sorszámuk és zárójelben a súlyuk szerint.



10.1 ábra



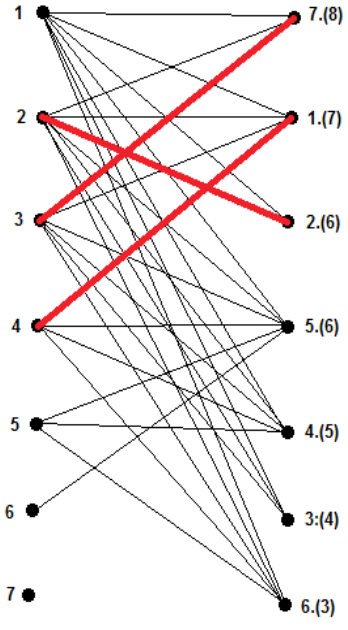
10.2 ábra



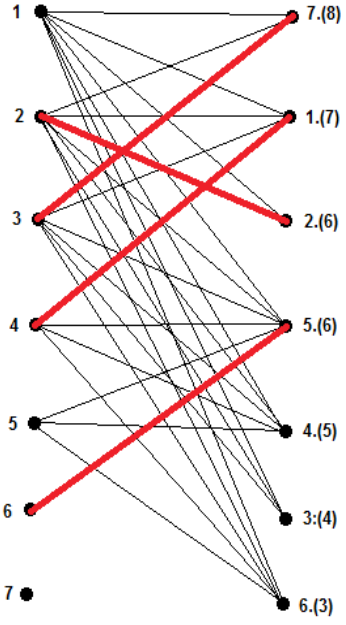
10.3 ábra

10.2.ábra: Az algoritmus szerint tegyük be az első munkát a legkésőbbi helyre, ahol még nem késik majd ezután tegyünk ugyanígy a második munkával is, ha nem foglalt a hely (ez a 10.3.ábrán látható), ha foglalt lenne akkor az eggyel előbbi helyet választjuk.

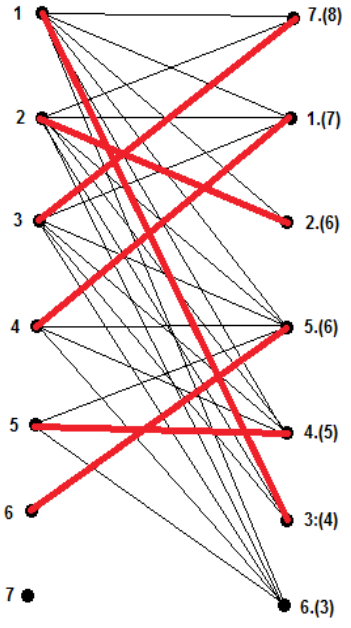
Az előző lépést ismételgetve kapjuk a 10.4.-es és 10.5.-ös ábrát a 10.6.ábrán jön elő először az a helyzet, amikor a munkát nem a saját határidejének megfelelően a legutolsó lehetséges helyre rakjuk, hanem a legkésőbbi üres helyre. Jelen esetben mivel a 2-es 3-as hely már foglalt csak az egyes lesz jó.



10.5 ábra



10.6 ábra



10.7 ábra

Így viszont a 6.(3) számú(súlyú) munkának nem jut hely, ami várható volt hiszen a 7. helynek megfelelő csúcshoz nem vezet egy él sem vagyis az oda kerülő munka - bármelyik is legyen - késni fog.

Ez a megoldás minden bizonnyal optimális, mivel a többi munka súlyozott késési értéke az utolsó pozícióban: 21,30,16,10,6,32. Ezek között nincs 6-nál kisebb így kisebb célfüggvény értékű optimális megoldás sincs.

11. feladat

Adjunk példát, olyan munkák halmazára amire a Moore algoritmusa elromlik vagyis nem ad optimális megoldást a $1||\sum \omega_i u_i$ *problémára.*

$\tau_i = (3,2,1)$
$d_i = (3,4,5)$
$\omega_i = (3,2,1)$

Megoldás:

Ha követjük Moore algoritmusát, akkor az első lépés hogy EDD-be rakjuk a munkákat. Most a munkák már EDD-ben vannak, ezért a következő lépés az, hogy végig nézzük a munkákat késők-e vagy nem. Az első késő munka a második munka lesz. Ekkor kivesszük a leghosszabb addigi munkát jelen esetben az elsőt, majd folytatjuk az ütemezést a harmadik munka így nem lesz késő vagyis a Moore algoritmus szerint 2.3.1. sorrend a legjobb vagyis:

$\tau_i = (2,1,3)$
$d_i = (4,5,3)$
$\omega_i = (2,1,3)$
$u_i = (0,0,1)$

itt az $\sum \omega_i u_i = 3$ de egyértelmű, hogy ennél például a következő sorrend is jobb

$\tau_i = (3,1,2)$
$d_i = (3,5,4)$
$\omega_i = (3,1,2)$
$u_i = (0,0,1)$

ahol a $\sum \omega_i u_i = 2$

12. feladat

Oldjuk meg Moore algoritmusával az $1\|\sum u_i$ *problémát a következő munkák halmazára*

$\tau_i = (3,5,2,4,2,6)$
$d_i = (10,12,14,10,15,9)$

Megoldás:

Első lépésként rakjuk EDD-be a munkákat:

$\tau_i = (6,3,4,5,2,2)$
$d_i = (9,10,10,12,14,15)$
$c_i = (6,9,13,18,20,22)$

Itt az első munka nem késik a második sem, de a harmadik már igen ezért a leghosszabb munkát vagyis az elsőt kivesszük és a végére tesszük.

$\tau_i = (3,4,5,2,2,6)$
$d_i = (10,10,12,14,15,9)$
$c_i = (3,7,12,14,16,22)$

Ekkor az első, második, harmadik, negyedik munka nem, csak az ötödik munka késik. Így már nem kell többet cserélni az eredmény attól nem javulhat. Így a végeredmény: $\sum u_i = 2$

13. feladat

Használjuk Lawler algoritmusát, hogy az $1\|f_{max}$ *problémára a következő munkák halmazán. Itt az 'f' büntetésfüggvények egy halmazát jelenti, aminek tagjai hozzá van rendelve az egyes munkákhoz.*

$\tau_i = (3,4,5,2,2,6)$
$d_i = (10,10,12,14,15,9)$
$c_i = (3,7,12,14,16,22)$

Megoldás:

Első lépésként számoljuk ki, hogy mennyi az első helyettesítési érték $\sum \tau_i = 24$ és helyettesítsük be minden függvénybe:

$f_i = (576,120,117.57,48.98,138.24,191102976)$

vagyis a 4-es munka kerül az utolsó pozícióba, ekkor az új helyettesítési érték: $\sum \tau_i = 18$

$f_i = (324,90,76.36,58.32,34012224)$

vagyis az 5-ös munka a legkisebb így az kerül az utolsó előtti helyre, ekkor az új helyettesítési érték

$\sum \tau_i = 16$

$f_i = (256,80,64,16777216)$

vagyis 3-dik munka lesz a legkisebb, tehát ez kerül az utolsó előtti előtti helyre, ekkor az új helyettesítési érték $\sum \tau_i = 12$

$f_i = (144,60,2985984)$

vagyis 2-dik munka lesz a legkisebb, tehát ez kerül hátulról a negyedik helyre, ekkor az új helyettesítési érték $\sum \tau_i = 9$

$f_i = (81,531441)$

vagyis az első munka lesz a legkisebb, tehát ez kerül hátulról az ötödik helyre, ekkor az új helyettesítési érték $\sum \tau_i = 4$

$f_i = (4096)$

vagyis, mivel ez a kivett értékek közül a legnagyobb, tehát ez lesz az f_{max} értéke

V. Többgépes ütemezési feladatok

14. feladat

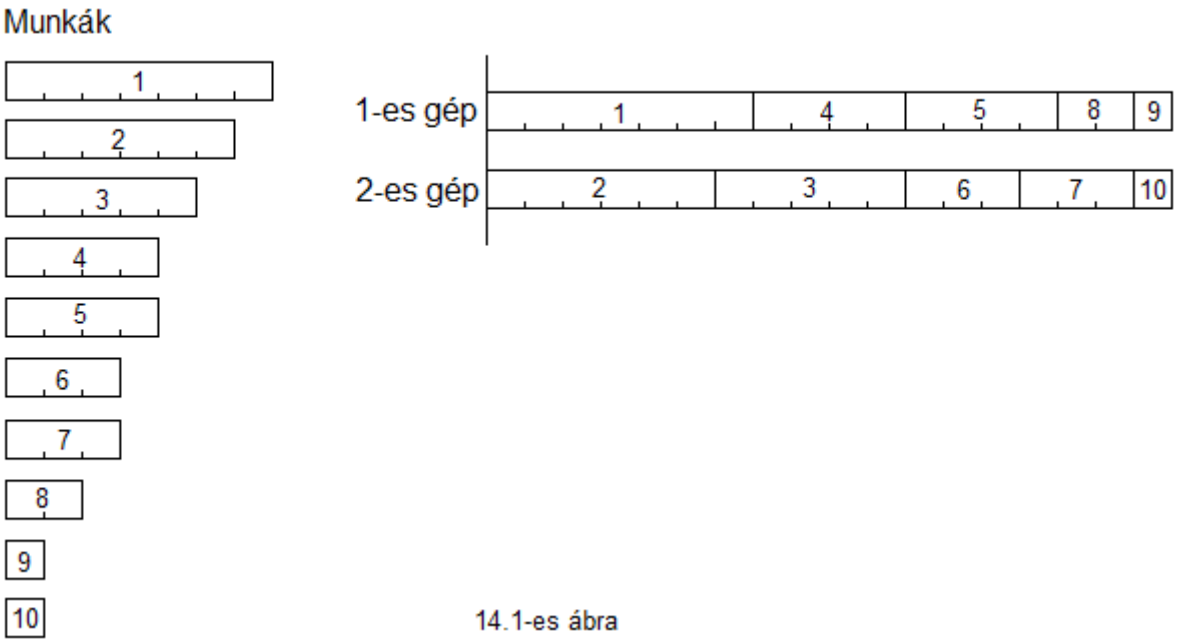
Adjunk algoritmust, ami polinom időben (nem feltétlenül optimálisan) megoldja a $P2||C_{max}$ *feladatot a* $\tau = (7,4,3,2,1,1,5,4,6,3)$ *munkák halmazán.*

Megoldás:

Ennél a feladatnál lényeges, hogy nem kell optimális megoldás, hiszen ennek a megtalálása NP-teljes. Például az LPT algortimus jól használható egy közel optimális megoldás megtalálásához. Első lépésként rendezzük a munkákat munkaidők szerint nem növekvő sorrendbe:

$\tau = (7,6,5,4,4,3,3,2,1,1)$

Majd pakoljuk be sorban ezeket a munkákat a két gép valamelyikére, amikor az épp elérhetővé válik.

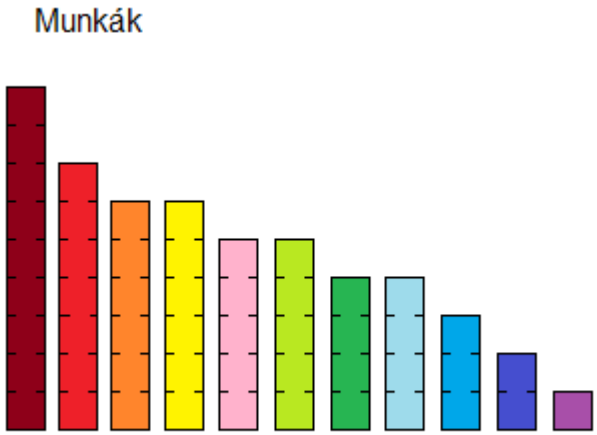


14.1-es ábra

Így mindkét gépen a munkaidők összege 18, vagyis szerencsésen pont az optimális megoldást kaptuk.

15. feladat

Alkalmazzuk a MULTIFIT algoritmust a $P||C_{max}$ *problémára* $m=3$ *gépen* $t=5$ *iterációs lépéssel a következő munkákon:* $\tau=(5,6,9,3,2,1,4,7,5,4,6)$



15.1 ábra

Megoldás:

Első lépésként határozzuk meg a kezdeti alsó és felső korlátokat:

$B_L = 17{,}33$
$B_U = 34{,}66$

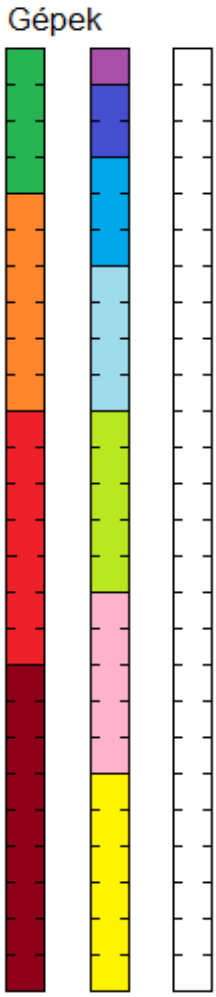
Majd második lépésként a kapacitást:

$C = (17{,}33 + 34{,}66)/2 = 26$

Harmadik lépésként vizsgáljuk meg, hogy emellett a kapacitás mellett az FFD algoritmus tudja e ütemezni az adott munkákat 3 gépen vagy több gép kell neki, ehhez nem növekvő sorrendbe kell rendezni a munkákat:

$\tau = (9{,}7{,}6{,}6{,}5{,}5{,}4{,}4{,}3{,}2{,}1)$

Majd ütemezzük sorban a gépekre ahova befférnek (3 gép van és most mindegyikbe 26 egységnyi munkaidejű munka fér), ennek a menete részletesen a következő: az első 3 munka kerül az első gépre majd a következő 3 a második gépre majd a 7. munka újra az első ládába majd az utolsó 4 a második ládába ez a 15.2-es ábrán látható:



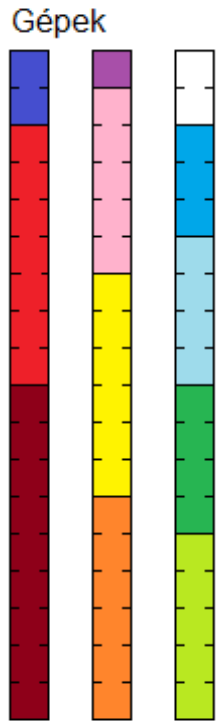
15.2 ábra

Sikerült két ládába belepakolni vagyis tovább az iteráció folytatásával tovább csökkenthető a kapacitás korlát.

A fent látható iterációs lépést a feladat szerint ötször kell megismételni:

$i = 1 : C = 26,\ FFD(T, C) \leqslant 3,\ B_1(1) = 26,\ B_2(1) = 17{,}33$
$i = 2 : C = 21{,}66,\ FFD(T, C) \leqslant 3,\ B_1(2) = 21{,}66,\ B_2(2) = 17{,}33$
$i = 3 : C = 19{,}5,\ FFD(T, C) \leqslant 3,\ B_1(3) = 19{,}5,\ B_2(3) = 17{,}33$
$i = 4 : C = 18{,}41,\ FFD(T, C) \leqslant 3,\ B_1(4) = 18{,}41,\ B_2(4) = 17{,}33$
$i = 5 : C = 17{,}87,\ FFD(T, C) > 3,\ B_1(5) = 18{,}41,\ B_2(5) = 17{,}87$
$i = 6 : Stop$

Így a végső FFD szerinti ütemezés a 15.3 ábrán látható:

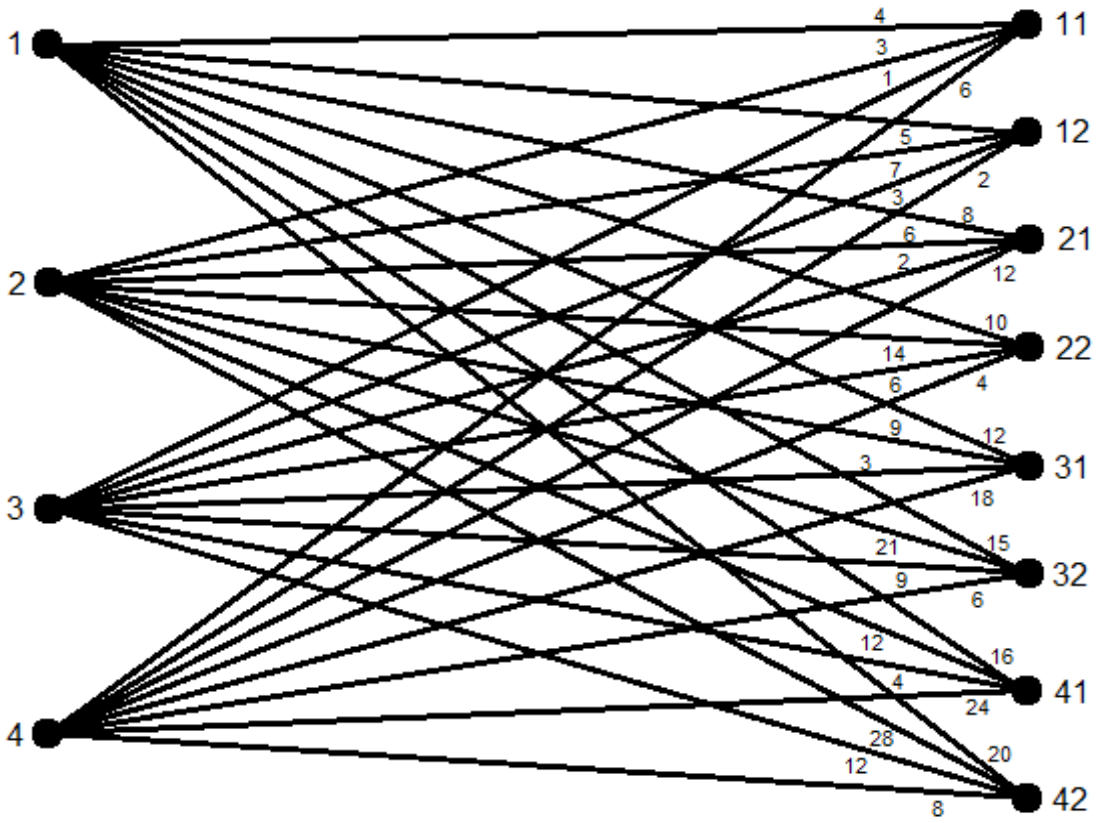


15.3 ábra

16. feladat

Alkalmazzuk a Bruno algoritmust az $R||\sum c_i$ *a következő munkák halmazára:*

$$\tau=\begin{bmatrix}4 & 5\\ 3 & 7\\ 1 & 3\\ 6 & 2\end{bmatrix}$$



16.1 ábra

Első lépésként készítsük el munkákból és pozíciókba helyezésekből a fent látható gráfot, majd keressük meg a maximális párosítást. (Ezt Gale Shaple algoritmussal és a Magyar módszerrel tehetjük meg)

Ebben a feladatban könnyen látható hogy az optimum: {(3,11)(1,12)(2,21)(4,22)}

17. feladat

Adjunk gyors algoritmust $P|prec,\tau_i=1|C_{max}\leqslant 2$ *feladat megoldására, vagyis keressünk olyan ütemezést, ami egységnyi hosszú munkákra párhuzamos gépeken ad gyors ütemezést, ha a maximális gépidő csak kevesebbet vagy egyenlő lehet mint 2.*

Tekintsük adottnak a munkák megelőzési feltételeit, mint egy irányított gráf, ahol az irányított él a megelőző munkától a megelőzendő felé mutat. Ekkor feltehetjük, hogy nincs 1-nél hosszabb irányított él különben a feladat nem megoldható. Továbbá azt is ellenőrizni kell, hogy nincs-e több munka, mint a gépek számának kétszerese, mivel ekkor szintén nem megoldható a feladat. Valamint azt is ellenőrizni kell, hogy nincs-e több az 1-nél nagyobb befokú pontokból mint amennyi gép van, hiszen ezek a munkák csak a második pozícióba kerülhetnek. A nullafokúakból lehet több mint amennyi gép van, hiszen ezek kerülhetnek első és második pozícióba, de második pozícióba már csak olyan munkák kerülhetnek amikből ki sem megy él. A feltételek ellenőrzése után az ütemezés már könnyű, hiszen csak a nulla befokúakat kell először ütemezni az első elérhető gépekre, majd a az 1 vagy annál nagyobb befokúakat a szabad gépekre a második pozícióba.

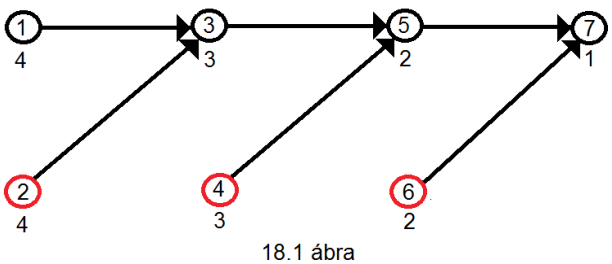
18. feladat

Adjunk példát olyan feladatra amiben a munkák egy irányított gráfban helyezkednek el, amiben minden pont kifoka 1, a munkák megmunkálási ideje pedig 1 vagy 2 egység lehet és Hu algoritmusát alkalmazva rossz vagyis nem optimális megoldást kapunk.

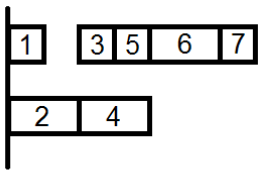
Vagyis: $P2|intree, \tau_i = \{1,2\}|C_{max}$

Megoldás:

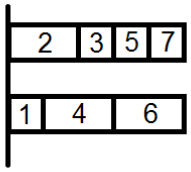
Adott 7 darab munka és 2 darab gép, ami közül 3 darab 2 egységnyi 4 darab 1 egységnyi megmunkálási idejű, és az 18.1-es ábrában látható befenyőben helyezkednek el (itt például 7-es munka befejezését követően következhet az 5-ös munka). Hu algoritmusa szerint rendeljünk értékeket minden munkához aszerint, hogy milyen hosszú a belőle kivezető leghosszabb irányított út. (a munkák számozása is eszerint van elkészítve). A sorszámok szerint végezzük el a két gépen a listás ütemezést, erre a 18.2.es ábra szerinti ütemezést kapjuk viszont ez nem optimális, hiszen a 18.3-as ábra szerinti ütemezés ugyancsak eleget tesz a feltételeknek de C_{max} értéke 2-vel kisebb.



18.1 ábra



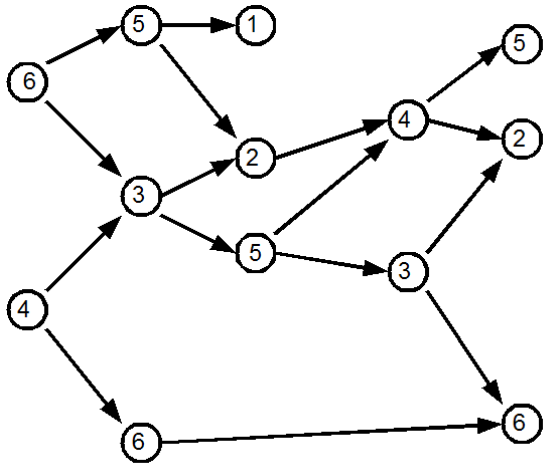
18.2 ábra



18.3 ábra

19. feladat

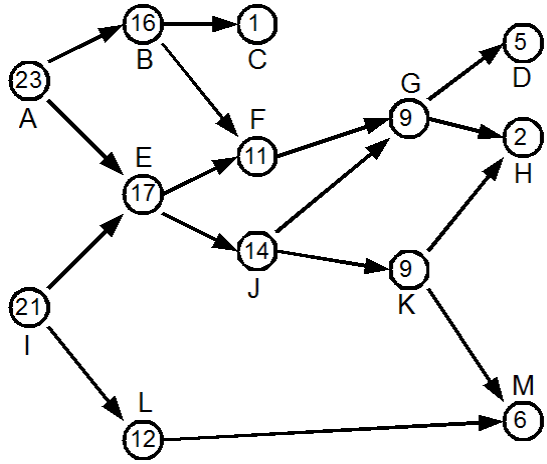
Alkalmazzuk a Leghosszabb út (Longest Path) algoritmust a $P3|prec|C_{max}$ *problémára a következő gráfon:*



19.1 ábra

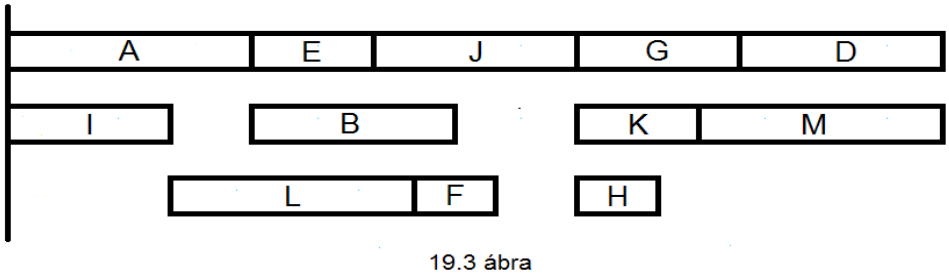
Megoldás:

Itt a csúcsok felelnek meg a munkáknak és az értékek pedig a munkaidők, ez alapján minden csúcshoz kiszámíthatjuk a leghosszabb belőle kivezető irányított út hosszát és ezt beírva készíthetünk egy új gráfot:



19.2 ábra

Így az ütemezés a következő lesz:



19.3 ábra

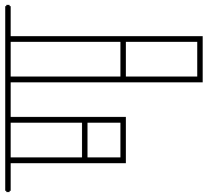
20. feladat

Adjunk példát olyan láda pakolási feladatra amit az FFD algoritmus nem ad optimális megoldást. A láda pakolási feladatban a gépeket ládáknak hívjuk és rendelkeznek kapacitással, hogy mennyi súlyt pakolhatunk bele. A tárgyaknak pedig nem munkaideje van, hanem súlya.

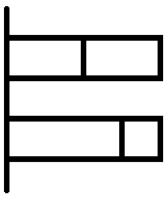
Megoldás:

Legyen például 2 láda és $\tau=\{2,1,3,2\}$ súlyú tárgyak valamint a gépek kapacitása legyen 5.

Az FFD algoritmushoz először LPT-be kell rendezni a munkákat. $\tau=\{3,2,2,1\}$ Majd az FFD szerint folytatva rakjuk az LPT szerinti első és második tárgyat az első ládába, majd a maradék két tárgyat a második ládába. Ekkor viszont látható, hogy nem kapunk optimális megoldást hiszen a 10.2-es ábrán látható ütemezés jobb lenne.



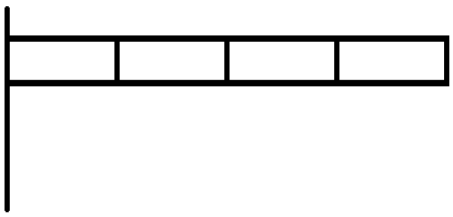
20.1 ábra



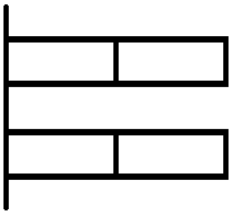
20.2 ábra

Általánosságban is, ha ládák kapacitása lényegesen nagyobb, mint amit az optimum megkövetelne akkor a megoldás is sokkal rosszabb lesz, mint az optimum.

Például ha van 4 tárgyam mindegyik 3 egység nagyságú és van két ládám mindegyikbe 12 egység fér, akkor az FFD azt a megoldást adja, hogy tegyük az összes tárgyat az első ládába, ez viszont már lényegesen rosszabb mint az optimális 6 egység az egyikben és 6 egység a másik ládában ütemezés.



20.3 ábra



20.4 ábra

21. feladat

Adjunk polinom idejű megoldást a Láda pakolási problémára, ahol a tárgyak súlya (C/3,C] intervallumban van, itt C a ládák kapacitása.

Megoldás:

A feladatban megszabott feltételek miatt maximum egy vagy kettő tárgy lehet egy ládában.

1. lépés:

Rendezzük a tárgyakat csökkenő sorrendbe a súlyuk szerint.

2. lépés:

Itt is alkalmazzuk a listás ütemezést. Vagyis rakjuk a legnagyobb tárgyat az első ládába majd a második legnagyobbat a másodikba és így tovább, amíg el nem fogynak a ládák. Ezután a következő tárgyat az első elérhető ládába az utána következőt a következő elérhető ládába és így tovább, amíg el nem fogynak a tárgyak.

Persze itt is megtörténhet, hogy túl lépjük a kapacitást, de ekkor biztos, hogy más ütemezéssel se lehetne megoldani hiszen, ha m láda van és már az első m tárgyat se tudtuk úgy elhelyezni, hogy valamelyik ne lépné túl a kapacitást. Ez azt jelenti, hogy már az elsőt se tudtuk. Vagyis a tárgy nagyobb a ládánál, ami ellent mond a feladat kitűzésének. Így csak akkor lehetséges, hogy túl lépjük a kapacitást, amikor a második tárgyat tesszük bele valamelyik ládába. Ezért ki kell kötni, hogy a legkisebb és a hozzá párosítható legnagyobb tárgy (amivel még nem lépi túl a láda kapacitását) között nincs 2m-nél több tárgy.

22. feladat

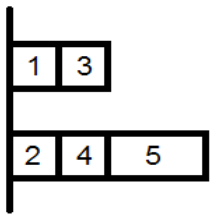
Adjunk példákat olyan feladatokra, amikor a listás ütemezés a $P||C_{max}$ *az optimálishoz képest lényegesen rosszabb megoldást ad.*

1-es példa

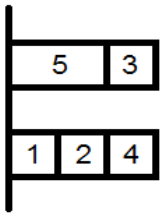
A munkák listája: $\tau=(1,1,1,1,2)$

22.1.1-es ábra: listás ütemezés

22.1.2-es ábra: optimális ütmezés (például az LPT szerinti ütemezés)



22.1.1 ábra



22.1.2 ábra

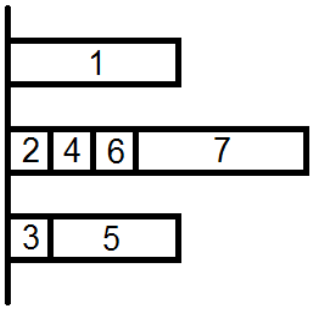
Itt könnyen látható, hogyha 2 gép van akár a leghosszabb munka munkaidejének a fele is lehet a különbség a listás és az optimális ütemezés között.

2-es példa

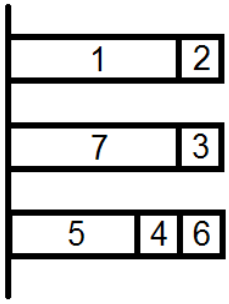
A munkák listája: $\tau=(4,1,1,3,1,1,4)$

22.2.1-es ábra: listás ütemezés

22.2.2-es ábra: optimális ütemezés (például az LPT szerinti ütemezés)



22.2.1 ábra



22.2.2 ábra

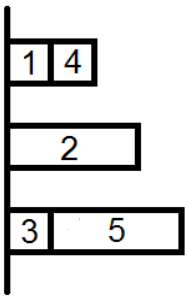
Itt az figyelhető meg, hogy az optimális és a listás ütemezés között, akár a leghosszabb munka nagyságrendű különbség is lehet.

3-as példa

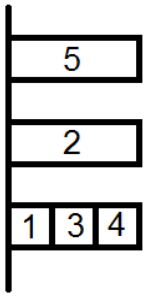
A munkák listája: $\tau=(1,3,1,3,1)$

22.3.1-es ábra: listás ütemezés

22.3.2-es ábra: optimális ütemezés (például az LPT szerinti ütemezés)



22.3.1 ábra



22.3.2 ábra

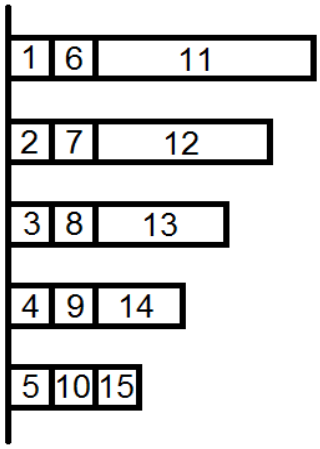
Itt az látható, hogy már viszonylag kevés munka esetén is előállhat olyan helyzet, amikor a listás ütemezés rossz megoldást ad. Jelen esetben például 2m-nél kevesebb munka van.

4-es példa

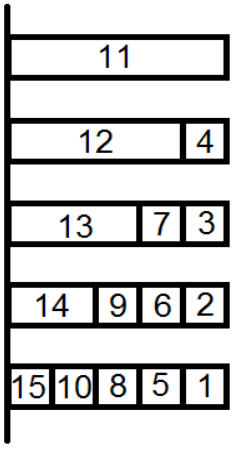
A munkák listája: $\tau=(1,1,1,1,1,1,1,1,1,1,5,4,3,2,1)$

22.4.1-es ábra: listás ütemezés

22.4.2-es ábra: optimális ütemezés (például az LPT szerinti ütemezés)



22.4.1 ábra



22.4.2 ábra

Itt az látható, ha sok munka van és azok között is sok egységnyi hosszú munka van, akkor se valószínű, hogy a listás ütemezés optimális megoldást ad. Ezzel szemben az LPT-nek annál jobb, minél több munka van és azon belül is, minél több az egységnyi munka.

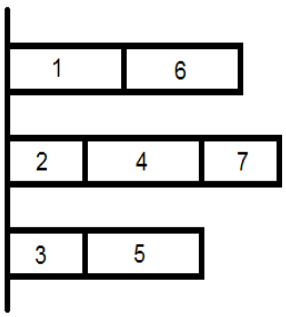
5-ös példa

A munkák listája: $\tau=(3,2,2,3,3,3,2)$

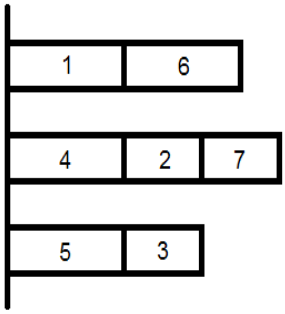
22.5.1-es ábra: listás ütemezés

22.5.2-es ábra: LPT szerinti ütemezés

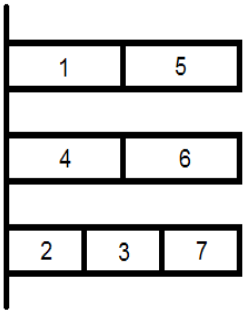
22.5.3-as ábra: optimális ütemezés



22.5.1 ábra



22.5.2 ábra



22.5.3 ábra

Itt az látható, hogy nem nehéz olyan feladatot találni amire egyik sem: se a listás, se az LPT szerinti ütemezés nem ad optimális megoldást, sőt a fenti esetben pontosan ugyanolyan rosszat adnak. Viszont ha kiszámoljuk, hogy egy gépre átlagosan mennyi munkaidőnek kellene jutnia és ezt az értéket adjuk meg maximumnak az FFD algoritmusnak, akkor az így egy optimális ütemezést ad de csak erre a feladatra.

23. feladat

Alkalmazzuk a Knuth-Kleitman Algoritmust a következő feladatra:

$m=5, \tau=(4,7,8,12,3,7,9,17,4,2,8,7,15,9,8,11)$, $k=7$

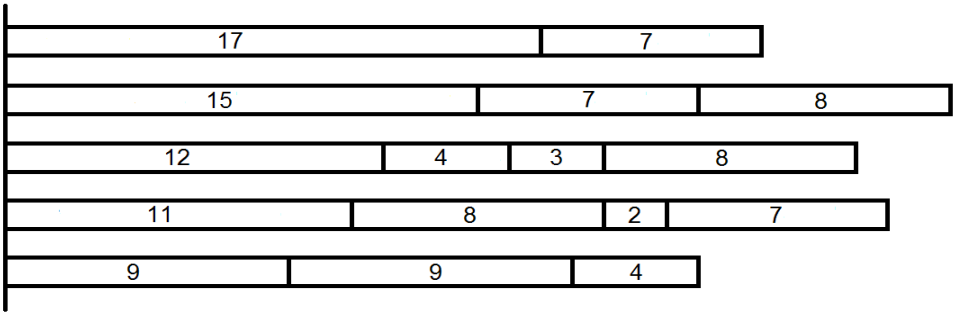
Megoldás:

Első lépésként szedjük ki a hét leghosszabb munkát: a 17-est, utána a 15-öst, majd a 12, 11, 9, 9, 8-ast így a maradék munkák listája a következő:

$\tau=(4,7,3,7,4,2,8,7,8)$

Ezek alapján a listás ütemezést használva, ahogy azt a Knuth-Kleitman algoritmus leírja a következő lesz:

Így a $C_{max}=30$



23.1 ábra

24. feladat

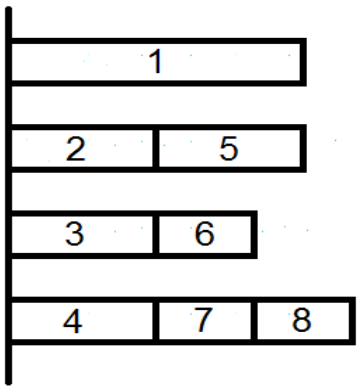
Adjunk példát olyan feladatra, amiben a n=2m vagyis kétszer annyi munka van mint gép és az LPT szerinti ütemezés nem ad optimális megoldást:

A munkák listája: $\tau=(6,3,3,3,3,2,2,2)$

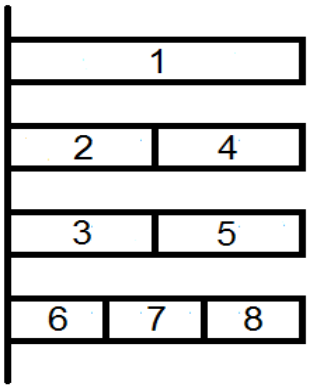
Megoldás:

24.1-es ábra: LPT szerinti ütemezés

24.2-as ábra: optimális ütemezés



24.1 ábra



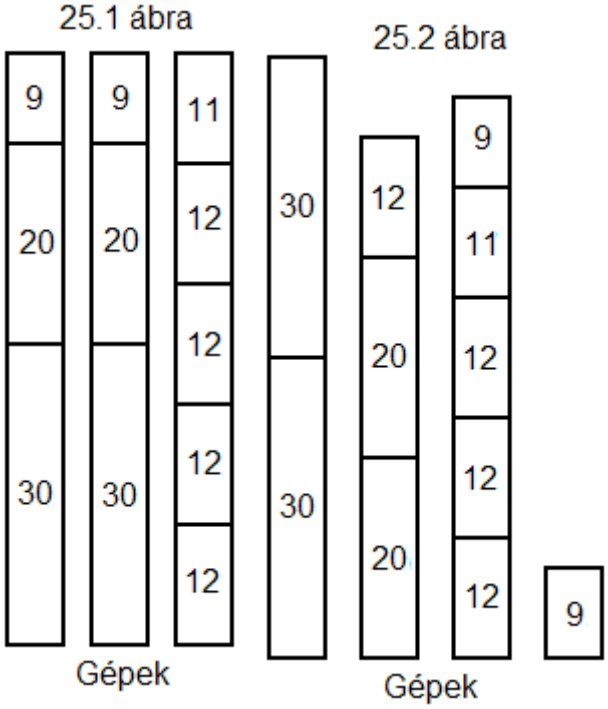
24.2 ábra

25. feladat

Hozzunk létre olyan feladatot, amikor az FFD algoritmus magasabb kapacitás mellett több gépet igényel mint ugyanazon dolgokra alacsonyabb kapacitás mellett.

A munkák listája:

$\tau=(18,11,10,5,5,5,5,5,5,5,5,5)$



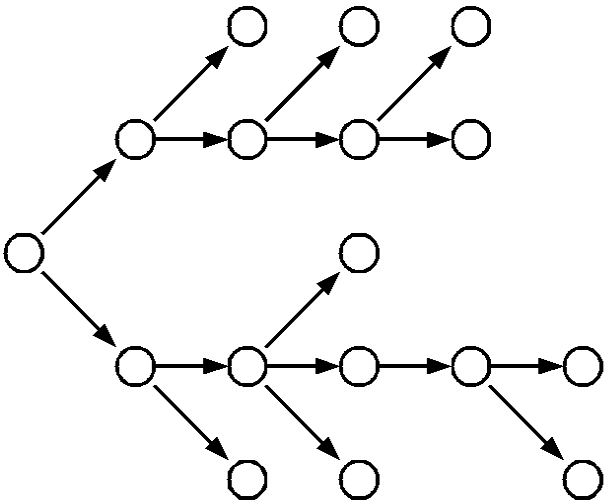
Megoldás:

25.1-es ábra: 59-es kapacitás

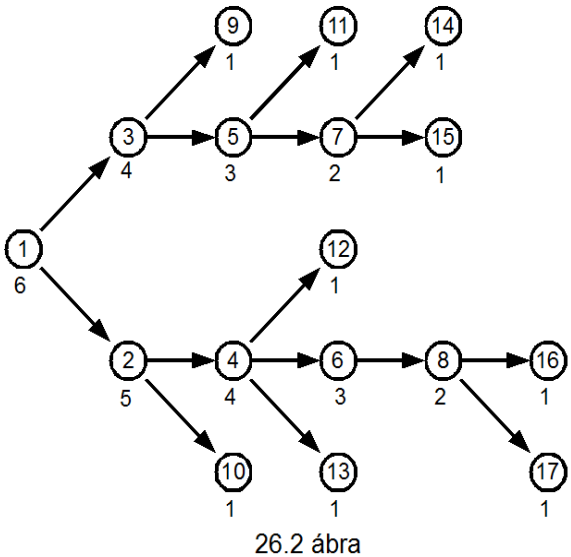
25.2-as ábra: 60-es kapacitás

26. feladat

Alkalmazzuk HU algoritmusát $P|tree,\tau_i=1|C_{max}$ *feladatra az 26.1-es ábrán látható irányított gráfon:*



26.1 ábra

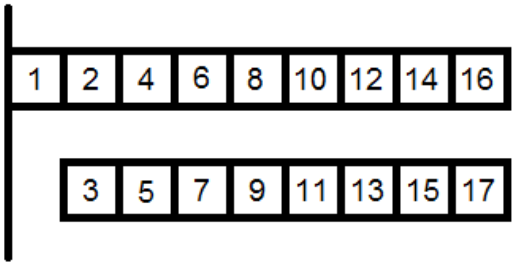


26.2 ábra

Első lépésként számozzuk meg az éleket aszerint, hogy mennyi a belőlük kivezető út (+1) majd számozzuk meg a csúcsokat először a legnagyobbakat, majd a számozás szerint csökkenő sorrendben az összes ponton menjünk végig, ez látható a 26.2 ábrán.

Második lépésként, mivel a feladat nem részletezi, hogy hány párhuzamos gépről van szó ezért először számoljuk végig 2 gépre a feladatot:

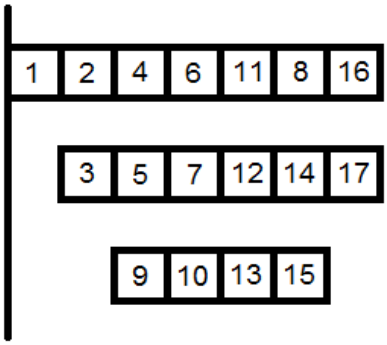
Ehhez mindössze annyit kell tennünk, hogy felhelyezzük az éppen szabad gépre az első szabad munkát vagyis ha a fenti gráfunkra úgy tekintünk mint folyamgráfra akkor a források közül rakjuk fel valamelyiket. Könnyű látni, hogy mindegy melyiket, hiszen a forrás-tulajdonság újabb pontok elvételével nem romolhat el. Ezek alapján a fenti feladat egy két gépes ütemezése a következő:



26.3 ábra

Könnyen látható hogy ez optimális hiszen kisebb C_{max} -ot a független munkák esetében se kaphatnánk.

Folytassuk 3 gép esetével:

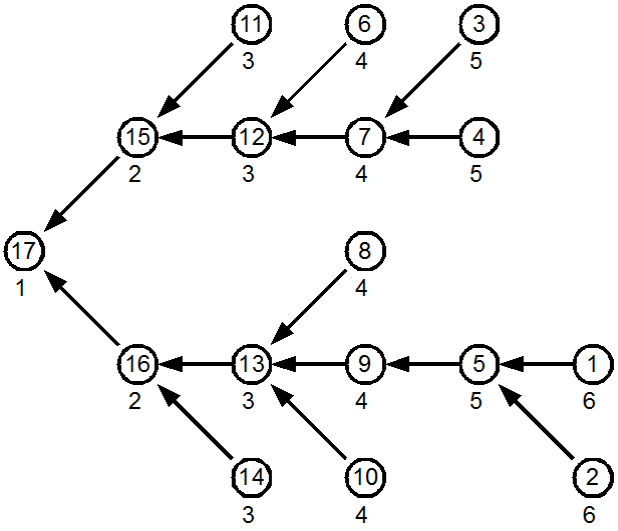


26.4 ábra

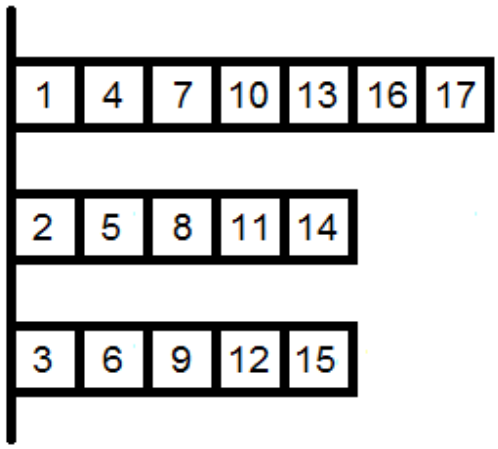
Itt is ugyanazt az algoritmust követtük mint az előbb, viszont ebben az esetben már nem tudjuk, hogy optimális megoldást kaptunk, hiszen HU algoritmusa elvileg csak befenyőkre működik. A független esetben is lenne jobb megoldás (például 6-6-5 munka elosztás). Ha közelebbről megvizsgáljuk kizárhatjuk, hogy bármelyik gépen az első pozícióba más munka is kerüljön az 1-es sorszámún kívül, hiszen nincs más forrás kezdetben vagyis közvetve minden munka előfeltétele az 1-es munka elkészülte.

Kevésbé heurisztikus megközelítést is alkalmazhatunk:

Vegyük észre hogy a megadott gráfban minden pont befoka 1 vagyis kifenyő. Fordítsuk meg az élek irányítását és újra alkalmazzuk HU algoritmusát.

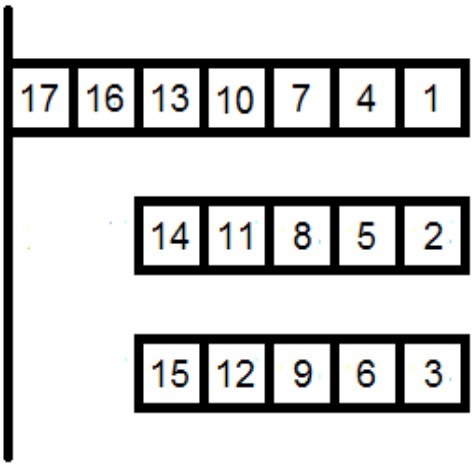


26.5 ábra

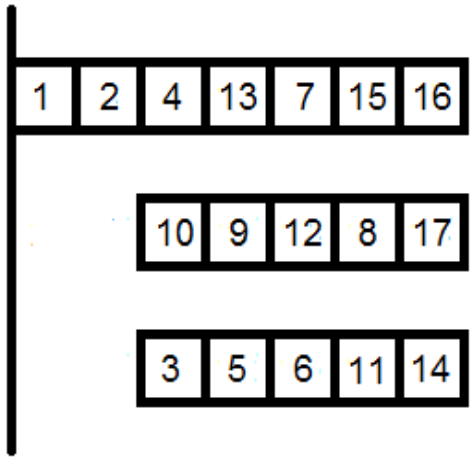


26.6 ábra

Ekkor az ütemezés a 26.6-os ábrán látható. Ha ezt tükrözzük akkor az eredeti feladatra kapunk egy optimális megoldást hiszen a C_{max} értéke nem változik a tükrözéstől és az irányítottság se romolhat el: ehhez az kellene, hogy egy munka és az előfeltétele legalább ugyanakkor legyenek ütemezve két gépen. Ez viszont ellentmond annak, hogy az új 26.5-ös ábrán látható gráfban is fennáll egy megelőzési feltétel köztük. Ha pedig a tükrözött ütemezésben előbb ütemezünk egy munkát mint előfeltételét az azt jelenti, hogy a 26.6-os ábrán látható ütemezésben követik helyesen egymást vagyis az eredeti megelőzési feltételnek tesznek eleget, ami ellentmond annak hogy minden megelőzési feltételt megfordítottunk.



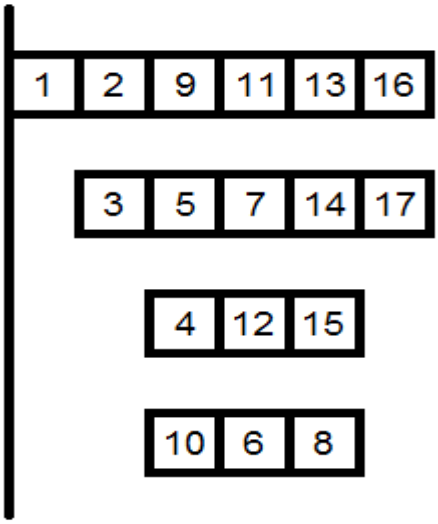
26.7 ábra



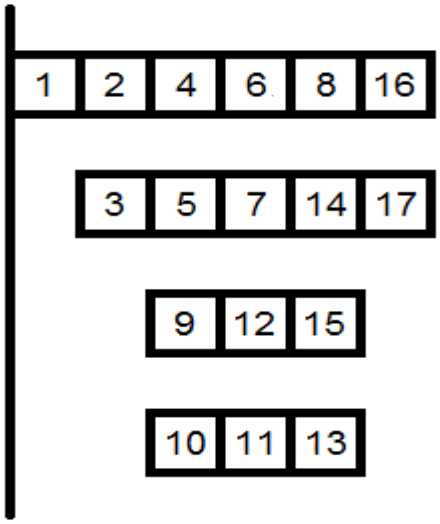
26.8 ábra

A 26.7-es ábrán látható a 26.6-os ábra tükörképe a 26.8-ason pedig az ennek megfelelő eredeti sorszámozás szerinti ütemezés. Így már látható hogy nincs jobb optimális megoldás és a minimális $C_{max}=7$.

Ezek után végezzük el 4 gépre is az ütemezést:



26.9 ábra

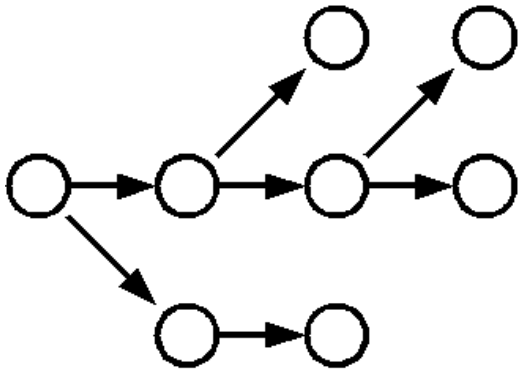


26.10 ábra

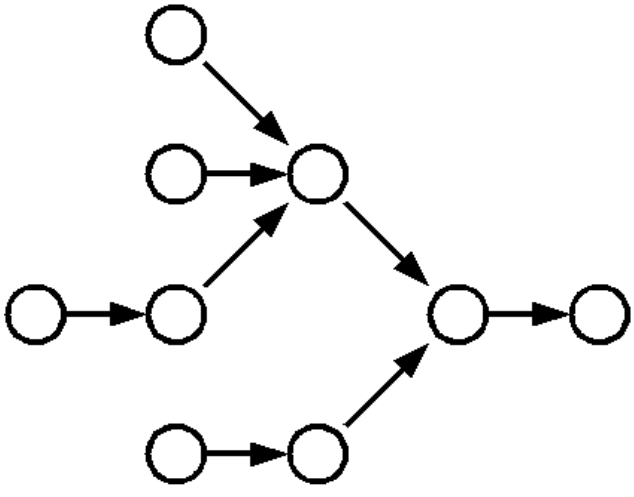
A 26.9-es ábrán látható az elérhető források szerinti listás ütemezés, a 10-es ábrán pedig ennek egy ekvivalens átalakítása a 26.10-es ábrán látható 1-2-4-6-8-16 csúcsok a gráfban egy irányított utat alkotnak. Ezért a C_{max} értéke nem csökkenthető tovább még további gépek hozzá vételével sem. Így ez a megoldás optimálisnak tekinthető az összes 4-nél nagyobb gépszámú esetre is.

27. feladat

Az alábbi ábrákon látható gráfokra alkalmazható e HU algoritmusa a $P|tree,\tau_i=1|C_{max}$ *problémára?*

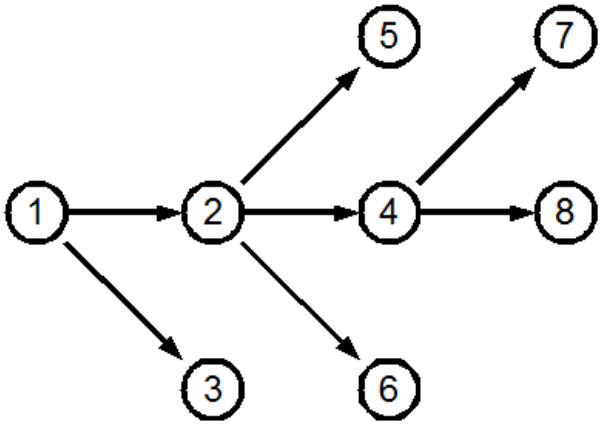


27.1 ábra



27.2 ábra

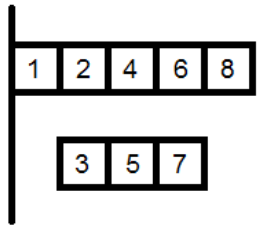
Azonnal látható hogy a 27.2-es ábrán látható gráf befenyő vagyis alkalmazható rá Hu algoritmusa míg az 27.1-es ábra kifenyő így, mint az előző feladatban elkészíthetjük befenyő párját amire már megoldható lesz, de először nézzük meg, hogy enélkül a lépés nélkül megoldható e a feladat. Készítsük el a pontok sorszámozását a nyelőktől való távolságuk szerint.



27.3 ábra

Első eset amikor csak két gép van:

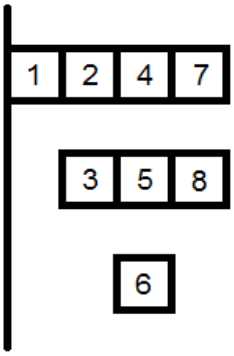
Úgy tűnhet, hogy ez nem optimális megoldás, hiszen ha sikerülne 4-4 munkát ütemezni mindkét gépre állásidő nélkül, akkor jobb megoldást kapnánk, de ha megnézzük az 27.1-es ábrán látható gráfot rögtön szembe tűnik, hogy csak egy forrás van. Vagyis az első pozícióba csak az a munka kerülhet, a többit csak utána ütemezhetjük. Így ez is optimális vagyis nem kell a befenyő párjára kiszámolni.



27.4 ábra

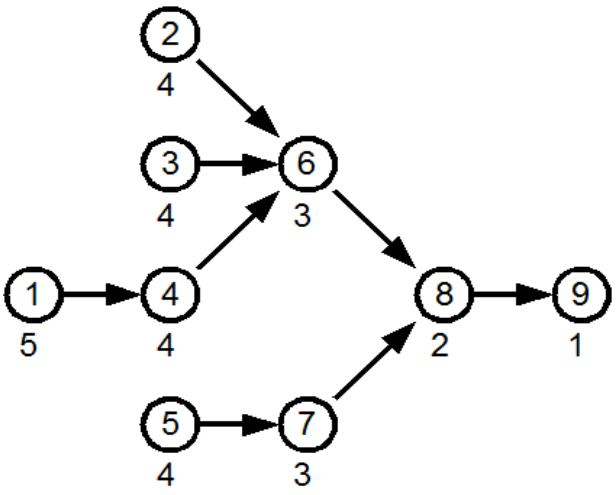
Második eset amikor 3 gép van:

Itt sem tűnik túl jó megoldásnak a kapott ütemezés viszont rögtön láthatjuk, hogy optimális hiszen az 1-2-4-7 csúcsokból álló út egy irányított út a gráfunkban, ezért ennél jobb ütemezést nem kaphatunk a gépek számának növelésével sem.



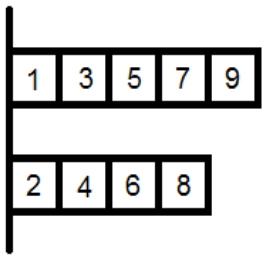
27.5 ábra

Vizsgáljuk meg most a 27.2-es ábrát amire tétel szerint alkalmazható Hu algoritmusa:

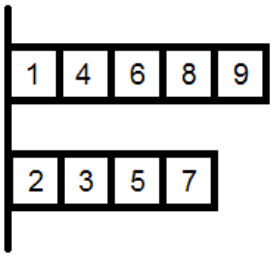


27.6 ábra

Itt az ütemezés két gép esetén a következő lesz: átrendezve pedig



27.7 ábra

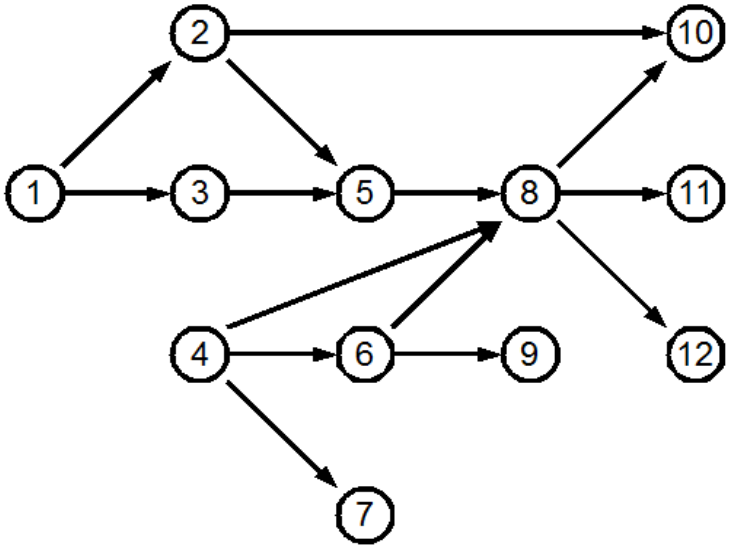


27.8 ábra

Az átrendezett ábrán látható 1-4-6-8-9 munkák a gráfban egy irányított utat alkotnak vagyis csak egymás után következhetnek. Így nincs értelme több gépre is megvizsgálni mindig ez lesz az optimális ütemezés.

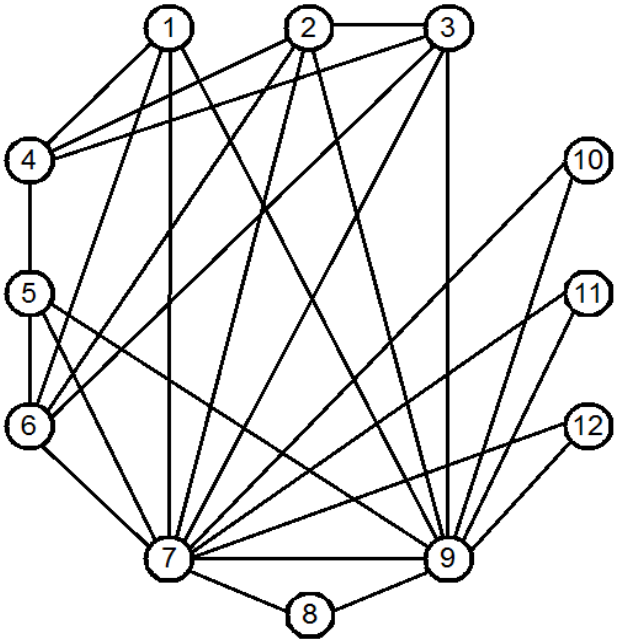
28. feladat

Alkalmazzuk Fujii-Kasami-Ninomiya két gépes esetre vonatkozó algoritmusát (FKN algoritmus) $P2|prec,\tau_i=1|C_{max}$ *feladatra az alábbi gráfon.*

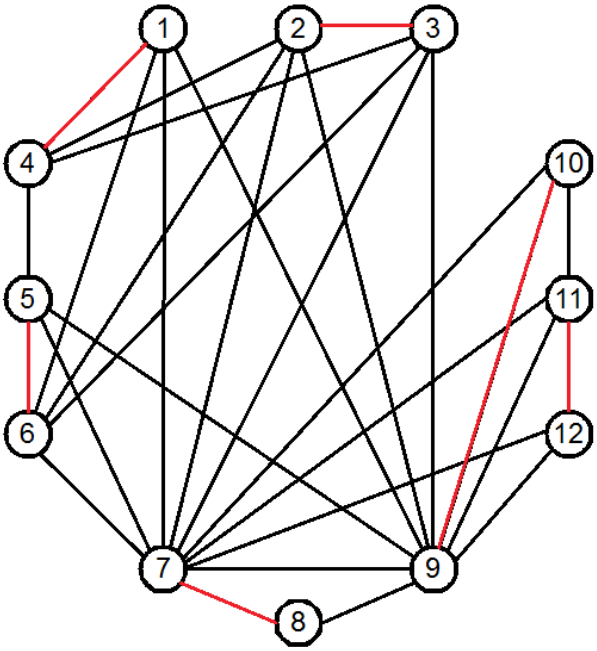


28.1 ábra

Készítsük el ebből azt a gráfot, amiben ugyanezek a pontok vannak és két pont között akkor megy él, ha az eredeti gráfban az egyikből sem tudunk eljutni a másikba (28.2-es ábra). Majd készítsük el a pontok egy maximális párosítását (28.3-as ábra). A teljes párosítást könnyű találni, ha a pontokon növekvő sorrendben megyünk végig és mindig a legkisebb sorszámú ponttal összekötő élét vesszük be a párosításba.



28.2 ábra



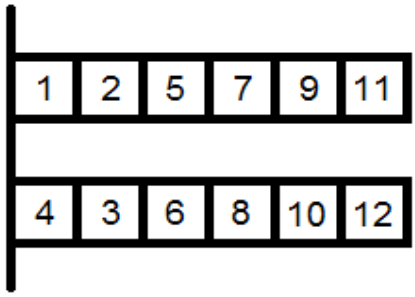
28.3 ábra

Az algoritmus szerint első lépésként vegyük be az ütemezésbe 1-4 párt mivel 1 és 4 is forrás a gráfunkban, ekkor már csak 2-3, 5-6, 7-8, 9-10, 11-12 él marad a párosításunkban.

Második lépésként vegyük be a 2-3 párt. Ezt szintén megtehetjük, mivel a maradék gráfban már 2,3,6,7 pontok is források.

Harmadik lépésként vegyük be 5-6 párt. Ezt szintén megtehetjük, mivel az előző lépés után az 5,6,7 pontok lettek források, most már 7,8,9 pontok a források, ezért 7-8 párt betudjuk venni az ütemezésbe.

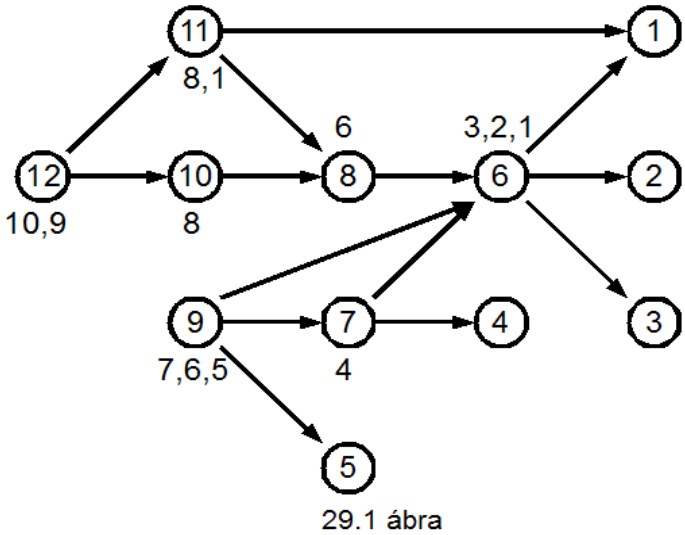
Ezzel elértünk oda, hogy az összes megmaradt pont 9,10,11,12 források (illetve izolált pontok) lettek vagyis mondjuk 9-10,11-12 párokban bevethetjük őket az ütemezésünkbe így a végső ütemezés a következőképpen fog kinézni:



28.4 ábra

29. feladat

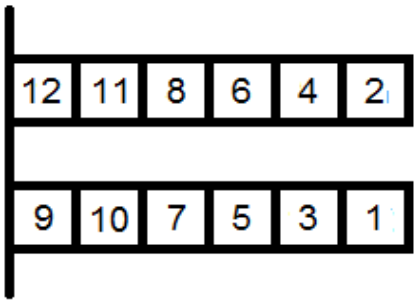
Alkalmazzuk Coffman – Graham két gépes esetre vonatkozó algoritmusát az előző példára vagyis $P2|prec, \tau_i = 1|C_{max}$ *problémára, az előző feladat gráfjára viszont most más sorszámozást kell alkalmazni az algoritmus szerint:*



29.1 ábra

Az ábrán látható sorszámozás, valamint címkézést úgy kaptuk, hogy elsőként megszámozzuk a nyelőket (1,2,3,4,5). Majd az ő szüleiket (vagyis azokat a pontokat, amikből csak a nyelőkbe vezetnek irányított élek) fel címkézzük aszerint, hogy mely nyelőkbe mutatnak, ezek a ({3,2,1}, {4}) címkéjű csúcsok lesznek. Mivel a {3,2,1} előbbre van lexikografikusan mint a {4} ezért az előbbi lesz a 6-os az utóbbi pedig a 7-es csúcs. Ezután megnézzük, hogy a már sorszámozott csúcsoknak mely csúcsok a szüleik. Ezek a ({6}{7,6,5}) címkéjű csúcsok lesznek és mivel a {6} előrébb van mint a {7,6,5} ezért a {6} címkéjű lesz 8-as sorszámú csúcs a {7,6,5} pedig a 9-es sorszámú. Ezt a lépést ismételgetve kapjuk, hogy a következő szinten a ({8}{8,1}) lesznek a szülők vagyis akkor a {8} címkéjű lesz a 10-sorszámú a {8,1}-es pedig a 11-es sorszámú. Mindezek után már csak egy csúcs marad így az értelemszerűen 12-es sorszámot kap valamint a {10,9}-es címkét.

Vagyis az ütemezés a következő lesz: Először vegyük be 12,9-es pontokat mert ezek a legnagyobb sorszámú források majd 11,10-es pontokat, mert a 12-est és 9-est kivéve ezek lesznek a legnagyobb sorszámú források és így tovább 8,7 aztán 6,5 majd 4,3 végül 2,1 sorszámú pontokat, ahogy azt a 29.2-es ábrán láthatjuk.



29.2 ábra

30. feladat

Adjunk példákat olyan esetekre amikor $P|prec|C_{max}$ *problémára a listás ütemezés nem ad optimális megoldást.*

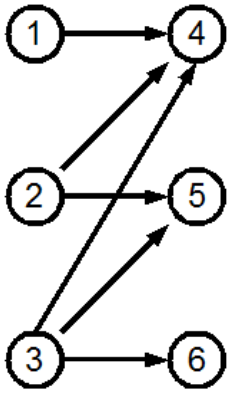
Két esetre veszünk példákat: - Az egyikben a munkák megmunkálási ideje egységnyi, viszont az egymástól nem függő munkák rossz ütemezési sorrendje miatt nem válik elérhetővé a gépek számának megfelelő munka ezért állás idő keletkezik. (1-es, 3-as példa)

- A másik esetben pedig olyan példákat, amikben egymástól nem függő, de eltérő megmunkálási idejű munkák felcserélhetőségéből adódik az optimum csökkentés lehetősége. (2-es,4-as példa)

1. példa

A munkák listája: $\tau=(1,1,1,1,1,1)$

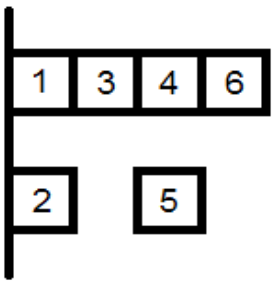
Megelőzési gráfja:



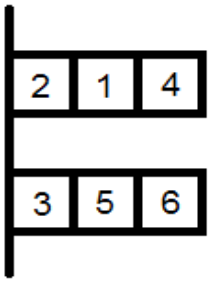
30.1 ábra

30.2 ábra: listás ütemezés

30.3 ábra: optimális ütemezés



30.2 ábra

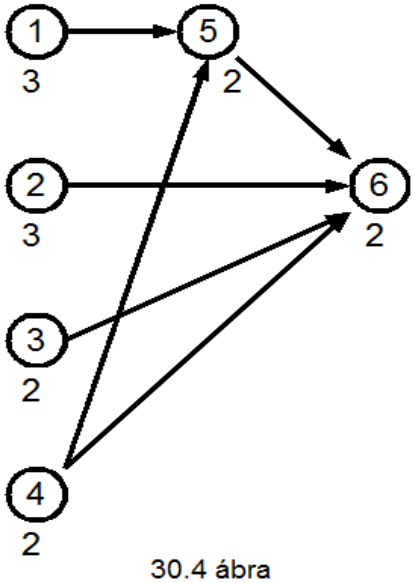


30.3 ábra

2. példa

A munkák listája: $\tau=(3,3,2,2,2,2)$

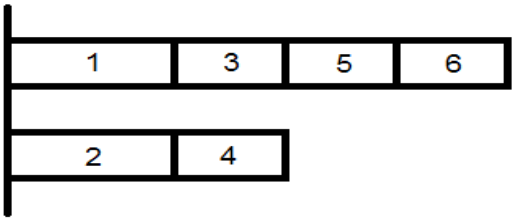
Megelőzési gráfja:



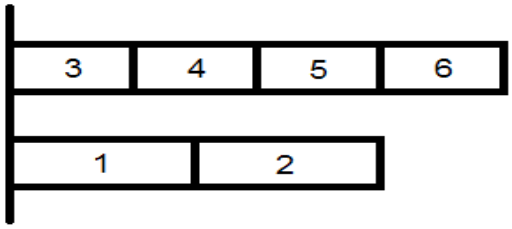
30.4 ábra

30.5-ös ábra: listás ütemezés

30.6-os ábra: optimális ütemezés



30.5 ábra

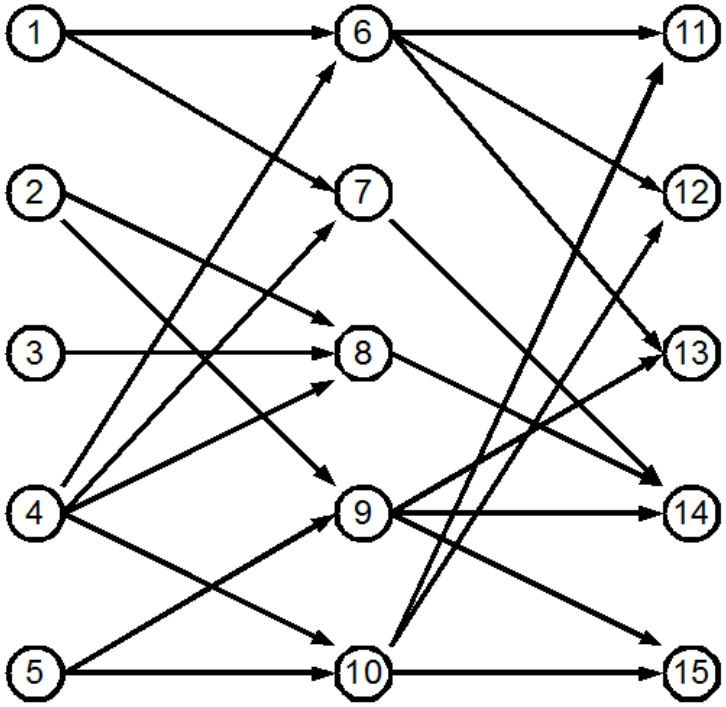


30.6 ábra

3. példa

A munkák listája: $\tau=(1,1,1,1,1,1,1,1,1,1,1,1,1,1,1)$

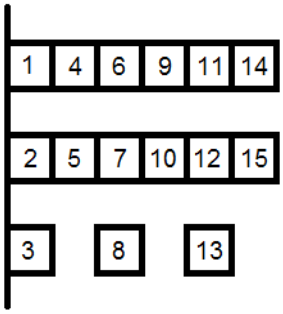
Megelőzési gráfja:



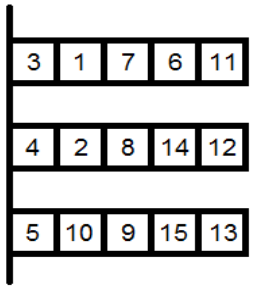
30.7 ábra

30.8-as ábra: listás ütemezés:

30.9-es ábra: optimális ütemezés:



30.8 ábra

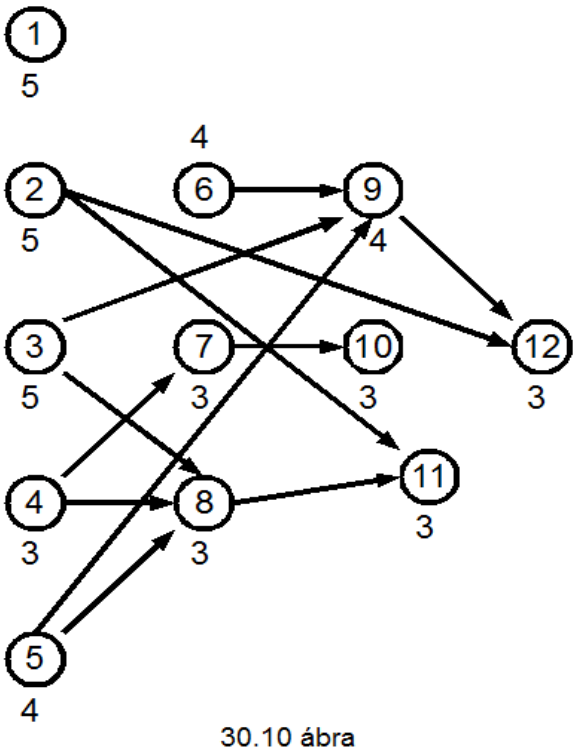


30.9 ábra

4. példa:

A munkák listája: $\tau=(5,5,5,3,4,4,3,3,4,3,3,3,2)$

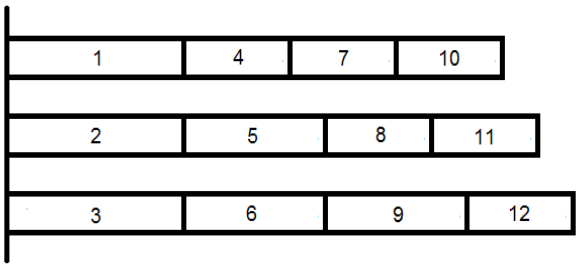
Megelőzési gráfja:



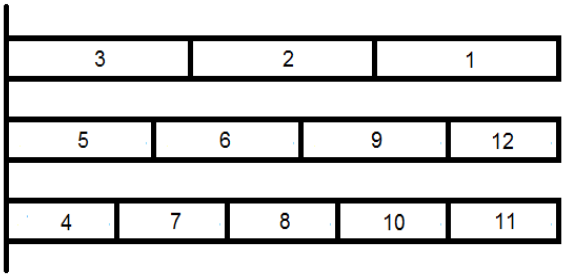
30.10 ábra

30.11-es ábra: listás ütemezés

30.12-es ábar: optimális ütemezés



30.11 ábra



30.12 ábra

Hivatkozások:

[1] R. GARY PARKER: Deterministic Scheduling Theory, Chapman and Hall, 1995.

[2] KATONA GYULA Y., RECSKI ANDRÁS, SZABÓ CSABA: A számítástudomány alapjai, Typotex

[3] RÓNYAI LAJOS, IVANYOS GÁBOR, SZABÓ RÉKA: Algoritmusok, Typotex (2005)