Haladó informatikai algoritmusok

Szerelőszalag

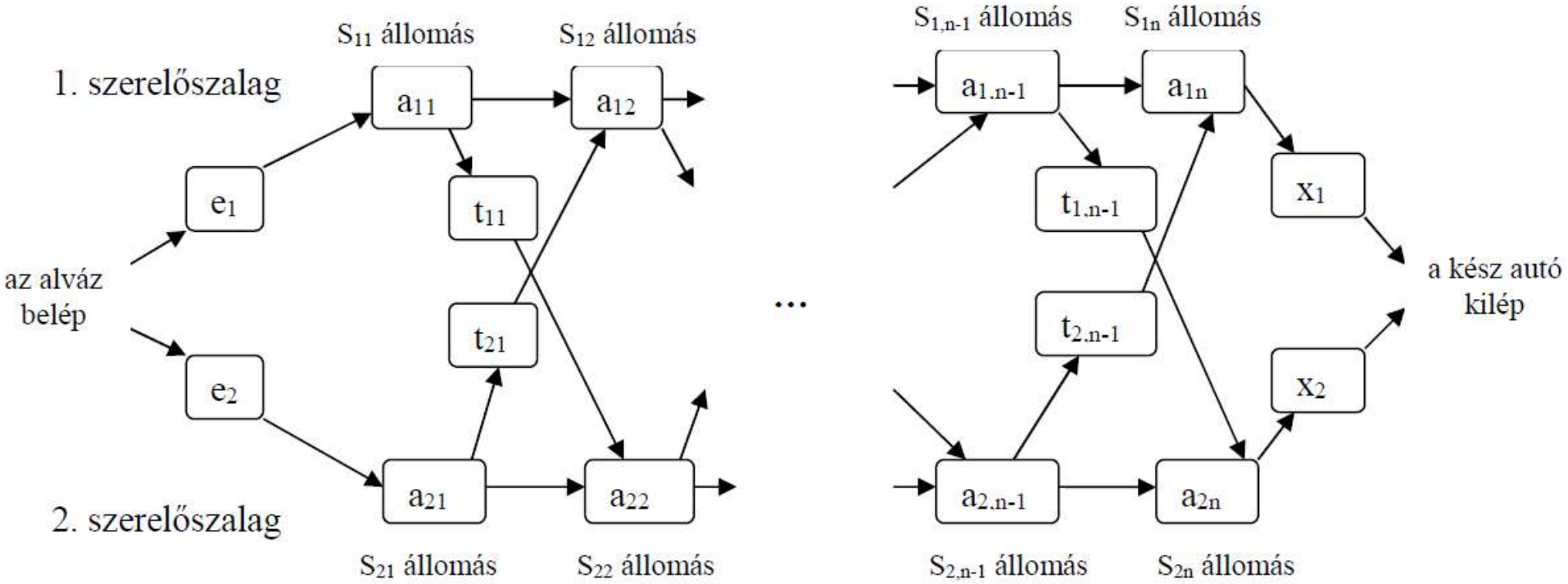
Dinamikus programozás jellemzői

Egy dinamikus programozási algoritmus kifejlesztése 4 lépésre bontható fel:

1. Jellemezzük az optimális megoldás szerkezetét.
2. Rekurzív módon definiáljuk az optimális megoldás értékét.
3. Kiszámítjuk az optimális megoldás értékét alulról felfelé történő módon.
4. A kiszámított információk alapján megszerkesztünk egy optimális megoldást.

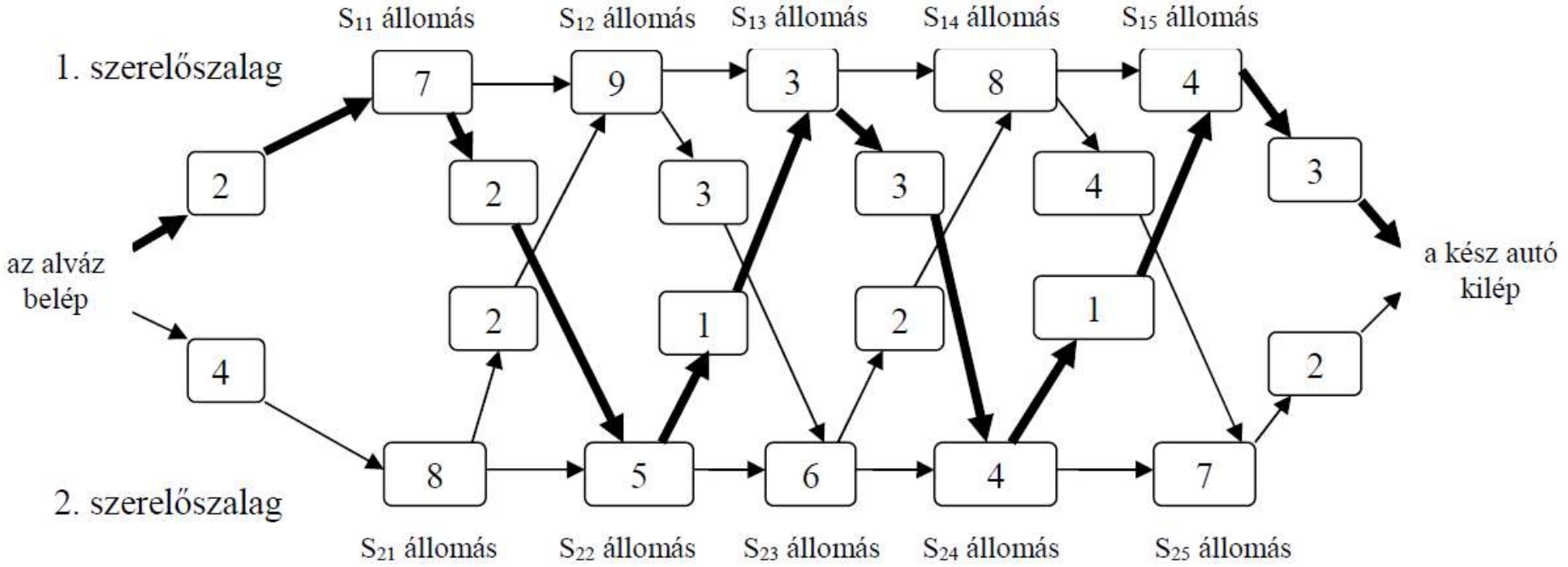
Feladat

Egy autógyár autókat állít elő az egyik gyárában, amelynek két szerelőszalagja van. Mindkét szerelőszalag elején egy alváz jelenik meg, melyhez adott számú állomáson hozzászerelik az alkatrészeket, majd a szalag végén távozik a kész autó.
Mindegyik szalagnak n állomása van, melyek indexei j=1, 2, …, n.
S_{ij} jelöli az i-edik (i=1 vagy 2) szalag j-edik állomását.
Az első szalag j-edik állomása (S_{1j}) ugyanazt a funkciót látja el, mint a második szalag j-edik állomása (S_{2j}).
A szalagokat különböző időpontokban különböző technológiával építették, ezért előfordulhat, hogy a két szalag azonos állomásán a terméknek különböző időt kell eltöltenie. Az S_{ij} állomáson a műveleti időt a_{ij} jelöli.



Feladat

Ahhoz is kell egy bizonyos idő, hogy az alváz a szalagra kerüljön, és ahhoz is, hogy az elkészült autó elhagyja a szalagot. Ezeket az i-edik szalag esetén e_i, illetve x_i jelöli.
Ha szeretnénk a megrendeléseket minél gyorsabban teljesíteni, a két szalag munkáját össze kell hangolni. Az i-edik szalag S_{ij} állomása után az alváznak a másik szalagra való átrakása t_{ij} időbe kerül $j=1, 2, \ldots, n-1$ (mivel n állomás után a kocsi már kész).
A feladat az, hogy meghatározzuk, mely állomásokat érintse az egyik és melyeket a másik szalagon, hogy az átfutási idő minimális legyen.



A teljes leszámlálás számítási ideje $\Omega(2^n)$. Ez nagy n mellett elfogadhatatlan.

1. lépés: a legrövidebb gyártási út

Tekintsük a lehetséges legrövidebb utat, ahogy egy alváz a kiindulási helyzetből túljut az S_{1j} állomáson.
Ha $j = 1$, akkor csak egy lehetőség van, és így könnyű meghatározni a szükséges időt.
$j = 2, \ldots, n$ esetén két lehetőség van:

- Az alváz érkezhet közvetlenül az $S_{1,j-1}$ állomásról, amikor ugyanannak a szalagnak a $(j-1)$-edik állomásáról a j-edik állomására való átszállítás ideje elhanyagolható.
- Az alváz $S_{2,j-1}$ felől érkezik, az átszállítás ideje $t_{2,j-1}$.

Tegyük fel, hogy az $S_1 j$ állomáson való túljutás leggyorsabb módja az, hogy oda az $S_{1,j-1}$ felől érkezünk.
Az alvázнak a legrövidebb módon kell túljutnia az $S_{1,j-1}$ állomáson, mert ha volna egy gyorsabb mód, hogy az alváz túljusson a $S_{1,j-1}$ állomáson, akkor ezt használva magán $S_{1,j}$-n is hamarabb jutna túl, ami **ellentmondás**.

1. lépés: a legrövidebb gyártási út

Általánosabban fogalmazva azt mondhatjuk, hogy a szerelőszalag ütemezésének
(hogy megtaláljuk a legrövidebb módot az S_{ij} állomáson való túljutásra)
optimális megoldása tartalmazza egy részfeladat
(vagy az $S_{1,j-1}$ vagy az $S_{2,j-1}$ állomáson való leggyorsabb áthaladás)
optimális megoldását.

Erre a tulajdonságra **optimális részstruktúra**ként fogunk hivatkozni.

Azaz részfeladatok optimális megoldásaiból megszerkeszthető a feladat optimális megoldása.

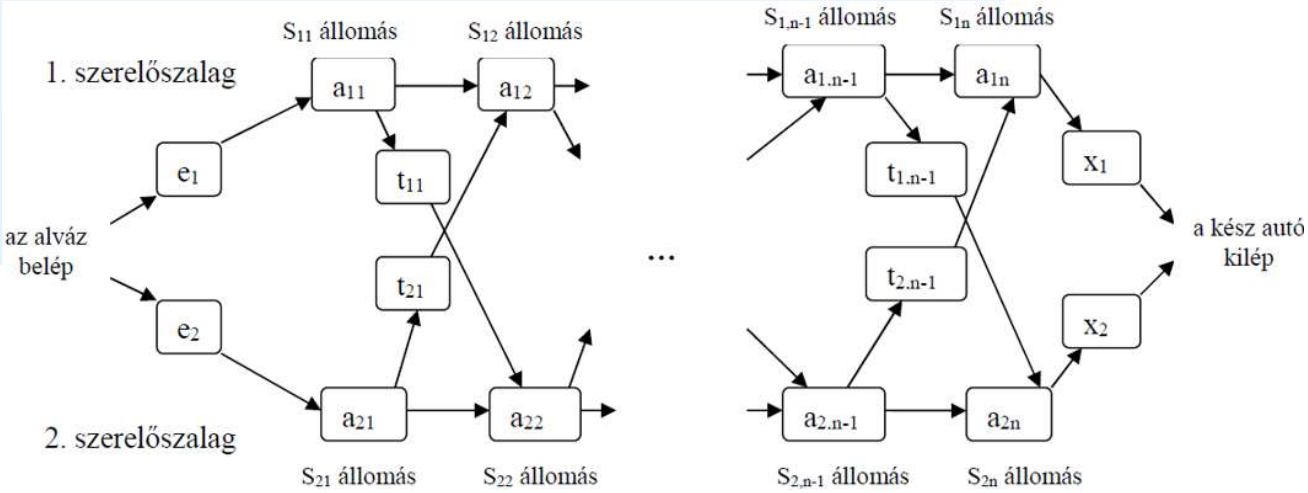
2. lépés: a rekurzív megoldás

Az optimális megoldás értékét a részfeladatok optimális értékeiből rekurzívan fejezzük ki.
A részfeladatok legyenek mindkét szalag j-edik állomásán való leggyorsabb áthaladás megkeresésének a feladatai $j = 1, \ldots, n$ mellett.
Jelölje $f_i[j]$ azt a legrövidebb időt, ami alatt az alváz túl tud jutni az S_{ij} állomáson.
Célunk annak a legrövidebb időnek a meghatározása, ami alatt az alváz az egész üzemen keresztül tud haladni.
Ezt az időt f^* jelöli.

$$f^* = min\{f_1[n] + x_1, f_2[n] + x_2\}$$

$$f_1[1] = e_1 + a_{11}$$

$$f_2[1] = e_2 + a_{21}$$

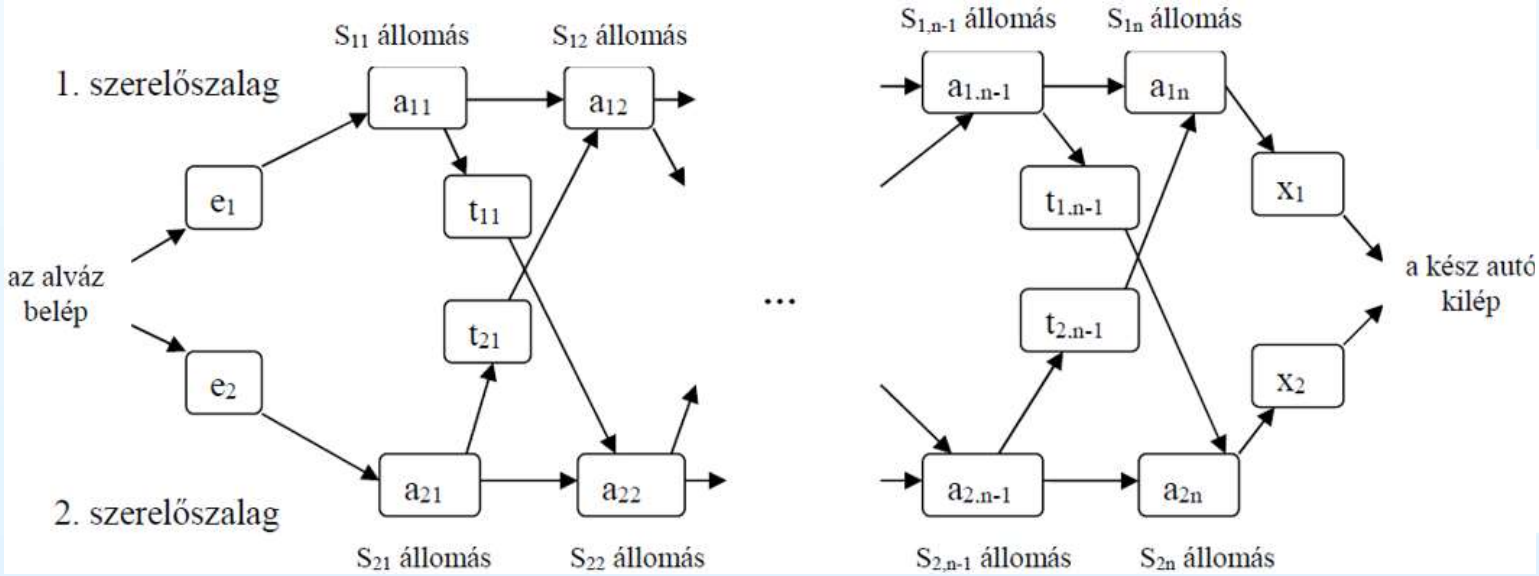


2. lépés: a rekurzív megoldás

Hogyan kell kiszámolni az $f_i[j]$-t $j = 2, \ldots, n$ és $i = 1, 2$ esetén?

$$f_1[j] = min\{f_1[j-1] + a_{1j}, f_2[j-1] + t_{2,j-1} + a_{1j}\}, \text{ ha } j = 2, \ldots, n.$$

$$f_2[j] = min\{f_2[j-1] + a_{2j}, f_1[j-1] + t_{1,j-1} + a_{2j}\}, \text{ ha } j = 2, \ldots, n.$$



2. lépés: a rekurzív megoldás

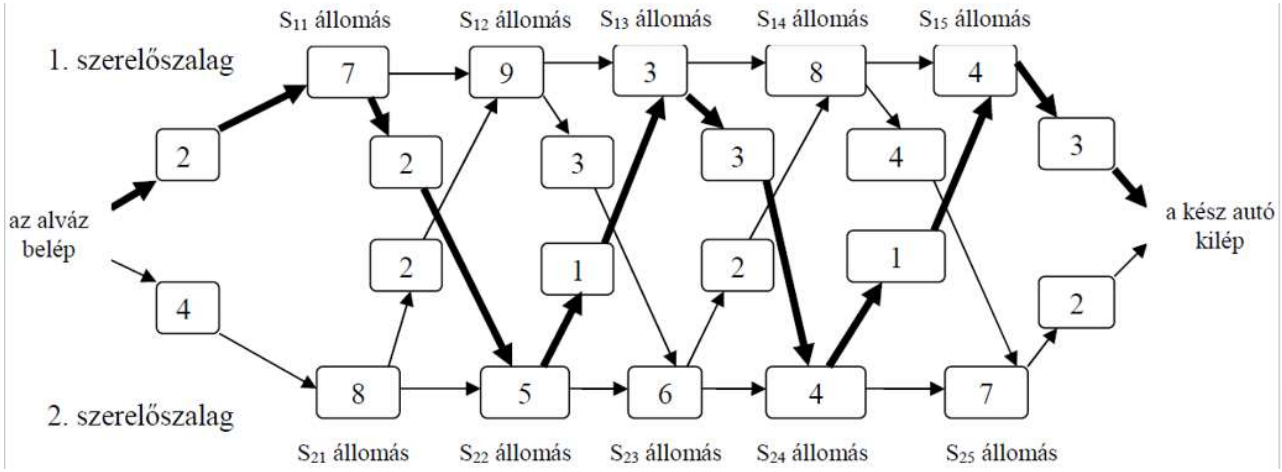
Az $f_i[j]$ mennyiségek a részfeladatok optimális érékei.

Annak érdekében, hogy vissza tudjuk keresni a legrövidebb utat, definiáljuk $l_i[j]$-t mint azt a szerelőszalagot, amelyiknek a $j-1$-edik állomását használtuk az S_{ij}-n való leggyorsabb keresztülhaladáskor.

Itt $i = 1, 2$ és $j = 2, \ldots, n$.
(Nem definiáljuk $l_i[1]$-et, mert S_{i1}-t egyik szalagon sem előzi meg másik állomás.)

Az l^* szalag definíció szerint az, amelyiknek az utolsó állomását használjuk az egész üzemen való áthaladáskor.
Az $l_i[j]$ értékek segítségével lehet nyomon követni a legrövidebb utat.

$$
\begin{aligned}
f_1[j] &= \begin{cases} e_1 + a_{11}, \text{ ha } j = 1, \\ min\{f_1[j-1] + a_{1j}, f_2[j-1] + t_{2,j-1} + a_{1j}\}, \text{ ha } j \geq 2 \end{cases} \\
f_2[j] &= \begin{cases} e_2 + a_{21}, \text{ ha } j = 1, \\ min\{f_2[j-1] + a_{2j}, f_1[j-1] + t_{1,j-1} + a_{2j}\}, \text{ ha } j \geq 2 \end{cases}
\end{aligned}
$$



j	1	2	3	4	5	
$f_1[j]$	2+7=9	1. 9+9=18 2. 12+2+9=23	18+3=21 16+1+3=20	20+8=28 22+2+8=32	28+4=32 26+1+4=31	31+3=34
$f_2[j]$	4+8=12	2. 12+5=17 1. 9+2+5=16	16+6=22 18+3+6=27	22+4=26 20+3+4=27	26+7=33 28+4+7=39	33+2=35

j	2	3	4	5
$l_1[j]$	1	2	1	2
$l_2[j]$	1	2	2	2

$l^*=1$

3. lépés: a legrövidebb átfutási idő kiszámítása

Sokkal jobban járunk, ha az $f_i[j]$ értékeket más sorrend szerint számoljuk, mint az a rekurzív módszerből adódik.

Vegyük észre, hogy $j \geq 2$ esetén $f_i[j]$ csak $f_1[j-1]$ -től és $f_2[j-1]$ -től függ.

Ha az $f_i[j]$ értékeket az állomások j indexének növekvő sorrendjében számoljuk, akkor a legrövidebb átfutási idő meghatározása $\Theta(n)$ ideig tart.

3. lépés: a legrövidebb átfutási idő kiszámítása - ALGORITMUS1(a,t,e,x,n)

1. $f_1[1] := e_1+a_{11}$
2. $f_2[1] := e_2+a_{21}$
3. for j:=2 to n
4. do if $f_1[j-1]+a_{1j} =< f_2[j-1]+ t_{2,j-1} +a_{1j}$
5. then $f_1[j]:= f_1[j-1]+a_{1j}$
6. $l_1[j]:=1$
7. else $f_1[j]:= f_2[j-1]+ t_{2,j-1} +a_{1j}$
8. $l_1[j]:=2$
9. if $f_2[j-1]+a_{2j} =< f_1[j-1]+ t_{1,j-1} +a_{2j}$
10. then $f_2[j]:= f_2[j-1]+a_{2j}$
11. $l_2[j]:=2$
12. else $f_2[j]:= f_1[j-1]+ t_{1,j-1} +a_{2j}$
13. $l_2[j]:=1$
14. if $f_1[n] +x_1 =< f_2[n] +x_2$
15. then $f^*=f_1[n] +x_1$
16. $l^*=1$
17. else $f^*=f_2[n] +x_2$
18. $l^*=2$

j	1	2	3	4	5
$f_1[j]$	9	18	20	28	31
$f_2[j]$	12	16	22	26	33

j	2	3	4	5
$l_1[j]$	1	2	1	2
$l_2[j]$	1	2	2	2

$l^*=1$

4. lépés: a legrövidebb átfutási idejű út

Az alábbi eljárás az út állomásait az indexek csökkenő sorrendjében nyomtatja ki.

```
ALGORITMUS2(l,n)
1.  i:= l*
2.  print i"-edik szalag" n,,-edik állomás"
3.  for j:= n downto 2
4.        do i:= li[j]
5.              print i"-edik szalag (" j-1,,)-edik állomás"
```

Eredmény a példánkban:

S_{15}, S_{24}, S_{23}, S_{22}, S_{11}

l*=1

j	2	3	4	5
$l_1[j]$	1	2	1	2
$l_2[j]$	1	2	2	2