12. előadás

Párhuzamosság

Párhuzamosítható algoritmusok, párhuzamosított algoritmusok analízise

Adatszerkezetek és algoritmusok előadás

2018. március 6.

Kósa Márk, Pánovics János,
Szathmáry László és Halász Gábor

Debreceni Egyetem
Informatikai Kar

Általános tudnivalók

Ajánlott irodalom:

- Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein:
  Új algoritmusok, Scolar Informatika, 2003.
- Donald E. Knuth: A számítógépprogramozás művészete 1. (Alapvető algoritmusok), Műszaki Könyvkiadó, 1994.
- Donald E. Knuth: A számítógépprogramozás művészete 3. (Keresés és rendezés), Műszaki Könyvkiadó, 1994.
- Seymour Lipschutz: Adatszerkezetek, Panem-McGraw-Hill, Budapest, 1993.
- Rónyai Lajos, Ivanyos Gábor, Szabó Réka: Algoritmusok, Typotex, Budapest, 2008.

Félév teljesítésének feltételei:

- Gyakorlati aláírás
  - 2 ZH
- Írásbeli vizsga, aminek az értékelésébe ...

További részletek:

`http://hallg.inf.unideb.hu/~halasz`

Mátrix-szorzás: Soros (egy-szálú) algoritmussal

„Legegyszerűbb” változat

```
procedure MATRIX_SZOR(A, B, C, n)
    for i ← 1 to n do
        for j ← 1 to n do
            C[i,j] ← 0
            for k ← 1 to n do
                C[i,j] ← C[i,j]+A[i,k]*B[k,j]
            end for
        end for
    end for
end procedure
```

Hány műveletet kell végrehajtani a sikeres számolás érdekében?

$$T(n) = \Theta(n^3)$$

Kósa Márk
Pánovics János
Szathmáry László
Halász Gábor

Mátrix-szorzás: Soros (egy-szálú) algoritmussal

„Legegyszerűbb" változat

```
procedure MATRIX_SZOR(A, B, C, n)
    for i ← 1 to n do
        for j ← 1 to n do
            C[i,j] ← 0
            for k ← 1 to n do
                C[i,j] ← C[i,j]+A[i,k]*B[k,j]
            end for
        end for
    end for
end procedure
```

Hány műveletet kell végrehajtani a sikeres számolás érdekében?

$$T(n) = \Theta(n^3)$$

Mátrix-szorzás: Soros (egy-szálú) algoritmussal

„Legegyszerűbb” változat

```
procedure MATRIX_SZOR(A, B, C, n)
  for i ← 1 to n do
    for j ← 1 to n do
      C[i,j] ← 0
      for k ← 1 to n do
        C[i,j] ← C[i,j]+A[i,k]*B[k,j]
      end for
    end for
  end for
end procedure
```

Hány műveletet kell végrehajtani a sikeres számolás érdekében?

$$T(n) = \Theta(n^3)$$

Mátrix-szorzás: Soros (egy-szálú) algoritmussal

„Egyszerű” Oszd-meg-és-uralkodj változat

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} & a_{16} & a_{17} & a_{18} \\ a_{21} & a_{22} & a_{23} & a_{24} & a_{25} & a_{26} & a_{27} & a_{28} \\ a_{31} & a_{32} & a_{33} & a_{34} & a_{35} & a_{36} & a_{37} & a_{38} \\ a_{41} & a_{42} & a_{43} & a_{44} & a_{45} & a_{46} & a_{47} & a_{48} \\ a_{51} & a_{52} & a_{53} & a_{54} & a_{55} & a_{56} & a_{57} & a_{58} \\ a_{61} & a_{62} & a_{63} & a_{64} & a_{65} & a_{66} & a_{67} & a_{68} \\ a_{71} & a_{72} & a_{73} & a_{74} & a_{75} & a_{76} & a_{77} & a_{78} \\ a_{81} & a_{82} & a_{83} & a_{84} & a_{85} & a_{86} & a_{87} & a_{88} \end{pmatrix}$$

$$\cdots A_{12} = \begin{pmatrix} a_{15} & a_{16} & a_{17} & a_{18} \\ a_{25} & a_{26} & a_{27} & a_{28} \\ a_{35} & a_{36} & a_{37} & a_{38} \\ a_{45} & a_{46} & a_{47} & a_{48} \end{pmatrix} \cdots$$

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix};\ B = \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix};\ C = \begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix}$$

$$C = A \cdot B = \begin{pmatrix} A_{11} \cdot B_{11} + A_{12} \cdot B_{21} & A_{11} \cdot B_{12} + A_{12} \cdot B_{22} \\ A_{21} \cdot B_{11} + A_{22} \cdot B_{21} & A_{21} \cdot B_{12} + A_{22} \cdot B_{22} \end{pmatrix};$$

Kósa Márk
Pánovics János
Szathmáry László
Halász Gábor

Mátrix-szorzás: Soros (egy-szálú) algoritmussal

„Egyszerű” Oszd-meg-és-uralkodj változat

```
procedure M_S(A, B, C, n)
  if n=1 then
      c11 ← c11 + a11 · b11
      return
  end if
  n ← n/2
  M_S(A11, B11, C11, n)
  M_S(A12, B21, C11, n)
  M_S(A11, B12, C12, n)
  M_S(A12, B22, C12, n)
  M_S(A21, B11, C21, n)
  M_S(A22, B21, C21, n)
  M_S(A21, B12, C22, n)
  M_S(A22, B22, C22, n)
end procedure
```

C ← C+A·B

$n \neq 2^k$?

Műveletek száma?

$$T(1) = \Theta(1)$$

$$T(n) = 8T(\frac{n}{2}) + \Theta(1)$$

$$T(n) = \Theta(n^3)$$

$$T(1) = \Theta(1)$$

$$T(n) = 8T(\frac{n}{2}) + \Theta(n^2)$$

Mátrix-szorzás: Soros (egy-szálú) algoritmussal

„Egyszerű” Oszd-meg-és-uralkodj változat

procedure M_S(A, B, C, n)

- **if** n=1 **then**
  - $c_{11} \leftarrow c_{11} + a_{11} \cdot b_{11}$
  - **return**
- **end if**
- n $\leftarrow$ n/2
- M_S(A_{11}, B_{11}, C_{11}, n)
- M_S(A_{12}, B_{21}, C_{11}, n)
- M_S(A_{11}, B_{12}, C_{12}, n)
- M_S(A_{12}, B_{22}, C_{12}, n)
- M_S(A_{21}, B_{11}, C_{21}, n)
- M_S(A_{22}, B_{21}, C_{21}, n)
- M_S(A_{21}, B_{12}, C_{22}, n)
- M_S(A_{22}, B_{22}, C_{22}, n)

end procedure

C $\leftarrow$ C+A·B

$n \neq 2^k$?

Műveletek száma?

$$T(1) = \Theta(1)$$

$$T(n) = 8T(\frac{n}{2}) + \Theta(1)$$

$$T(n) = \Theta(n^3)$$

$$T(1) = \Theta(1)$$

$$T(n) = 8T(\frac{n}{2}) + \Theta(n^2)$$

Kósa Márk
Pánovics János
Szathmáry László
Halász Gábor

Mátrix-szorzás: Soros (egy-szálú) algoritmussal

„Egyszerű” Oszd-meg-és-uralkodj változat

procedure M_S(A, B, C, n)
  if n=1 **then**
    $c_{11} \leftarrow c_{11} + a_{11} \cdot b_{11}$
    return
  end if
  n ← n/2
  M_S(A_{11}, B_{11}, C_{11}, n)
  M_S(A_{12}, B_{21}, C_{11}, n)
  M_S(A_{11}, B_{12}, C_{12}, n)
  M_S(A_{12}, B_{22}, C_{12}, n)
  M_S(A_{21}, B_{11}, C_{21}, n)
  M_S(A_{22}, B_{21}, C_{21}, n)
  M_S(A_{21}, B_{12}, C_{22}, n)
  M_S(A_{22}, B_{22}, C_{22}, n)
end procedure

C ← C+A·B
$n \neq 2^k$?
Műveletek száma?

$T(1) = \Theta(1)$

$T(n) = 8T(\frac{n}{2}) + \Theta(1)$

$T(n) = \Theta(n^3)$

$T(1) = \Theta(1)$

$T(n) = 8T(\frac{n}{2}) + \Theta(n^2)$

Kósa Márk
Pánovics János
Szathmáry László
Halász Gábor

Mátrix-szorzás: Soros (egy-szálú) algoritmussal

„Egyszerű” Oszd-meg-és-uralkodj változat

```
procedure M_S(A, B, C, n)
  if n=1 then
      c11 ← c11 + a11 · b11
      return
  end if
  n ← n/2
  M_S(A11, B11, C11, n)
  M_S(A12, B21, C11, n)
  M_S(A11, B12, C12, n)
  M_S(A12, B22, C12, n)
  M_S(A21, B11, C21, n)
  M_S(A22, B21, C21, n)
  M_S(A21, B12, C22, n)
  M_S(A22, B22, C22, n)
end procedure
```

C ← C+A·B

$n \neq 2^k$?

Műveletek száma?

$$T(1) = \Theta(1)$$

$$T(n) = 8T(\frac{n}{2}) + \Theta(1)$$

$$T(n) = \Theta(n^3)$$

$$T(1) = \Theta(1)$$

$$T(n) = 8T(\frac{n}{2}) + \Theta(n^2)$$

Mátrix-szorzás: Soros (egy-szálú) algoritmussal

„Egyszerű” Oszd-meg-és-uralkodj változat

```
procedure M_S(A, B, C, n)
  if n=1 then
      c11 ← c11 + a11 · b11
      return
  end if
  n ← n/2
  M_S(A11, B11, C11, n)
  M_S(A12, B21, C11, n)
  M_S(A11, B12, C12, n)
  M_S(A12, B22, C12, n)
  M_S(A21, B11, C21, n)
  M_S(A22, B21, C21, n)
  M_S(A21, B12, C22, n)
  M_S(A22, B22, C22, n)
end procedure
```

C ← C+A·B

$n \neq 2^k$?

Műveletek száma?

$$T(1) = \Theta(1)$$

$$T(n) = 8T(\frac{n}{2}) + \Theta(1)$$

$$T(n) = \Theta(n^3)$$

$$T(1) = \Theta(1)$$

$$T(n) = 8T(\frac{n}{2}) + \Theta(n^2)$$

Mátrix-szorzás: Soros (egy-szálú) algoritmussal

„Egyszerű” Oszd-meg-és-uralkodj változat

procedure M_S(A, B, C, n)
  if n=1 **then**
    $c_{11} \leftarrow c_{11} + a_{11} \cdot b_{11}$
    return
  end if
  n $\leftarrow$ n/2
  M_S(A_{11}, B_{11}, C_{11}, n)
  M_S(A_{12}, B_{21}, C_{11}, n)
  M_S(A_{11}, B_{12}, C_{12}, n)
  M_S(A_{12}, B_{22}, C_{12}, n)
  M_S(A_{21}, B_{11}, C_{21}, n)
  M_S(A_{22}, B_{21}, C_{21}, n)
  M_S(A_{21}, B_{12}, C_{22}, n)
  M_S(A_{22}, B_{22}, C_{22}, n)
end procedure

C $\leftarrow$ C+A·B
$n \neq 2^k$?
Műveletek száma?

$$T(1) = \Theta(1)$$
$$T(n) = 8T(\frac{n}{2}) + \Theta(1)$$

$$T(n) = \Theta(n^3)$$

$$T(1) = \Theta(1)$$
$$T(n) = 8T(\frac{n}{2}) + \Theta(n^2)$$

Kósa Márk
Pánovics János
Szathmáry László
Halász Gábor

Mátrix-szorzás: Soros (egy-szálú) algoritmussal

„Egyszerű” Oszd-meg-és-uralkodj változat

```
procedure M_S(A, a, x, B, b, y, C, c, z, n)
  if n=1 then
      C[c,z] ←C[c,z]+A[a,x]*B[b,y]
      return
  end if
  n ← n/2
  M_S(A, a, x, B, b, y, C, c, z, n)
  M_S(A, a, x+n, B, b+n, y, C, c, z, n)
  M_S(A, a, x, B, b, y+n, C, c, z+n, n)
  M_S(A, a, x+n, B, b+n, y+n, C, c, z+n, n)
  M_S(A, a+n, x, B, b, y, C, c+n, z, n)
  M_S(A, a+n, x+n, B, b+n, y, C, c+n, z, n)
  M_S(A, a+n, x, B, b, y+n, C, c+n, z+n, n)
  M_S(A, a+n, x+n, B, b+n, y+n, C, c+n, z+n, n)
end procedure
```


Kósa Márk
Pánovics János
Szathmáry László
Halász Gábor

Mátrix-szorzás: Soros (egy-szálú) algoritmussal
Strassen algoritmusa

1. Osszuk fel az A és B mátrixokat $\frac{n}{2} \times \frac{n}{2}$ méretű részmátrixokra.
2. Hozzunk létre 10 darab, szintén $\frac{n}{2} \times \frac{n}{2}$ méretű új mátrixot az előbb definiált részmátrixokkal meghatározva:
   $S_1 = B_{12} - B_{22}$, $S_2 = A_{11} + A_{12}$, $S_3 = A_{21} + A_{22}$,
   $S_4 = B_{21} - B_{11}$, $S_5 = A_{11} + A_{22}$, $S_6 = B_{11} + B_{22}$,
   $S_7 = A_{12} - A_{22}$, $S_8 = B_{21} + B_{22}$, $S_9 = A_{11} - A_{21}$,
   $S_{10} = B_{11} + B_{12}$.
3. Ezek felhasználásával számoljunk ki újabb 7 mátrixot:
   $P_1 = A_{11} \cdot S_1$, $P_2 = S_2 \cdot B_{22}$, $P_3 = S_3 \cdot B_{11}$, $P_4 = A_{22} \cdot S_4$,
   $P_5 = S_5 \cdot S_6$, $P_6 = S_7 \cdot S_8$, $P_7 = S_9 \cdot S_{10}$.
4. És végül határozzuk meg a C (szintén $\frac{n}{2} \times \frac{n}{2}$ méretű) részmátrixait:
   $C_{11} = P_5 + P_4 - P_2 + P_6$, $C_{12} = P_1 + P_2$,
   $C_{21} = P_3 + P_4$, $C_{22} = P_5 + P_1 - P_3 - P_7$.

Kósa Márk
Pánovics János
Szathmáry László
Halász Gábor

Mátrix-szorzás: Soros (egy-szálú) algoritmussal
Strassen algoritmusa

- Ami biztos:
  - Ez „sokkal” bonyolultabb, mint az előzőek
- Amit nem nyilvánvaló
  - Helyes ez egyáltalán?
  - Megéri a bonyodalmat?
    Azaz hatékonyabb, mint a korábbi algoritmusok?

$S(n) = 7S(n/2) + \Theta(n^2)$
$S(n) = \Theta(n^{\lg 7})$

Kósa Márk
Pánovics János
Szathmáry László
Halász Gábor

Mátrix-szorzás: Soros (egy-szálú) algoritmussal
Strassen algoritmusa

- Ami biztos:
  - Ez „sokkal” bonyolultabb, mint az előzőek
- Amit nem nyilvánvaló
  - Helyes ez egyáltalán?
  - Megéri a bonyodalmat?
    Azaz hatékonyabb, mint a korábbi algoritmusok?

$$S(n) = 7S(n/2) + \Theta(n^2)$$

$$S(n) = \Theta(n^{\lg 7})$$

Kósa Márk
Pánovics János
Szathmáry László
Halász Gábor

Mátrix-szorzás: Soros (egy-szálú) algoritmussal
Strassen algoritmusa

- Ami biztos:
  - Ez „sokkal” bonyolultabb, mint az előzőek
- Amit nem nyilvánvaló
  - Helyes ez egyáltalán?
  - Megéri a bonyodalmat?
    Azaz hatékonyabb, mint a korábbi algoritmusok?

$$S(n) = 7S(n/2) + \Theta(n^2)$$
$$S(n) = \Theta(n^{\lg 7})$$

Párhuzamos algoritmus
„Legegyszerűbb” változat

```
procedure MATRIX_SZOR(A, B, C, n)
1: parallel for i ← 1 to n do
2:     parallel for j ← 1 to n do
3:         C[i,j] ← 0
4:         for k ← 1 to n do
5:             C[i,j] ← C[i,j]+A[i,k]*B[k,j]
6:         end for
7:     end for
8: end for
end procedure
```

- work: $T_1(n) = \Theta(n^3)$
- span: $T_\infty(n) = \Theta(n)$ $(2\Theta(\lg n) + \Theta(n))$
- párhuzamosság: $\Theta(n^2)$
- Ez tovább javítható $\Theta(n^3/\lg n)$-re, ha párhuzamosítjuk a 4.–6. sor ciklusát. DE ...

Párhuzamos algoritmus

„Legegyszerűbb” változat

```
procedure MATRIX_SZOR(A, B, C, n)
  1: parallel for i ← 1 to n do
  2:     parallel for j ← 1 to n do
  3:         C[i,j] ← 0
  4:         for k ← 1 to n do
  5:             C[i,j] ← C[i,j]+A[i,k]*B[k,j]
  6:         end for
  7:     end for
  8: end for
end procedure
```

- work: $T_1(n) = \Theta(n^3)$
- span: $T_\infty(n) = \Theta(n)$ $(2\Theta(\lg n) + \Theta(n))$
- párhuzamosság: $\Theta(n^2)$
- Ez tovább javítható $\Theta(n^3 / \lg n)$-re, ha párhuzamosítjuk a 4.–6. sor ciklusát. DE ...

Párhuzamos algoritmus

„Legegyszerűbb” változat

procedure MATRIX_SZOR(A, B, C, n)

```
1: parallel for i ← 1 to n do
2:     parallel for j ← 1 to n do
3:         C[i,j] ← 0
4:         for k ← 1 to n do
5:             C[i,j] ← C[i,j]+A[i,k]*B[k,j]
6:         end for
7:     end for
8: end for
end procedure
```

- work: $T_1(n) = \Theta(n^3)$
- span: $T_\infty(n) = \Theta(n)$ $(2\Theta(\lg n) + \Theta(n))$
- párhuzamosság: $\Theta(n^2)$
- Ez tovább javítható $\Theta(n^3 / \lg n)$-re, ha párhuzamosítjuk a 4.–6. sor ciklusát. DE ...

Párhuzamos algoritmus

„Legegyszerűbb” változat

```
procedure MATRIX_SZOR(A, B, C, n)
1: parallel for i ← 1 to n do
2:     parallel for j ← 1 to n do
3:         C[i,j] ← 0
4:         for k ← 1 to n do
5:             C[i,j] ← C[i,j]+A[i,k]*B[k,j]
6:         end for
7:     end for
8: end for
end procedure
```

- work: $T_1(n) = \Theta(n^3)$
- span: $T_\infty(n) = \Theta(n)$ $(2\Theta(\lg n) + \Theta(n))$
- párhuzamosság: $\Theta(n^2)$
- Ez tovább javítható $\Theta(n^3/\lg n)$-re, ha párhuzamosítjuk a 4.–6. sor ciklusát. DE ...

Párhuzamos algoritmus

„Legegyszerűbb” változat

```
procedure MATRIX_SZOR(A, B, C, n)
1: parallel for i ← 1 to n do
2:     parallel for j ← 1 to n do
3:         C[i,j] ← 0
4:         for k ← 1 to n do
5:             C[i,j] ← C[i,j]+A[i,k]*B[k,j]
6:         end for
7:     end for
8: end for
end procedure
```

- work: $T_1(n) = \Theta(n^3)$
- span: $T_\infty(n) = \Theta(n)$ $(2\Theta(\lg n) + \Theta(n))$
- párhuzamosság: $\Theta(n^2)$
- Ez tovább javítható $\Theta(n^3/\lg n)$-re, ha párhuzamosítjuk a 4.–6. sor ciklusát. DE ...

Kósa Márk
Pánovics János
Szathmáry László
Halász Gábor

Párhuzamos algoritmus

„Egyszerű” Oszd-meg-és-uralkodj változat

procedure M_S(A, B, C, n)
- **if** n=1 **then**
  - $c_{11} \leftarrow c_{11} + a_{11} \cdot b_{11}$
- **else**
  - n $\leftarrow$ n/2
  - **spawn** M_S(A_{11}, B_{11}, C_{11}, n)
  - **spawn** M_S(A_{12}, B_{21}, C_{11}, n)
  - **spawn** M_S(A_{11}, B_{12}, C_{12}, n)
  - **spawn** M_S(A_{12}, B_{22}, C_{12}, n)
  - **spawn** M_S(A_{21}, B_{11}, C_{21}, n)
  - **spawn** M_S(A_{22}, B_{21}, C_{21}, n)
  - **spawn** M_S(A_{21}, B_{12}, C_{22}, n)
  - M_S(A_{22}, B_{22}, C_{22}, n)
  - **sync**
- **end if**

end procedure

$C \leftarrow C + A \cdot B$

Helyes ez?

Sajnos nem: versenyhelyzet

Kósa Márk
Pánovics János
Szathmáry László
Halász Gábor

Párhuzamos algoritmus

„Egyszerű” Oszd-meg-és-uralkodj változat

```
procedure M_S(A, B, C, n)
  if n=1 then
      c11 ← c11 + a11 · b11
  else
      n ← n/2
      spawn M_S(A11, B11, C11, n)
      spawn M_S(A12, B21, C11, n)
      spawn M_S(A11, B12, C12, n)
      spawn M_S(A12, B22, C12, n)
      spawn M_S(A21, B11, C21, n)
      spawn M_S(A22, B21, C21, n)
      spawn M_S(A21, B12, C22, n)
      M_S(A22, B22, C22, n)
      sync
  end if
end procedure
```

$C \leftarrow C + A \cdot B$

Helyes ez?

Sajnos nem: versenyhelyzet

Kósa Márk
Pánovics János
Szathmáry László
Halász Gábor

Párhuzamos algoritmus

„Egyszerű” Oszd-meg-és-uralkodj változat

```
procedure M_S(A, B, C, n)
  if n=1 then
      c11 ← c11 + a11 · b11
  else
      n ← n/2
      spawn M_S(A11, B11, C11, n)
      spawn M_S(A12, B21, C11, n)
      spawn M_S(A11, B12, C12, n)
      spawn M_S(A12, B22, C12, n)
      spawn M_S(A21, B11, C21, n)
      spawn M_S(A22, B21, C21, n)
      spawn M_S(A21, B12, C22, n)
      M_S(A22, B22, C22, n)
      sync
  end if
end procedure
```

C ← C+A·B

Helyes ez?

Sajnos nem: versenyhelyzet

Kósa Márk
Pánovics János
Szathmáry László
Halász Gábor

Párhuzamos algoritmus

„Egyszerű” Oszd-meg-és-uralkodj változat

procedure M_S(A, B, C, n)

- **if** n=1 **then**
  - $c_{11} \leftarrow c_{11} + a_{11} \cdot b_{11}$
- **else**
  - n ← n/2
  - **spawn** M_S(A_{11}, B_{11}, C_{11}, n)
  - **spawn** M_S(A_{12}, B_{21}, T_{11}, n)
  - **spawn** M_S(A_{11}, B_{12}, C_{12}, n)
  - **spawn** M_S(A_{12}, B_{22}, T_{12}, n)
  - **spawn** M_S(A_{21}, B_{11}, C_{21}, n)
  - **spawn** M_S(A_{22}, B_{21}, T_{21}, n)
  - **spawn** M_S(A_{21}, B_{12}, C_{22}, n)
  - M_S(A_{22}, B_{22}, T_{22}, n)
  - **sync**
  - M_ADD(C, T, n)
- **end if**

end procedure

procedure M_ADD(C, T, n)

- **parallel for** i ← 1 **to** n **do**
  - **parallel for** j ← 1 **to** n **do**
    - C[i,j] ← C[i,j] + T[i,j]
  - **end for**
- **end for**

end procedure

- work: $M_1(n) = \Theta(n^3)$ $(8M_1(n/2) + \Theta(n^2))$
- span: $M_\infty(n) = \Theta(\lg^2 n)$ $(M_\infty(n/2) + \Theta(\lg n))$
- párhuzamosság: $\Theta(n^3 / \lg^2 n)$ Ez jobb, mint az „egyszerű” algoritmus párhuzamossága.

Párhuzamos algoritmus

„Egyszerű” Oszd-meg-és-uralkodj változat

procedure M_S(A, B, C, n)
```
if n=1 then
    c11 ← c11 + a11 · b11
else
    n ← n/2
    spawn M_S(A11, B11, C11, n)
    spawn M_S(A12, B21, T11, n)
    spawn M_S(A11, B12, C12, n)
    spawn M_S(A12, B22, T12, n)
    spawn M_S(A21, B11, C21, n)
    spawn M_S(A22, B21, T21, n)
    spawn M_S(A21, B12, C22, n)
    M_S(A22, B22, T22, n)
    sync
    M_ADD(C, T, n)
end if
```
end procedure

procedure M_ADD(C, T, n)
```
parallel for i ← 1 to n do
    parallel for j ← 1 to n do
        C[i,j] ← C[i,j] + T[i,j]
    end for
end for
```
end procedure

- work: $M_1(n) = \Theta(n^3)$ $(8M_1(n/2) + \Theta(n^2))$
- span: $M_\infty(n) = \Theta(\lg^2 n)$ $(M_\infty(n/2) + \Theta(\lg n))$
- párhuzamosság: $\Theta(n^3 / \lg^2 n)$ Ez jobb, mint az „egyszerű” algoritmus párhuzamossága.

Kósa Márk
Pánovics János
Szathmáry László
Halász Gábor

Párhuzamos algoritmus

„Egyszerű” Oszd-meg-és-uralkodj változat

procedure M_S(A, B, C, n)

if n=1 **then**
    $c_{11} \leftarrow c_{11} + a_{11} \cdot b_{11}$
else
    n $\leftarrow$ n/2
    spawn M_S(A_{11}, B_{11}, C_{11}, n)
    spawn M_S(A_{12}, B_{21}, T_{11}, n)
    spawn M_S(A_{11}, B_{12}, C_{12}, n)
    spawn M_S(A_{12}, B_{22}, T_{12}, n)
    spawn M_S(A_{21}, B_{11}, C_{21}, n)
    spawn M_S(A_{22}, B_{21}, T_{21}, n)
    spawn M_S(A_{21}, B_{12}, C_{22}, n)
    M_S(A_{22}, B_{22}, T_{22}, n)
    sync
    M_ADD(C, T, n)
end if

end procedure

procedure M_ADD(C, T, n)

parallel for i $\leftarrow$ 1 **to** n **do**
    parallel for j $\leftarrow$ 1 **to** n **do**
        C[i,j] $\leftarrow$ C[i,j] + T[i,j]
    end for
end for

end procedure

- work: $M_1(n) = \Theta(n^3)$ $(8M_1(n/2) + \Theta(n^2))$
- span: $M_\infty(n) = \Theta(\lg^2 n)$ $(M_\infty(n/2) + \Theta(\lg n))$
- párhuzamosság: $\Theta(n^3 / \lg^2 n)$ Ez jobb, mint az „egyszerű” algoritmus párhuzamossága.

Párhuzamos algoritmus

„Egyszerű” Oszd-meg-és-uralkodj változat

```
procedure M_S(A, B, C, n)
    if n=1 then
        c11 ← c11 + a11 · b11
    else
        n ← n/2
        spawn M_S(A11, B11, C11, n)
        spawn M_S(A12, B21, T11, n)
        spawn M_S(A11, B12, C12, n)
        spawn M_S(A12, B22, T12, n)
        spawn M_S(A21, B11, C21, n)
        spawn M_S(A22, B21, T21, n)
        spawn M_S(A21, B12, C22, n)
        M_S(A22, B22, T22, n)
        sync
        M_ADD(C, T, n)
    end if
end procedure
```

```
procedure M_ADD(C, T, n)
    parallel for i ← 1 to n do
        parallel for j ← 1 to n do
            C[i,j] ← C[i,j] + T[i,j]
        end for
    end for
end procedure
```

- work: $M_1(n) = \Theta(n^3)$ $(8M_1(n/2) + \Theta(n^2))$
- span: $M_\infty(n) = \Theta(\lg^2 n)$ $(M_\infty(n/2) + \Theta(\lg n))$
- párhuzamosság: $\Theta(n^3 / \lg^2 n)$ Ez jobb, mint az „egyszerű” algoritmus párhuzamossága.

$M_\infty(n) = M_\infty(n/2) + \Theta(\lg n)$

Theorem (Mester tétel)

Legyenek $a \geq 1$, $b > 1$ állandók, $f(n)$ egy függvény, $T(n)$ pedig a nemnegatív egészeken a

$$T(n) = aT\left(\frac{n}{b}\right) + f(n)$$

rekurzív egyenlettel definiált függvény, ahol n/b jelentheti akár az $\lfloor n/b \rfloor$ egészrészt, akár az $\lceil n/b \rceil$ egészrészt. Ekkor:

1. *Ha $f(n) = O\left(n^{\log_b a - \epsilon}\right)$, egy $\epsilon > 0$ állandóval, akkor* $\longrightarrow T(n) = \Theta\left(n^{\log_b a}\right)$.
2. *Ha $f(n) = \Theta\left(n^{\log_b a}\right)$, akkor* $\longrightarrow T(n) = \Theta\left(n^{\log_b a} \lg n\right) \left(= \Theta\left(f(n) \lg n\right)\right)$.
3. *Ha $f(n) = \Omega\left(n^{\log_b a + \epsilon}\right)$ és $a \cdot f(n/b) \leq c \cdot f(n)$ valamely $\epsilon > 0$ és $c < 1$ állandóra és elég nagy n-re, akkor* $\longrightarrow T(n) = \Theta\left(f(n)\right)$.

$M_\infty(n) = M_\infty(n/2) + \Theta(\lg n)$

Theorem (Tankönyv 4.6-2 feladat)

Ha $f(n) = \Theta(n^{\log_b a} \lg^k n)$, *ahol* $k \geq 0$, *akkor a mester rekurzió megoldása:*

$$T(n) = \Theta\left(n^{\log_b a} \lg^{k+1} n\right).$$

Esetünkben: a=1, b=2, k=1: $T(n) = \Theta(n^{\lg 1} \lg^2 n) = \Theta(\lg^2 n)$

Párhuzamos algoritmus

Strassen algoritmusa

1. Osszuk fel az A és B mátrixokat $\frac{n}{2} \times \frac{n}{2}$ méretű részmátrixokra. work=span=$\Theta(1)$
2. Hozzunk létre 10 darab, szintén $\frac{n}{2} \times \frac{n}{2}$ méretű új mátrixot az előbb definiált részmátrixokkal meghatározva:
   $$\begin{array}{lll} S_1 = B_{12} - B_{22}, & S_2 = A_{11} + A_{12}, & S_3 = A_{21} + A_{22}, \\ S_4 = B_{21} - B_{11}, & S_5 = A_{11} + A_{22}, & S_6 = B_{11} + B_{22}, \\ S_7 = A_{12} - A_{22}, & S_8 = B_{21} + B_{22}, & S_9 = A_{11} - A_{21}, \\ S_{10} = B_{11} + B_{12}. & \text{work: } \Theta(n^2), & \text{span: } \Theta(\lg n) \end{array}$$
3. Ezek felhasználásával számoljunk ki újabb 7 mátrixot:
   $$P_1 = A_{11} \cdot S_1,\ P_2 = S_2 \cdot B_{22},\ P_3 = S_3 \cdot B_{11},\ P_4 = A_{22} \cdot S_4,$$
   $$P_5 = S_5 \cdot S_6,\quad P_6 = S_7 \cdot S_8,\quad P_7 = S_9 \cdot S_{10}.$$
   work: $\Theta(n^{\lg 7})$, span: $\Theta(\lg^2 n)$
4. És végül határozzuk meg a C (szintén $\frac{n}{2} \times \frac{n}{2}$ méretű) részmátrixait:
   $$\begin{array}{ll} C_{11} = P_5 + P_4 - P_2 + P_6, & C_{12} = P_1 + P_2, \\ C_{21} = P_3 + P_4, & C_{22} = P_5 + P_1 - P_3 - P_7. \end{array}$$
   work: $\Theta(n^2)$, span: $\Theta(\lg n)$

- work: $\Theta(n^{\lg 7})$, span: $\Theta(\lg^2 n)$,
- párhuzamosság: $\Theta(n^{\lg 7} / \lg^2 n)$

Összefésülő rendezés

```
procedure RENDEZ'(A, p, r)
  if p < r then
      q ← ⌊(p + q)/2⌋
      spawn RENDEZ'(A, p, q)
      RENDEZ'(A, q+1, r)
      sync
      ÖSSZEFÉSÜL(A, p, q, r)
  end if
end procedure
```

```
procedure ÖSSZEFÉSÜL(A,p,q,r)
    n1 ← q-p+1
    n2 ← r-q
    for i ← 1 to n1 do
        L[i] ← A[p+i-1]
    end for
    for j ← 1 to n2 do
        R[j] ← A[q+j]
    end for
    L[n1+1] ← R[n2+1] ← ∞
    i ← j ← 1
    for k ← p to r do
        if L[i] ≤ R[j] then
            A[k] ← L[i]
            i ← i + 1
        else
            A[k] ← R[j]
            j ← j + 1
        end if
    end for
end procedure
```


Kósa Márk
Pánovics János
Szathmáry László
Halász Gábor

Összefésülő rendezés

```
procedure RENDEZ'(A, p, r)
  if p < r then
      q ← ⌊(p + q)/2⌋
      spawn RENDEZ'(A, p, q)
      RENDEZ'(A, q+1, r)
      sync
      ÖSSZEFÉSÜL(A, p, q, r)
  end if
end procedure
```

```
procedure ÖSSZEFÉSÜL(A,p,q,r)
  n1 ← q-p+1
  n2 ← r-q
  for i ← 1 to n1 do
      L[i] ← A[p+i-1]
  end for
  for j ← 1 to n2 do
      R[j] ← A[q+j]
  end for
  L[n1+1] ← R[n2+1] ← ∞
  i ← j ← 1
  for k ← p to r do
      if L[i] ≤ R[j] then
          A[k] ← L[i]
          i ← i + 1
      else
          A[k] ← R[j]
          j ← j + 1
      end if
  end for
end procedure
```

- work:
  $R_1'(n) = \Theta(n \lg n)$
  $(2R_1'(n/2) + \Theta(n))$
- span:
  $R_\infty'(n) = \Theta(n)$
  $(R_\infty'(n/2) + \Theta(n))$
- párhuzamosság:
  $\Theta(\lg n)$
  Ez elég „sovány”. Nincs annyi adat, hogy néhány száz processzort érdemes legyen használni a rendezéshez.
- Lehetne-e az összefésülést is párhuzamosítani?

Összefésülő rendezés

procedure RENDEZ'(A, p, r)

```
if p < r then
    q ← ⌊(p + q)/2⌋
    spawn RENDEZ'(A, p, q)
    RENDEZ'(A, q+1, r)
    sync
    ÖSSZEFÉSÜL(A, p, q, r)
end if
end procedure
```

```
procedure ÖSSZEFÉSÜL(A,p,q,r)
    n1 ← q-p+1
    n2 ← r-q
    for i ← 1 to n1 do
        L[i] ← A[p+i-1]
    end for
    for j ← 1 to n2 do
        R[j] ← A[q+j]
    end for
    L[n1+1] ← R[n2+1] ← ∞
    i ← j ← 1
    for k ← p to r do
        if L[i] ≤ R[j] then
            A[k] ← L[i]
            i ← i + 1
        else
            A[k] ← R[j]
            j ← j + 1
        end if
    end for
end procedure
```

- work:
  $R_1'(n) = \Theta(n \lg n)$
  $(2R_1'(n/2) + \Theta(n))$
- span:
  $R_\infty'(n) = \Theta(n)$
  $(R_\infty'(n/2) + \Theta(n))$
- párhuzamosság:
  $\Theta(\lg n)$
  Ez elég „sovány”. Nincs annyi adat, hogy néhány száz processzort érdemes legyen használni a rendezéshez.
- Lehetne-e az összefésülést is párhuzamosítani?

Összefésülő rendezés

procedure RENDEZ'(A, p, r)

if p < r **then**
  q $\leftarrow \lfloor (p+q)/2 \rfloor$
  spawn RENDEZ'(A, p, q)
  RENDEZ'(A, q+1, r)
  sync
  ÖSSZEFÉSÜL(A, p, q, r)
end if

end procedure

procedure ÖSSZEFÉSÜL(A,p,q,r)

  $n_1 \leftarrow$ q-p+1
  $n_2 \leftarrow$ r-q
  for i $\leftarrow$ 1 **to** n_1 **do**
    L[i] $\leftarrow$ A[p+i-1]
  end for
  for j $\leftarrow$ 1 **to** n_2 **do**
    R[j] $\leftarrow$ A[q+j]
  end for
  L[n_1+1]$\leftarrow$R[n_2+1]$\leftarrow \infty$
  i $\leftarrow$ j $\leftarrow$ 1
  for k $\leftarrow$ p **to** r **do**
    if L[i] $\leq$ R[j] **then**
      A[k] $\leftarrow$ L[i]
      i $\leftarrow$ i + 1
    else
      A[k] $\leftarrow$ R[j]
      j $\leftarrow$ j + 1
    end if
  end for

end procedure

- work:
  $R'_1(n) = \Theta(n \lg n)$
  $(2R'_1(n/2) + \Theta(n))$
- span:
  $R'_\infty(n) = \Theta(n)$
  $(R'_\infty(n/2) + \Theta(n))$
- párhuzamosság:
  $\Theta(\lg n)$
  Ez elég „sovány”. Nincs annyi adat, hogy néhány száz processzort érdemes legyen használni a rendezéshez.
- Lehetne-e az összefésülést is párhuzamosítani?

Összefésülő rendezés

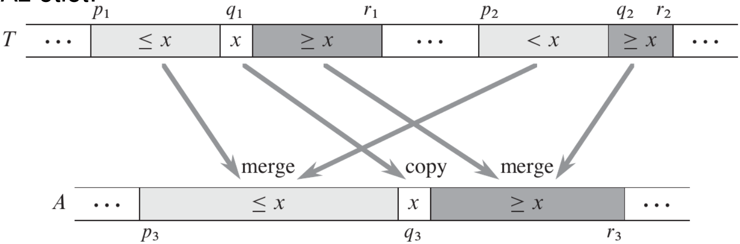
```
procedure RENDEZ'(A, p, r)
    if p < r then
        q ← ⌊(p + q)/2⌋
        spawn RENDEZ'(A, p, q)
        RENDEZ'(A, q+1, r)
        sync
        ÖSSZEFÉSÜL(A, p, q, r)
    end if
end procedure
```

```
procedure ÖSSZEFÉSÜL(A,p,q,r)
    n1 ← q-p+1
    n2 ← r-q
    for i ← 1 to n1 do
        L[i] ← A[p+i-1]
    end for
    for j ← 1 to n2 do
        R[j] ← A[q+j]
    end for
    L[n1+1]←R[n2+1]←∞
    i ← j ← 1
    for k ← p to r do
        if L[i] ≤ R[j] then
            A[k] ← L[i]
            i ← i + 1
        else
            A[k] ← R[j]
            j ← j + 1
        end if
    end for
end procedure
```

- work:
  $R'_1(n) = \Theta(n \lg n)$
  $(2R'_1(n/2) + \Theta(n))$
- span:
  $R'_\infty(n) = \Theta(n)$
  $(R'_\infty(n/2) + \Theta(n))$
- párhuzamosság: $\Theta(\lg n)$
  Ez elég „sovány”. Nincs annyi adat, hogy néhány száz processzort érdemes legyen használni a rendezéshez.
- Lehetne-e az összefésülést is párhuzamosítani?

Párhuzamos összefésülés

Az ötlet:




```
procedure BINÁRIS_KERES(x, A, p, r)
  alsó ← p
  felső ← MAX(p, r+1)
  while alsó < felső do
      középső ← ⌊(alsó+felső)/2⌋
      if x ≤ A[középső] then
          felső ← középső
      else
          alsó ← középső+1
      end if
  end while
  return felső
end procedure
```


Párhuzamosság

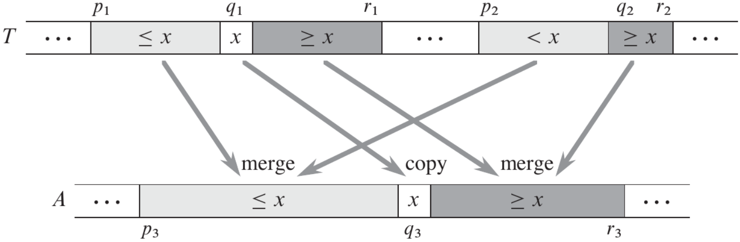
Kósa Márk
Pánovics János
Szathmáry László
Halász Gábor

Mátrix-szorzás
Soros algoritmus
Strassen algoritmusa
Párhuzamos algoritmus

Összefésülő rendezés
Párhuzamos összefésülés
Batcher-féle páros-páratlan összefésülés

Párhuzamos összefésülés

Az ötlet:



procedure BINÁRIS_KERES (x, A, p, r)

```
alsó ← p
felső ← MAX(p, r+1)
while alsó < felső do
    középső ← ⌊(alsó+felső)/2⌋
    if x ≤ A[középső] then
        felső ← középső
    else
        alsó ← középső+1
    end if
end while
return felső
```

end procedure

Párhuzamos összefésülés

```
procedure P_ÖSSZEFÉSÜL(T, p1, r1, p2, r2, A, p3)
  n1 ← r1-p1+1
  n2 ← r2-p2+1
  if n1 < n2 then
      (p1, r1, n1) és (p2, r2, n2) felcserélése
  end if
  if n1>0 then
      q1 ← ⌊(p1+p2)/2⌋
      q2 ← BINÁRIS_KERES(T[q1], T, p2, r2)
      q3 ← p3+(q1-p1)+(q2-p2)
      A[q3] ← T[q1]
      spawn P_ÖSSZEFÉSÜL(T, p1, q1-1, p2, q2-1, A, p3)
      P_ÖSSZEFÉSÜL(T, q1+1, r1, q2, r2, A, q3+1)
      sync
  end if
end procedure
```


Kósa Márk
Pánovics János
Szathmáry László
Halász Gábor

Párhuzamos összefésülés

procedure P_ÖSSZEFÉSÜL(T, p1, r1, p2, r2, A, p3)

$n1 \leftarrow$ r1-p1+1
$n2 \leftarrow$ r2-p2+1
if n1 < n2 **then**
    (p1, r1, n1) és (p2, r2, n2) felcserélése
end if
if n1>0 **then**
    $q1 \leftarrow \lfloor (p1+p2)/2 \rfloor$
    $q2 \leftarrow$ BINÁRIS_KERES(T[q1], T, p2, r2)
    $q3 \leftarrow$ p3+(q1-p1)+(q2-p2)
    $A[q3] \leftarrow T[q1]$
    spawn P_ÖSSZEFÉSÜL
        (T, p1, q1-1, p2, q2-1, A, p3)
    P_ÖSSZEFÉSÜL(T, q1+1, r1, q2, r2, A, q3+1)
    sync
end if
end procedure

- work: $F_1(n) = \Theta(n)$
  $(F_1(\alpha n) + F_1((1-\alpha)n) + \Theta(\lg n))$
  $0.25 \leq \alpha \leq 0.75$
- span: $F_\infty(n) = \Theta(\lg^2 n)$
  $(F_\infty(3n/4) + \Theta(\lg n))$
- párhuzamosság: $\Theta(n/\lg^2 n)$

Párhuzamos összefésülés

```
procedure P_ÖSSZEFÉSÜL(T, p1, r1, p2, r2, A, p3)
  n1 ← r1-p1+1
  n2 ← r2-p2+1
  if n1 < n2 then
      (p1, r1, n1) és (p2, r2, n2) felcserélése
  end if
  if n1>0 then
      q1 ← ⌊(p1+p2)/2⌋
      q2 ← BINÁRIS_KERES(T[q1], T, p2, r2)
      q3 ← p3+(q1-p1)+(q2-p2)
      A[q3] ← T[q1]
      spawn P_ÖSSZEFÉSÜL
                      (T, p1, q1-1, p2, q2-1, A, p3)
      P_ÖSSZEFÉSÜL(T, q1+1, r1, q2, r2, A, q3+1)
      sync
  end if
end procedure
```

- work: $F_1(n) = \Theta(n)$
  $(F_1(\alpha n) + F_1((1-\alpha)n) + \Theta(\lg n))$
  $0.25 \leq \alpha \leq 0.75$
- span:
  $F_\infty(n) = \Theta(\lg^2 n)$
  $(F_\infty(3n/4) + \Theta(\lg n))$
- párhuzamosság:
  $\Theta(n/\lg^2 n)$

Párhuzamos összefésülés

```
procedure P_ÖSSZEFÉSÜL(T, p1, r1, p2, r2, A, p3)
    n1 ← r1-p1+1
    n2 ← r2-p2+1
    if n1 < n2 then
        (p1, r1, n1) és (p2, r2, n2) felcserélése
    end if
    if n1>0 then
        q1 ← ⌊(p1+p2)/2⌋
        q2 ← BINÁRIS_KERES(T[q1], T, p2, r2)
        q3 ← p3+(q1-p1)+(q2-p2)
        A[q3] ← T[q1]
        spawn P_ÖSSZEFÉSÜL
                          (T, p1, q1-1, p2, q2-1, A, p3)
        P_ÖSSZEFÉSÜL(T, q1+1, r1, q2, r2, A, q3+1)
        sync
    end if
end procedure
```

- work: $F_1(n) = \Theta(n)$
  $(F_1(\alpha n) + F_1((1-\alpha)n)$
  $+\Theta(\lg n))$
  $0.25 \le \alpha \le 0.75$
- span:
  $F_\infty(n) = \Theta(\lg^2 n)$
  $(F_\infty(3n/4) + \Theta(\lg n))$
- párhuzamosság:
  $\Theta(n/\lg^2 n)$

Összefésülő rendezés

```
procedure RENDEZ(A, p, r, B, s)
    n ← r-p+1
    if n = 1 then
        B[s] ← A[p]
    else
        Legyen T[1..n] új tömb
        q1 ← ⌊(p+r)/2⌋
        q2 ← q1-p+1
        spawn RENDEZ(A, p, q1, T, 1)
        RENDEZ(A, q1+1, r, T, q2+1)
        sync
        P_ÖSSZEFÉSÜL
                (T, 1, q2, q2+1, n, B, s)
    end if
end procedure
```

- work: $R_1(n) = \Theta(n \lg n)$ $(2R_1(n/2) + \Theta(n))$
- span: $R_\infty(n) = \Theta(\lg^3 n)$ $\left(R_\infty(n/2) + \Theta(\lg^2 n)\right)$
- párhuzamosság: $\Theta(n/\lg^2 n)$

Itt már egy masszív párhuzamosítást lehet elérni.

Kósa Márk
Pánovics János
Szathmáry László
Halász Gábor

Összefésülő rendezés

```
procedure RENDEZ(A, p, r, B, s)
  n ← r-p+1
  if n = 1 then
      B[s] ← A[p]
  else
      Legyen T[1..n] új tömb
      q1 ← ⌊(p+r)/2⌋
      q2 ← q1-p+1
      spawn RENDEZ(A, p, q1, T, 1)
      RENDEZ(A, q1+1, r, T, q2+1)
      sync
      P_ÖSSZEFÉSÜL
                (T, 1, q2, q2+1, n, B, s)
  end if
end procedure
```

- work: $R_1(n) = \Theta(n \lg n)$ $(2R_1(n/2) + \Theta(n))$
- span: $R_\infty(n) = \Theta(\lg^3 n)$ $\left(R_\infty(n/2) + \Theta(\lg^2 n)\right)$
- párhuzamosság: $\Theta(n/\lg^2 n)$

Itt már egy masszív párhuzamosítást lehet elérni.

Kósa Márk
Pánovics János
Szathmáry László
Halász Gábor

Összefésülő rendezés

```
procedure RENDEZ(A, p, r, B, s)
  n ← r-p+1
  if n = 1 then
      B[s] ← A[p]
  else
      Legyen T[1..n] új tömb
      q1 ← ⌊(p+r)/2⌋
      q2 ← q1-p+1
      spawn RENDEZ(A, p, q1, T, 1)
      RENDEZ(A, q1+1, r, T, q2+1)
      sync
      P_ÖSSZEFÉSÜL
                (T, 1, q2, q2+1, n, B, s)
  end if
end procedure
```

- work:
  $R_1(n) = \Theta(n \lg n)$
  $(2R_1(n/2) + \Theta(n))$
- span:
  $R_\infty(n) = \Theta(\lg^3 n)$
  $\left(R_\infty(n/2) + \Theta(\lg^2 n)\right)$
- párhuzamosság:
  $\Theta(n/\lg^2 n)$

Itt már egy masszív párhuzamosítást lehet elérni.

Összefésülő rendezés

```
procedure RENDEZ(A, p, r, B, s)
  n ← r-p+1
  if n = 1 then
      B[s] ← A[p]
  else
      Legyen T[1..n] új tömb
      q1 ← ⌊(p+r)/2⌋
      q2 ← q1-p+1
      spawn RENDEZ(A, p, q1, T, 1)
      RENDEZ(A, q1+1, r, T, q2+1)
      sync
      P_ÖSSZEFÉSÜL
                (T, 1, q2, q2+1, n, B, s)
  end if
end procedure
```

- work:
  $R_1(n) = \Theta(n \lg n)$
  $(2R_1(n/2) + \Theta(n))$
- span:
  $R_\infty(n) = \Theta(\lg^3 n)$
  $\left(R_\infty(n/2) + \Theta(\lg^2 n)\right)$
- párhuzamosság:
  $\Theta(n/\lg^2 n)$

Itt már egy masszív párhuzamosítást lehet elérni.

Batcher-féle páros-páratlan összefésülés

Az ötlet:

```
A={ 1, 2, 3, 4, 5, 9,12,14,19,20};
B={ 6, 7, 8,10,11,13,15,16,17,18};

u={ 1, 3, 5, 7,10,12,13,16,18,19};
v={ 2, 4, 6, 8, 9,11,14,15,17,20};
```

Batcher-féle páros-páratlan összefésülés

Az ötlet:

```
A={ 1, 2, 3, 4, 5, 9,12,14,19,20};
B={ 6, 7, 8,10,11,13,15,16,17,18};

u={ 1, 3, 5, 7,10,12,13,16,18,19};
v={ 2, 4, 6, 8, 9,11,14,15,17,20};
```


Kósa Márk
Pánovics János
Szathmáry László
Halász Gábor

Batcher-féle páros-páratlan összefésülés

Az ötlet:

```
A={ 1, 2, 3, 4, 5, 9,12,14,19,20};
B={ 6, 7, 8,10,11,13,15,16,17,18};

u={ 1, 3, 5, 7,10,12,13,16,18,19};
v={ 2, 4, 6, 8, 9,11,14,15,17,20};
```

Batcher-féle páros-páratlan összefésülés

```
procedure B_ÖSSZEFÉSÜL(T, p1, r1, p2, r2, A, p3, k)
  n1 ← ⌈(r1-p1+1)/k⌉
  n2 ← ⌈(r2-p2+1)/k⌉
  if p1>r1 then
     parallel for i ← 0 to n2-1 do
        A[p3+k*i] ← T[p2+k*i]
     end for
  else if p2>r2 then
     parallel for i ← 0 to n1-1 do
        A[p3+k*i] ← T[p1+k*i]
     end for
  end if
  else
     kk ← k+k
     spawn B_ÖSSZEFÉSÜL(T, p1, r1, p2+k, r2, A, p3,kk)
     B_ÖSSZEFÉSÜL(T, p1+k, r1, p2, r2, A, p3,kk)
     sync
     parallel for i ← 1 to ⌊(n1+n2)/2⌋ do
        if A[p3+i*kk-kk]>A[p3+i*kk] then
           A[p3+i*kk-kk] és A[p3+i*kk] felcserélése
        end if
     end for
  end if
end procedure
```

Az ötlet:

```
A={ 1, 2, 3, 4, 5, 9,12,14,19,20};
B={ 6, 7, 8,10,11,13,15,16,17,18};

u={ 1, 3, 5, 7,10,12,13,16,18,19};
v={ 2, 4, 6, 8, 9,11,14,15,17,20};
```