Optimálne paralelné algoritmy

- **PRAM** = **Parallel Random Access Machine**
  - model s (neobmedzenou) zdieľanou pamäťou
  - **synchrónny model** – spoločné hodiny
  - **SIMD** – v každom kroku sa vykonáva rovnaká inštrukcia
  - vstup aj výstup v zdieľanej pamäti
  - každý procesor má ID a pozná celkový počet procesorov
  - pamäťové modely: EREW, CREW, CRCW (common, arbitrary, priority)
  - zrýchlenie: $S_p(n) = T^*(n) / T_p(n)$
  - cena: $C_p(n) = p.T_p(n)$

Kód pre procesor i ($1 \leq i \leq n$):

```
global_read(A[i], a);
global_write(a, B[i]);
for h:=1 to log n do
   if (i <= n/2^h) then
   begin
     global_read(B[2*i-1], x);
     global_read(B[2*i], y);
     z := x + y;
     global_write(z, B[i]);
   end;
if i = 1 then global_write(z, S);
```

EREW model

$T^*(n) \in \Theta(n)$

$T_n(n) \in \Theta(\log(n))$

$S_n(n) \in \Theta(n/\log(n))$

$C_n(n) \in \Theta(n.\log(n))$

$E_n(n) \in \Theta(1/\log(n))$

nie je cenovo optimálny

index prvého výskytu M[n+1] v poli M[1..n] zapísať do M[n+2]

Kód pre procesor i ($1 \le i \le n$):

```
global_read(M[n+1], x);
global_read(M[i], y);
global_write(0, M[n+2]);
if x==y then global_write(i, M[n+2]);
```

CRCW-priority model

$T^*(n) \in \Theta(n)$

$T_n(n) \in \Theta(1)$

$S_n(n) \in \Theta(n)$

$C_n(n) \in \Theta(n)$

$E_n(n) = \frac{n+2}{n*4} \approx \frac{1}{4} \in \Theta(1)$

cenovo optimálny

Work-time zápis algoritmov

- WT presentation framework:
  - zápis **abstrahujúci** od detailov PRAM implementácie
  - **neviaže sa na fixný počet** použitých procesorov
  - **postupnosť časových jednotiek**, kde každá môže obsahovať ľubovoľne veľa konkurentných operácií
  - sekvenčný algoritmus s konštrukciou „paralelný for“, ktorá vykoná zadaný príkaz konkurentne pre všetky hodnoty **i** (i je identifikátor príslušného procesu)

```
for i := 1 to n pardo
    príkaz;
```

Kód pre procesor i (1 ≤ i ≤n):

```
global_read(A[i], a);
global_write(a, B[i]);
for h := 1 to log n do
   if (i <= n/2^h) then
   begin
     global_read(B[2*i-1], x);
     global_read(B[2*i], y);
     z := x + y;
     global_write(z, B[i]);
   end;
if i = 1 then global_write(z, S);
```

```
for i := 1 to n pardo
  B[i] := A[i];

for h := 1 to log n do
  for i := 1 to n/2^h pardo
    B[i] := B[2*i-1] + B[2*i];

S := B[1];
```

- **T (n)** – počet časových jednotiek, ktoré potrebuje algoritmus na vstup veľkosti n
- **W(n)** – celkový počet operácií, ktoré vykoná algoritmus na vstupe veľkosti n (práca algoritmu)

```
for i := 1 to n pardo
    A[i] := i;
```

Počet časových jednotiek: **1**
Počet vykonaných operácií: **n**

Priradenie: WT vs. PRAM

- `A[j] := A[j] + 1`

```
global_read(A[j], x);
x := x + 1;
global_write(x, A[j]);
```

Paralelný čas: O(1)

- synchronizáciu zabezpečí systém

Paralelné sčítanie vo WT (2)

Kód pre procesor i (1 ≤ i ≤n):

```
global_read(A[i], a);
global_write(a, B[i]);
for h := 1 to log n do
   if (i <= n/2^h) then
   begin
     global_read(B[2*i-1], x);
     global_read(B[2*i], y);
     z := x + y;
     global_write(z, B[i]);
   end;
if i = 1 then global_write(z, S);
```

```
for i := 1 to n pardo
  B[i] := A[i];
// 1 krok, n operácií
for h := 1 to log n do
  for i := 1 to n/2^h pardo
    B[i] := B[2*i-1] + B[2*i];
```
// log n krokov, $\sum_{h=1}^{\log n} n/2^h$ operácií
```
S := B[1];
```

EREW
Počet časových jednotiek:
T(n) ϵ Θ(log(n))
Počet vykonaných operácií:
W(n) ϵ Θ(n)

WT-rozvrhovanie

- WT-rozvrhovanie = mapovanie na PRAM procesory
- **for** i := 1 **to** n **pardo** na p procesoroch:
  - procesor 1 vykoná príkaz pre i=1, p+1, 2p+1, 3p+1, …
  - procesor 2 vykoná príkaz pre i=2, p+2, 2p+2, …
  - …
  - procesor j vykoná príkaz pre i=j, p+j, 2p+j, 3p+j, …

paralelný čas:

$$\lceil n/p \rceil \leq \lfloor n/p \rfloor + 1$$

WT-rozvrhovanie

- Algoritmus s prácou **W(n)** a časom **T (n)** simulovaný na p procesoroch:
  - $T_p(n) \leq \lfloor W(n)/p \rfloor + T(n)$
    - každý krok (časovú jednotku) s w operáciami vieme odsimulovať na p procesoroch v čase ≤ w/p+1
  - $C_p(n) = p.\ T_p(n) \leq p.(\lfloor W(n)/p \rfloor + T(n)) \in \Theta(W(n) + p.T(n))$
    - pre $p \in O(W(n)/T(n))$ dostávame $C_p(n) \in O(W(n))$

(Brent) $T(n)/p \leq T_p(n) \leq (T(n) - T_\infty)/p + T_\infty \leq T(n)/p + T_\infty$

WT a optimalita

- **T(n)** – počet časových jednotiek, ktoré vykoná algoritmus na vstupe veľkosti n
- **W(n)** – celkový počet operácií, ktoré vykoná algoritmus na vstupe veľkosti n (práca algoritmu)
- optimalita
  - optimálny algoritmus: $W(n) \in O(T^*(n))$
  - WT-optimálny algoritmus:
    $W(n) \in O(T^*(n))$ a $T(n)$ sa nedá zlepšiť
- ak je algoritmus optimálny, potom $S_p(n) \in \Theta(p)$ pre $p \in O(T^*(n)/T(n))$

Paralelné sčítanie vo WT (3)

-

```
for i := 1 to n pardo
  B[i] := A[i];
// 1 krok, n operácií
for h := 1 to log n do
  for i := 1 to n/2^h pardo
    B[i] := B[2*i-1] + B[2*i];
// log n krokov, ∑_{h=1}^{log n} n/2^h operácií
S := B[1];
```

EREW
T(n) ∊ $\Theta(\log n)$
W(n) ∊ $\Theta(n)$
C_p(n) ∊ $\Theta(n + p.\log n)$

T*(n) ∊ $\Theta(n)$
W(n) ∊ $\Theta(T^*(n))$ optimálny

optimálne zrýchlenie
pre p ∊ $\Theta(n/\log n)$

pre CREW PRAM
T(n) ∊ $\Omega(\log n)$, teda
WT-optimálny pre CREW

- **Vstup:**
  - binárna asociatívna operácia + (*, min, max, or, ...)
  - n-prvkové pole A, kde $n = 2^k$
- **Výstup**:
  - n-prvkové pole S také, že S[i] = A[1] + A[2] + ... + A[i]
- **Sekvenčný algoritmus v čase O(n):**

```
S[1] := A[1];
for i := 2 to N do
  S[i] := S[i-1] + A[i];
```

Hillis-Steele prefix sum vo WT

```
for i := 1 to n pardo S[i] := A[i]; // 1 krok, n operácií
for h := 0 to log(n)-1 do
      for i := 2^h+1 to n pardo
            S[i] := S[i] + S[i-2^h];
```

je možné realizovať EREW modelom ?

// log n krokov, $\sum_{h=0}^{\log(n)-1}(n-2^h)$ operácií

Hillis-Steele prefix sum vo WT

```
for i := 1 to n pardo S[i] := A[i]; // 1 krok, n operácií
for h := 0 to log(n)-1 do
      for i := 2^h+1 to n pardo
            S[i] := S[i] + S[i-2^h];
```

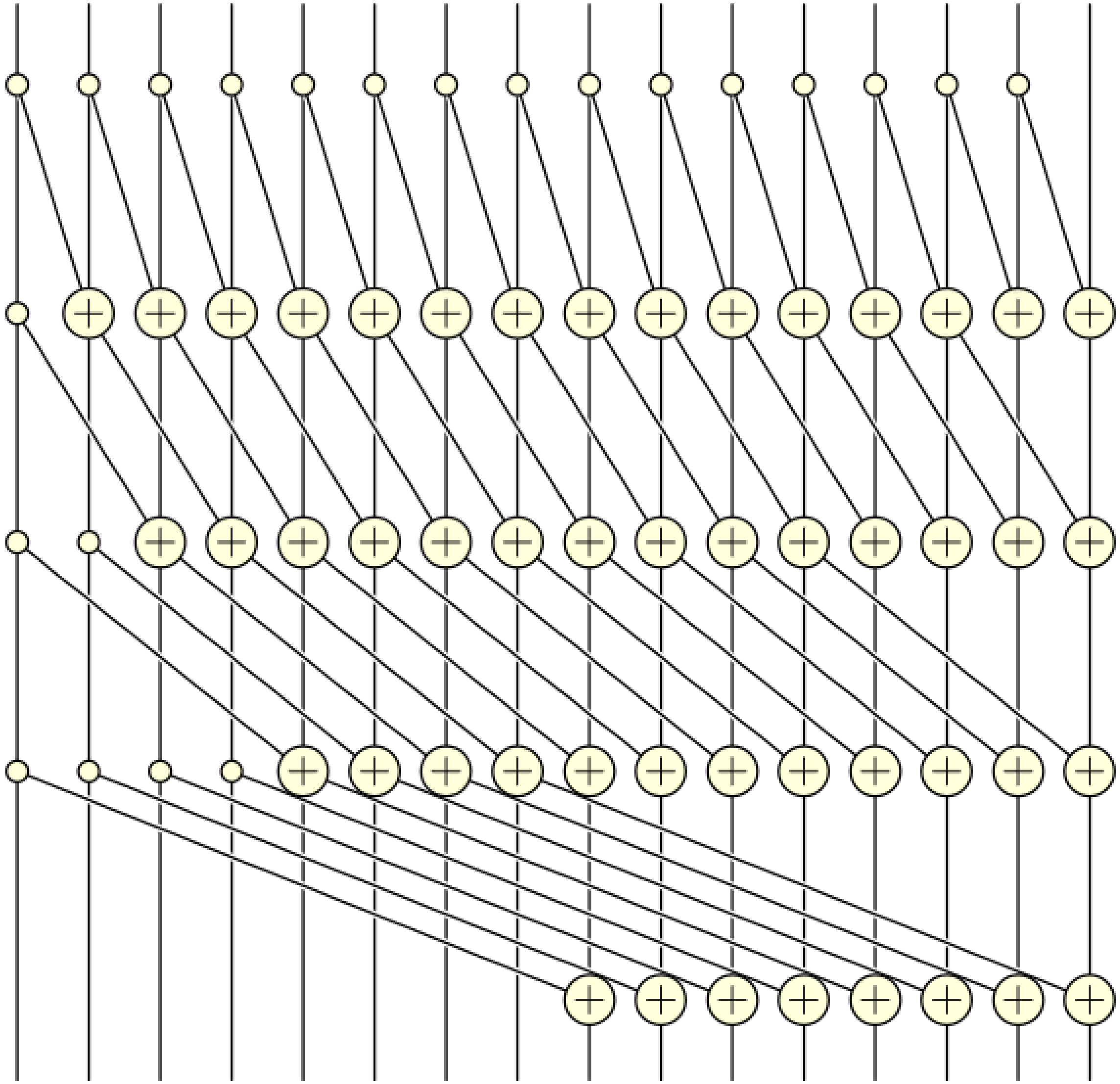
je možné realizovať EREW modelom ?

// log n krokov, $\sum_{h=0}^{\log(n)-1}(n-2^h)$ operácií

EREW
T(n) ϵ Θ(log n)
W(n) ϵ Θ(n.log n)

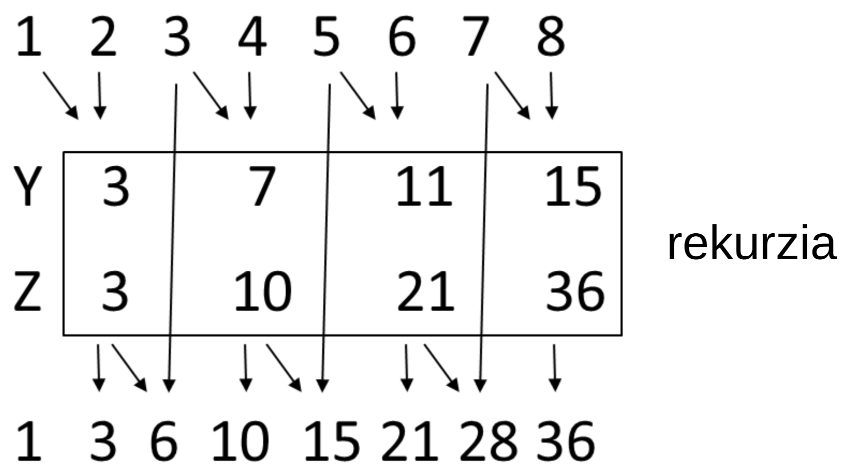
T*(n) ϵ Θ(n)
W(n) nie je v Θ(T*(n)) nie je optimálny

Hillis-Steele prefix sum



Binárna redukcia problému

- A[1], ..., A[n] →
  - Y[1] = A[1] + A[2]
  - Y[2] = A[3] + A[4]
  - ...
  - Y[i] = A[2i-1] + A[2i]



- Z[1], ..., Z[n/2] – prefixové súčty pre Y[1], ..., Y[n/2]
- Z[i] = Y[1] + Y[2] + ... + Y[i] =
  A[1] + A[2] + ... + A[2i-1] + A[2i] = S[2i]
- S[2i+1] = S[2i] + A[2i+1] = Z[i] + A[2i+1]

Prefixové súčty rekurzívne

Správnosť – matematickou indukciou podľa k, kde $n=2^k$

```
{ pre n = 2^k }
if n > 1 then
  for i := 1 to n/2 pardo
    Y[i] := A[2*i-1]+A[2*i];

Rekurzívne vypočítaj prefixovú sumu
  pre pole Y a ulož ju do poľa Z

S[1] := A[1];
for i := 2 to n pardo
      if i mod 2 = 0 then S[i] := Z[i/2]
                     else S[i] := Z[i/2] + A[i];
```

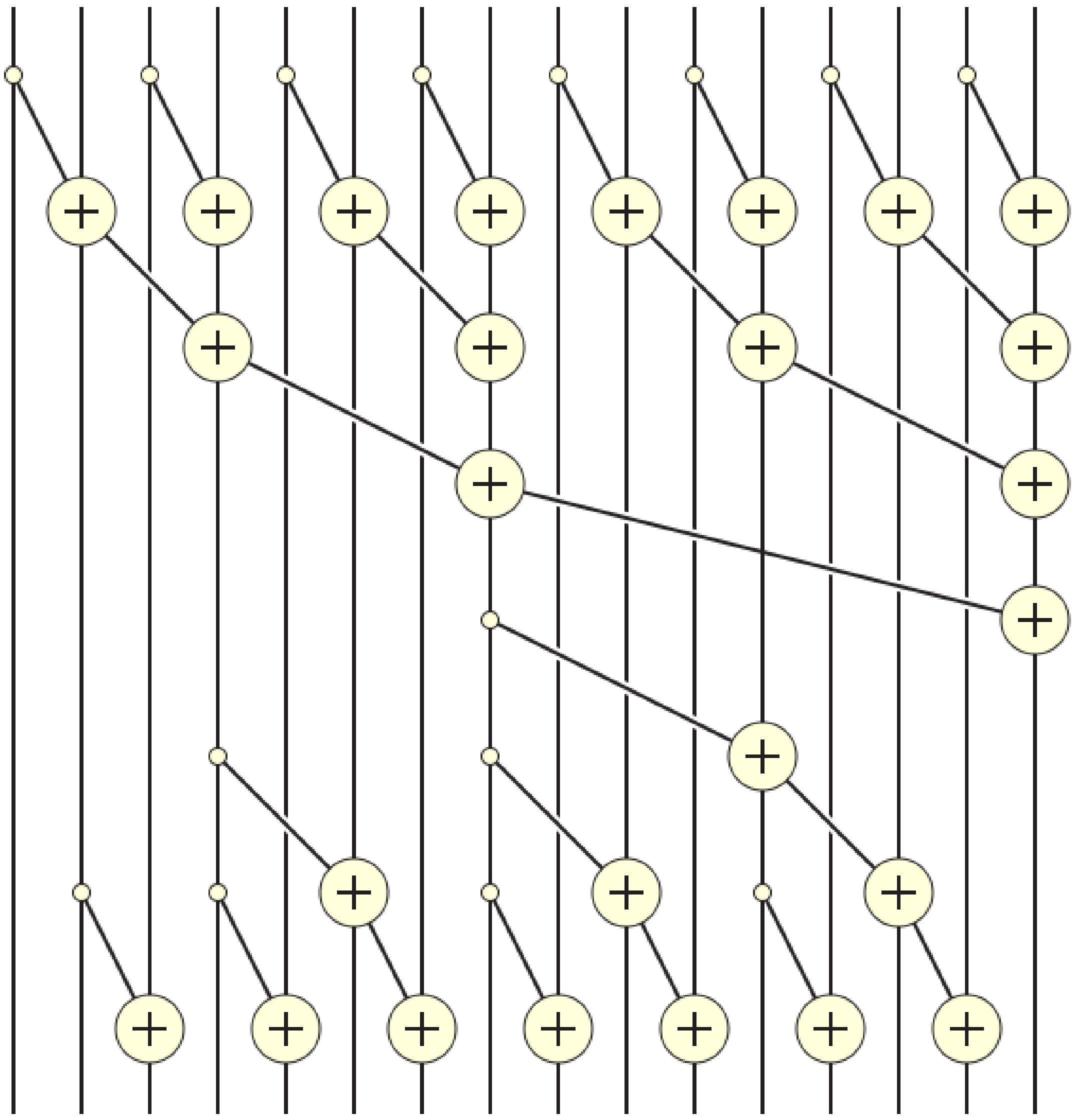
Prefixové súčty rekurzívne

```
{ pre n = 2^k }
if n > 1 then
   for i := 1 to n/2 pardo
      Y[i] := A[2*i-1]+A[2*i];

Rekurzívne vypočítaj prefixovú sumu
   pre pole Y a ulož ju do poľa Z

S[1] := A[1];
for i := 2 to n pardo
      if i mod 2 = 0 then S[i] := Z[i/2]
                      else S[i] := Z[i/2] + A[i];
```

EREW
$T(n) = O(1) + T(n/2)$
$W(n) = O(n) + W(n/2)$
…
$T(n)$ $\in O(\log(n))$
$W(n)$ $\in O(n)$

$W(n) \in \Theta(T^*(n)) \rightarrow$ optimálny algoritmus

Prefixové súčty – priebeh rekurzie



Prefixové súčty nerekurzívne

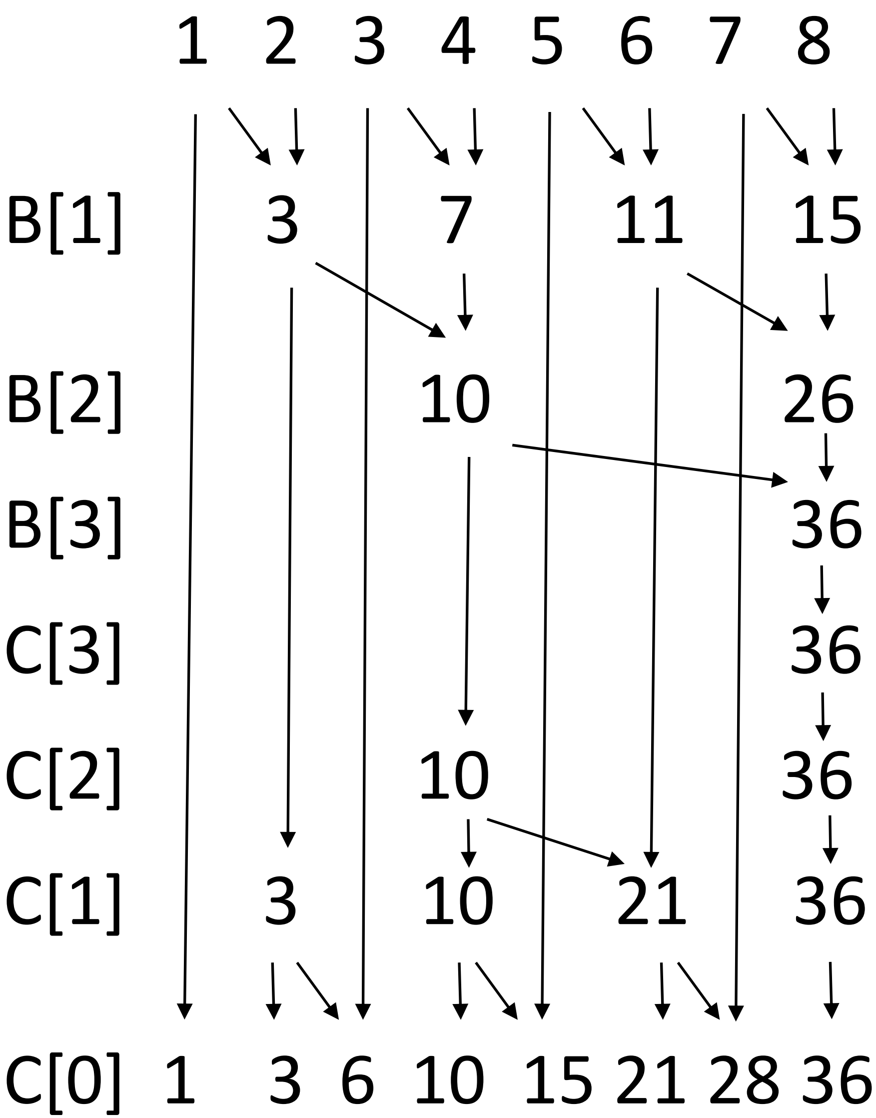
```
for i := 1 to n pardo
   B[0, i] := A[i];
for h:=1 to log(n) do
   for i:=1 to n/2^h pardo
       B[h, i] := B[h-1, 2*i-1] + B[h-1, 2*i];
for h:=log(n) downto 0 do
  for i:=1 to n/2^h pardo
  if i = 1 then C[h, i] := B[h, i]
           else if i mod 2 = 0 then C[h, i] := C[h+1, i/2]
           else C[h, i] := C[h+1,(i-1)/2] + B[h, i];
S[i] := C[0, i];
```

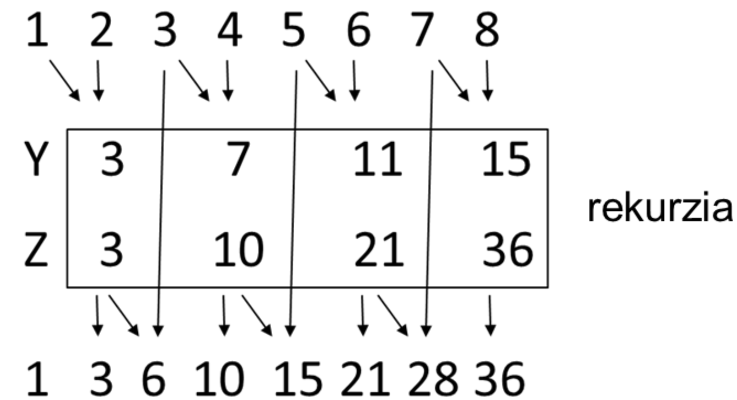
B[h, i] = i-ty prvok „vstupu“ (veľkosti n/2^h) pri h-tom vnorení rekurzie

C[h, i] = i-ty prvok „výstupu“ pri h-tom vnorení rekurzie

C[h, ...] je prefixová suma pre B[h, ...]

nerekurzívny prepis






```
{ nerekurzívne bez pomocných polí }
for i := 1 to n pardo S[i] := A[i];
for h:=1 to log(n) do
    for i:=1 to n/2^h pardo
      S[i*2^h] := S[i*2^h] + S[i*2^h-2^(h-1)];
for h:=log(n)-1 downto 1 do
  for i:=1 to (n/2^h)-1 pardo
    S[i*2^h+2^(h-1)] := S[i*2^h+2^(h-1)] + S[i*2^h];
```

Ďakujem za pozornosť !