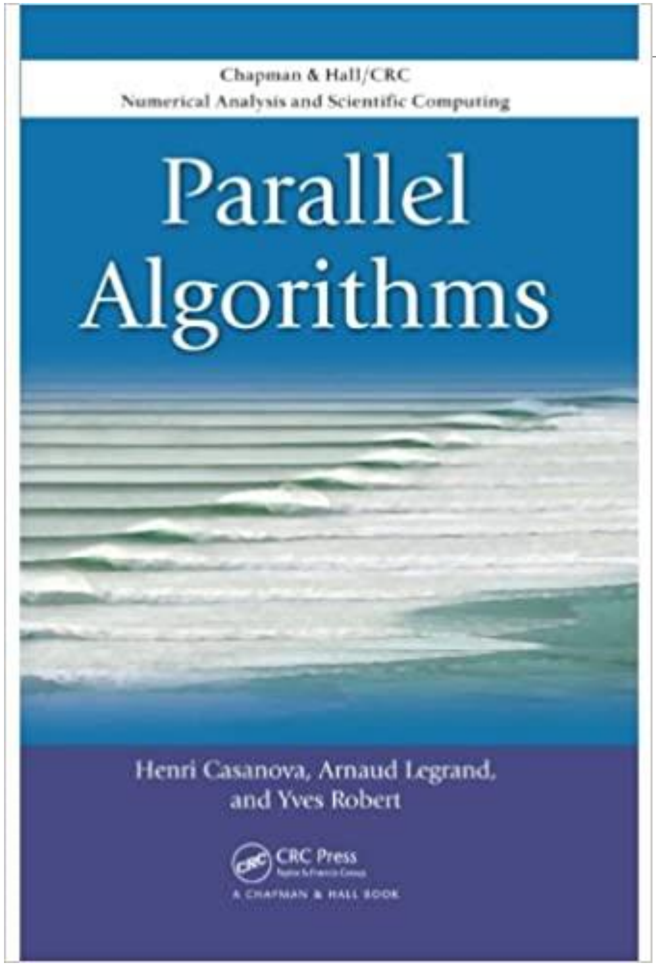
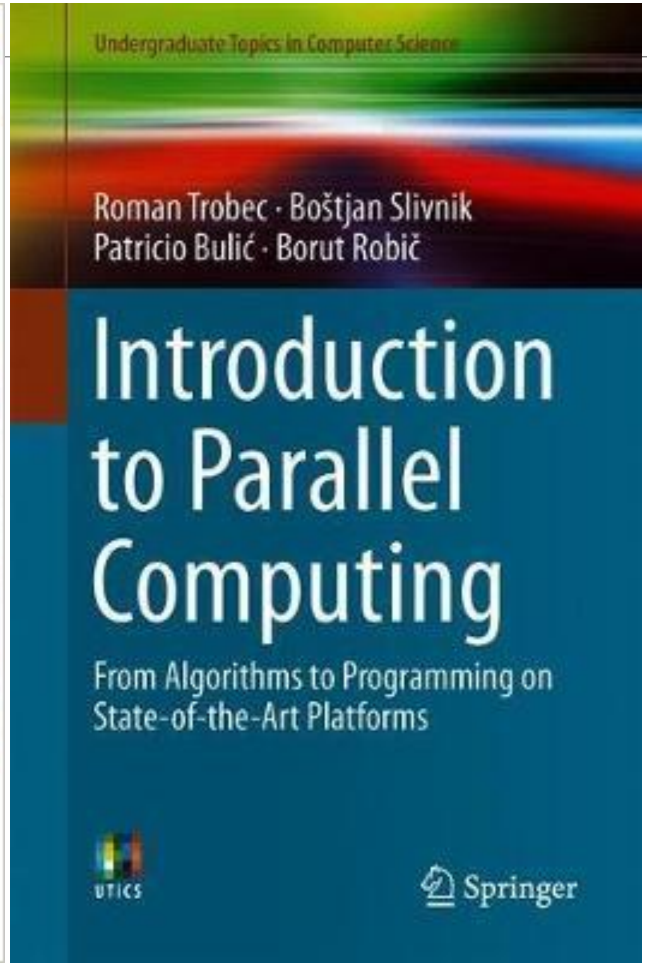
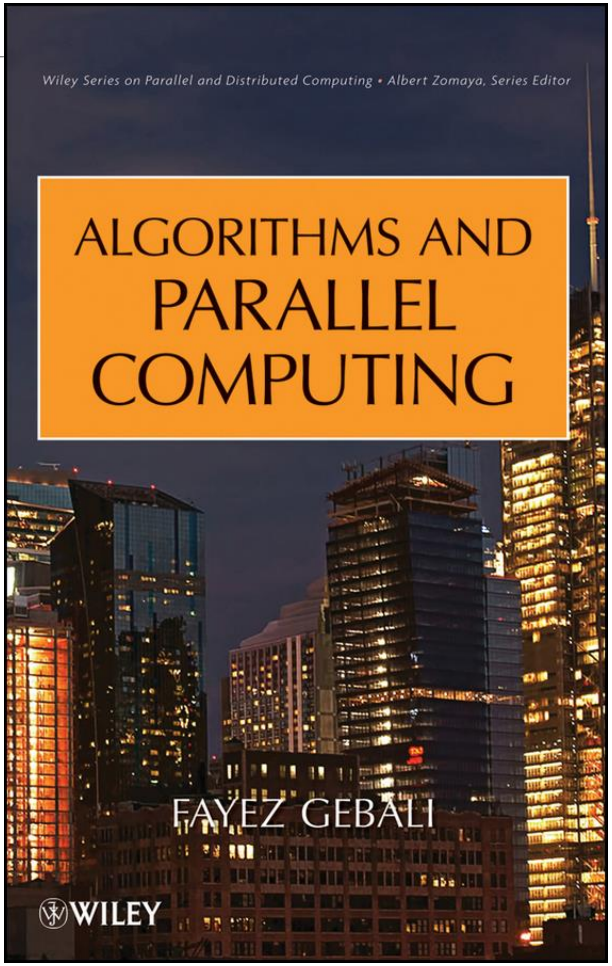
Parallel Programming

Lec 3

Books

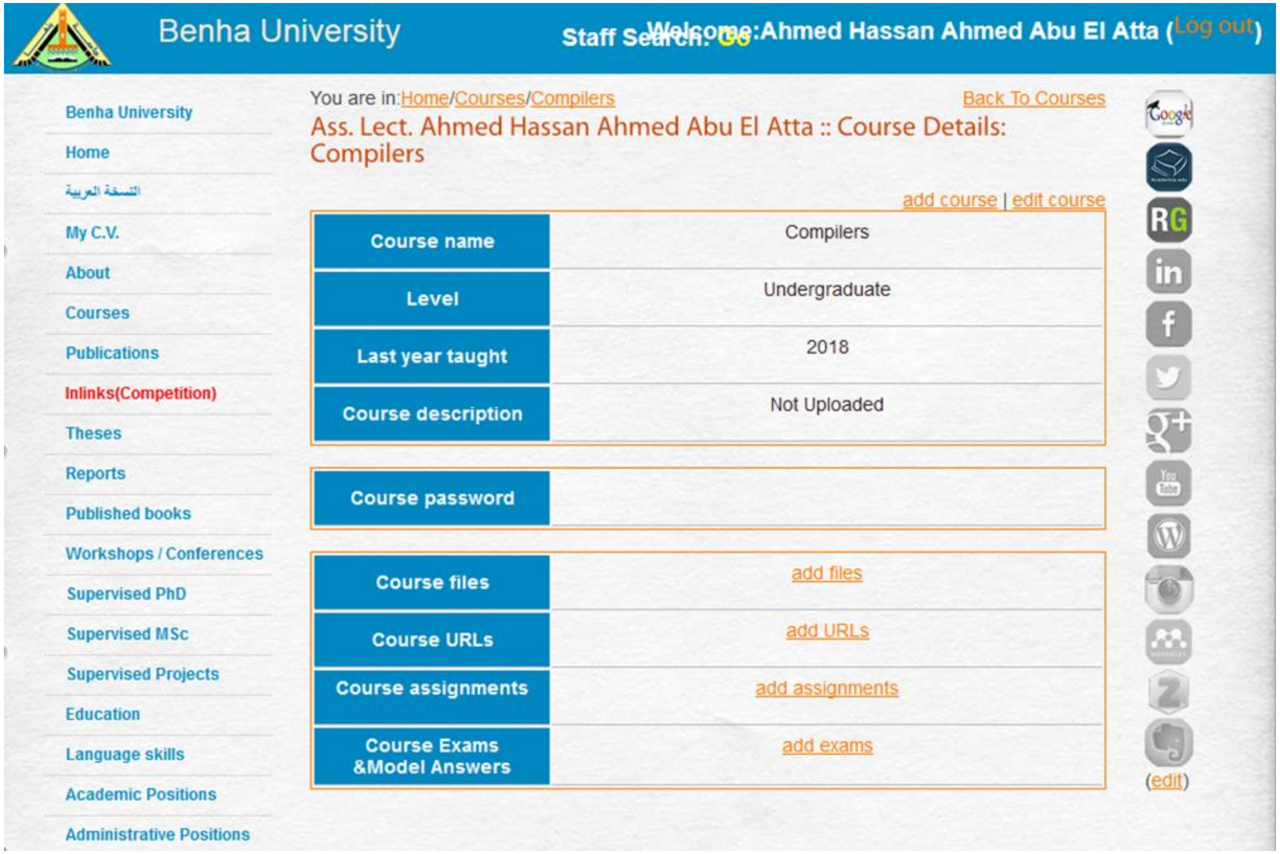

Chapman & Hall/CRC
Numerical Analysis and Scientific Computing
Parallel Algorithms
Henri Casanova, Arnaud Legrand, and Yves Robert
CRC Press
A CHAPMAN & HALL BOOK


Undergraduate Topics in Computer Science
Roman Trobec · Boštjan Slivnik
Patricio Bulić · Borut Robič
Introduction to Parallel Computing
From Algorithms to Programming on State-of-the-Art Platforms
UTICS
Springer


Wiley Series on Parallel and Distributed Computing • Albert Zomaya, Series Editor
ALGORITHMS AND PARALLEL COMPUTING
FAYEZ GEBALI
WILEY

PowerPoint

http://www.bu.edu.eg/staff/ahmedaboalatah14-courses/14779



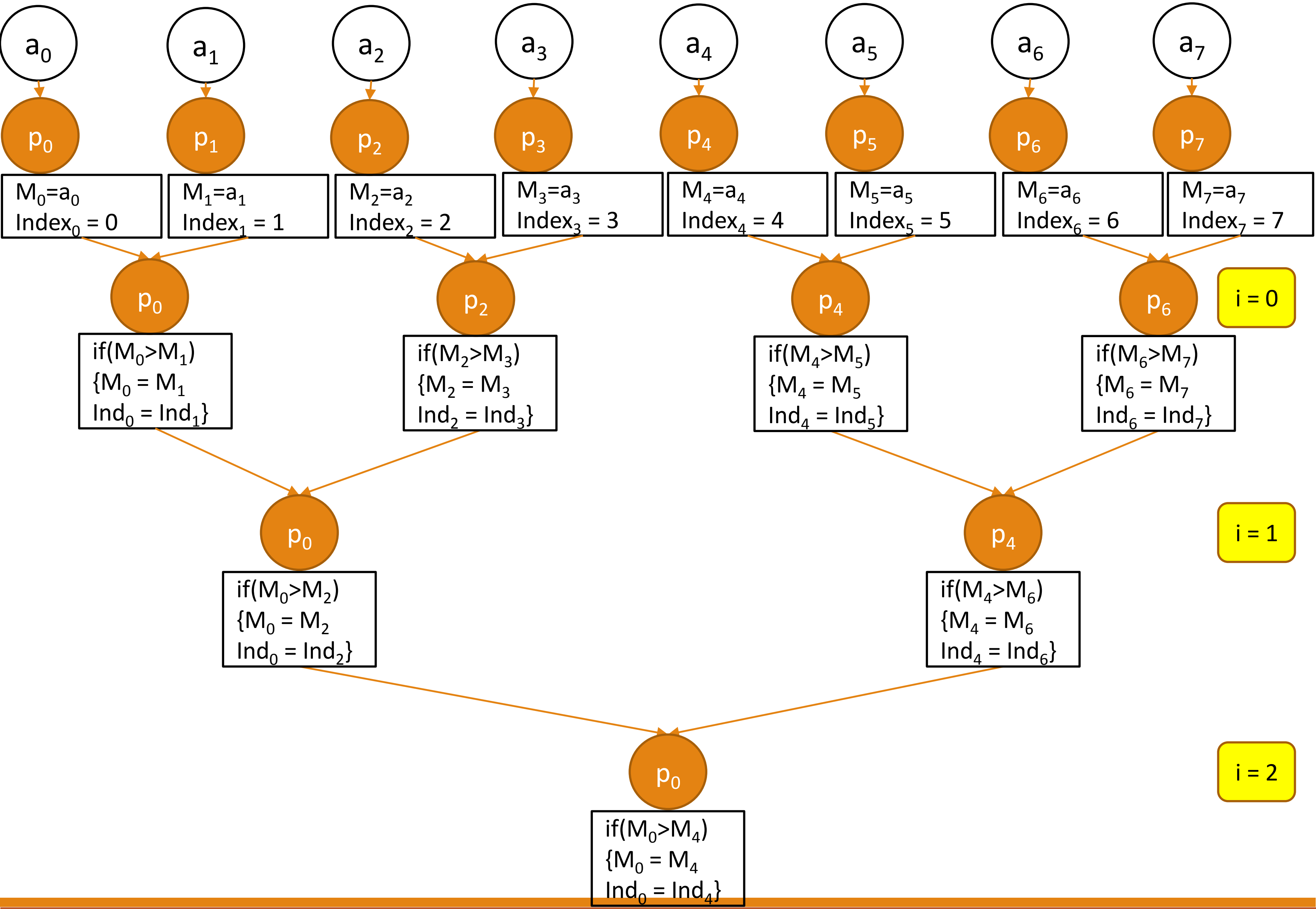
Find the minimum value in an array of integer numbers

Suppose that we are given the problem P ≡ “Find minimum in “n” given numbers.”

The fastest sequential algorithm for finding the minimum number is :

```
Min = a_0
Index  = 0
for (i = 1; i ≤ n-1; i++)
        if(Min > a_i)
        {
                Min = a_i
                index = i
        }
```

$T_{seq}(n) = O(n)$


a0 a1 a2 a3 a4 a5 a6 a7
p0 p1 p2 p3 p4 p5 p6 p7
M0=a0 Index0 = 0
M1=a1 Index1 = 1
M2=a2 Index2 = 2
M3=a3 Index3 = 3
M4=a4 Index4 = 4
M5=a5 Index5 = 5
M6=a6 Index6 = 6
M7=a7 Index7 = 7
p0 p2 p4 p6
if(M0>M1) {M0 = M1 Ind0 = Ind1}
if(M2>M3) {M2 = M3 Ind2 = Ind3}
if(M4>M5) {M4 = M5 Ind4 = Ind5}
if(M6>M7) {M6 = M7 Ind6 = Ind7}
i = 0
p0 p4
if(M0>M2) {M0 = M2 Ind0 = Ind2}
if(M4>M6) {M4 = M6 Ind4 = Ind6}
i = 1
p0
if(M0>M4) {M0 = M4 Ind0 = Ind4}
i = 2

Find the minimum value in an array of integer numbers

For $(j = 0 ; j < n ; j{+}{+})$ do parallel

  $M_j = a_j$

  $index_j = j$

For $i = 0$ to $i < \log(n)$ do

  For $(j = 0 ; j < n ; j \mathrel{+}= 2^{(i+1)})$ do in parallel

    if$(M_j > M_{j+2^i})$

    {

      $M_j = M_{j+2^i}$

      $index_j = index_{j+2^i}$

    }

$Min = M_0$

$Index = index_0$

Find the minimum value in an array of integer numbers

In general, instances of size n of P can be solved in parallel time $T_{par} = O(logn)$

speedup is $S(n) = T_{seq}(n) / T_{par}(n) = O(n/logn)$.

Cost $C(n) = n*O(logn) = O(nlogn)$

$E(n) = T_{seq}(n) / C(n) = O(n / (nlogn)) = O(1/logn) < 1$

Reducing the Processors Number to Reach to More Efficient Parallel Algorithm

$E_p(n) = C_s(n)/C_p(n) = 1$

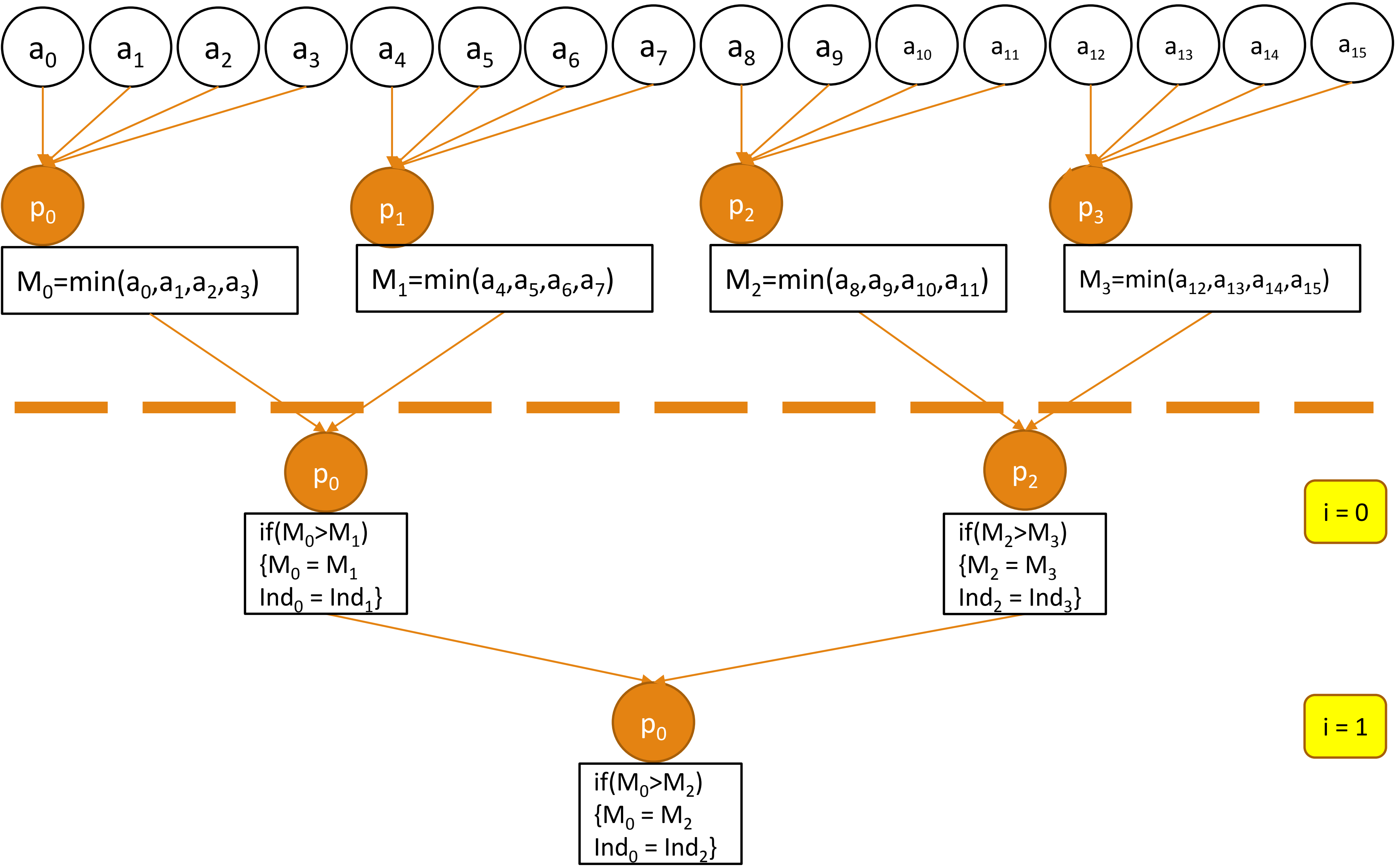
$C_s(n)/C_p(n) = 1 \quad \rightarrow \quad C_s(n) = C_p(n)$

$1*T_s(n) = p*T_p(n) \quad \rightarrow \quad p = T_s(n)/T_p(n)$

In the summation problem:

$p = T_s(n)/T_p(n) = n/\log(n)$

$$p = n/\log(n)$$


a0 a1 a2 a3 a4 a5 a6 a7 a8 a9 a10 a11 a12 a13 a14 a15
p0
p1
p2
p3
M0=min(a0,a1,a2,a3)
M1=min(a4,a5,a6,a7)
M2=min(a8,a9,a10,a11)
M3=min(a12,a13,a14,a15)
p0
p2
i = 0
if(M0>M1) {M0 = M1 Ind0 = Ind1}
if(M2>M3) {M2 = M3 Ind2 = Ind3}
p0
i = 1
if(M0>M2) {M0 = M2 Ind0 = Ind2}

More Efficient Algorithm

```
For (j = 0 ; j < n/log(n) ; j ++ )do parallel
        M_j = a_{j*log(n)}
        index_j = _{j*log(n)}
```

```
        For k = ((j*log(n))+1) to k < ((j+1)*log(n)) do
                if(M_j > a_k)
                {
                        M_j = a_k
                        index_j = index_k
                }

For i = 0 to i<log(n/log(n)) do
        For (j = 0 ; j < n ; j += 2^(i+1)) do in parallel
                if(M_j > M_{j+2^i})
                {
                        M_j = M_{j+2^i}
                        index_j = index_{j+2^i}
                }

Min = M_0

Index = index_0
```

More Efficient Algorithm

In general, instances of size n of P can be solved in parallel time $T_{par} = O(\log n)$ with number of processors equals $p = n/\log(n)$

speedup is $S(n) = T_{seq}(n) / T_{par}(n) = O(n/\log n)$.

Cost $C(n) = (n/\log(n))*O(\log n) = O(n)$

$E(n) = T_{seq}(n) / C(n) = O(n / n) = 1$

Search for a key value in an array of integer numbers

Suppose that we are given the problem P ≡ "search for a key value x in "n" given numbers."

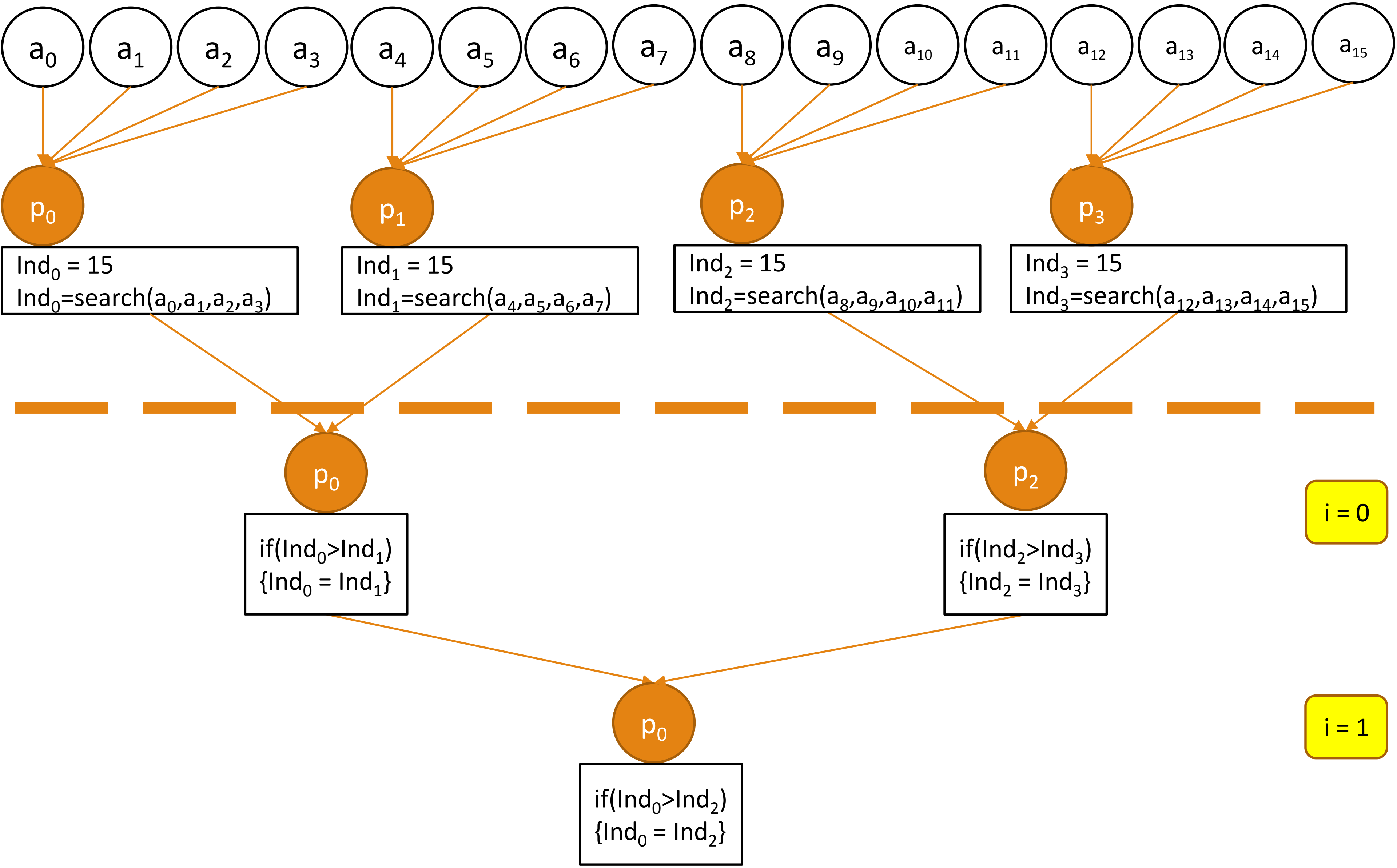
The fastest sequential algorithm for finding x in the array is :

```
Index  = -1 // can be replaced by size of array + 1 to find minimum
for (i = 0; i ≤ n-1; i++)
        if(a_i == x)
       {
                index = i
                break;
        }
```

If(Index != -1)then the number is found in the position = index

Else the number is not found

$T_{seq}(n) = O(n)$



a_0 a_1 a_2 a_3 a_4 a_5 a_6 a_7 a_8 a_9 a_{10} a_{11} a_{12} a_{13} a_{14} a_{15}
p_0
$Ind_0 = 15$
$Ind_0=search(a_0,a_1,a_2,a_3)$
p_1
$Ind_1 = 15$
$Ind_1=search(a_4,a_5,a_6,a_7)$
p_2
$Ind_2 = 15$
$Ind_2=search(a_8,a_9,a_{10},a_{11})$
p_3
$Ind_3 = 15$
$Ind_3=search(a_{12},a_{13},a_{14},a_{15})$
p_0
if(Ind_0>Ind_1)
{$Ind_0 = Ind_1$}
p_2
if(Ind_2>Ind_3)
{$Ind_2 = Ind_3$}
i = 0
p_0
if(Ind_0>Ind_2)
{$Ind_0 = Ind_2$}
i = 1

Efficient Search Algorithm

```
For (j = 0 ; j < n/log(n) ; j ++ )do parallel
        index_j = n
        For k = (j*log(n) to k < ((j+1)*log(n)) do
                if(a_k == X)
                {
                        index_j = k
                        break
                }
For i = 0 to i<log(n/log(n)) do
        For (j = 0 ; j < n ; j += 2^(i+1)) do in parallel
                if(index_j > index_{j+2^i})
                {
                        index_j = index_{j+2^i}
                }
Index = index_0
If(Index != -1 )then the number is found in the position = index
Else the number is not found
```

Search Algorithm

In general, instances of size n of P can be solved in parallel time $T_{par} = O(\log n)$ with number of processors equals $p = n/\log(n)$

speedup is $S(n) = T_{seq}(n) / T_{par}(n) = O(n/\log n)$.

Cost $C(n) = (n/\log(n))*O(\log n) = O(n)$

$E(n) = T_{seq}(n) / C(n) = O(n / n) = 1$