

6.23 Modi...ed

a.CRCW.

```
/******1*****
```

Assumptions:

We have n unsorted values $d_1; d_2; \dots; d_n$ with no repetitions.

There are n^2 processors arranged as a $n \times n$ matrix.

Column i of processors is going to calculate the sorted rank of d_i .

In the shared memory we have:

$D=[d_i]_{1 \leq i \leq n}$: Array with the unsorted values

$R=[r_i]_{1 \leq i \leq n}$: Array with sorted ranks.

The policy for concurrent writing is SUM, i.e.,

when more than one processor tries to write

a memory position the sum of the values is written

```
*****/
```

```
ENUMSORT(row,column)
```

```
/* label is the processor id */
```

```
begin
```

```
  if (D[row] < D[column]) then R[column] =1;
```

```
  else R[column] =0;
```

```
end.
```

$$T_s^a(n; c) = O(n \log n)$$

$$T_p(n; c) = O(1)$$

$$S(n; c) = \frac{T_s^a(n; c)}{T_p(n; c)} = \frac{O(n \log n)}{O(1)} = O(n \log n)$$

$$W(n; c) = P T_p(n; c) = n^2 O(1) = O(n^2)$$

$$E(n; c) = \frac{S}{p} = \frac{O(n \log n)}{n^2} = O\left(\frac{\log n}{n}\right)$$

```

c.CREW
/*****1*****/
Assumptions:
We have n unsorted values d1; d2; :::; dn with no repetitions.
There are n2 processors arranged as a n × n matrix.
Column i of processors is going to calculate the sorted rank of di.
In the shared memory we have:
D=[di]1 ≤ i ≤ n: Array with the unsorted values
R=[ri]1 ≤ i ≤ n: Array with sorted ranks.
RR=[rij]n × n: Auxiliar array to calculate the sorted ranks.
*****/
ENUMSORT(row,column)
/* label is the processor id */
begin
  if (D[row] < D[column]) then RR[column,column] = 1;
  else R[column,column] = 0;
  SUM(RR[1::n,column],R[column]);
end.

```

$$\begin{aligned}
T_s^a(n; c) &= O(n \log n) \\
T_p(n; c) &= O(\log n) \\
S(n; c) &= \frac{T_s^a(n; c)}{T_p(n; c)} = \frac{O(n \log n)}{O(\log n)} = O(n) \\
W(n; c) &= P T_p(n; c) = n^2 O(\log n) = O(n^2 \log n) \\
E(n; c) &= \frac{S}{p} = \frac{O(n^2 \log n)}{n^2} = O(\log n)
\end{aligned}$$

```

c.EREW
/*****1*****/
Assumptions:
We have n unsorted values d1; d2; :::; dn with no repetitions.
There are n2 processors arranged as a n × n matrix.
Column i of processors is going to calculate the sorted rank of di.
In the shared memory we have:
D=[di]1 ≤ i ≤ n: Array with the unsorted values
DD[dij]1 ≤ i ≤ n: Each column of this array is a copy of D.
R=[rri]1 ≤ i ≤ n: Array with sorted ranks.
RR=[rij]1 ≤ i ≤ n: Auxiliar array to calculate the sorted ranks.
The policy for concurrent writing is SUM, i.e.,
when more than one processor tries to write
a memory position the sum of the values is written
*****/
ENUMSORT(row,column)
/* label is the processor id */
begin
  /* The ..rst processor of each column makes a copy in a the
  corresponding column of DD, each one of these
  starts reading D in a different position to avoid concurrent readings*/
  if ( row == 1) then
    begin
      for i=0 to n-1 do
        DD[(column+ i) mod n,column] =D[(column+i) mod n];
      end;
    end
    if (DD[row,column] < DD[column,column]) then RR[column,column] =1;
    else R[column,column] =0;
    SUM(RR[1 to n,column],R[column]);
  end.

```

$$\begin{aligned}
T_s^a(n; c) &= O(n \log n) \\
T_p(n; c) &= O(n) \\
S(n; c) &= \frac{T_s^a(n; c)}{T_p(n; c)} = \frac{O(n \log n)}{O(n)} = O(\log n) \\
C(n; c) &= P T_p(n; c) = n^2 O(n) = O(n^3) \\
E(n; c) &= \frac{S}{p} = \frac{O(\log n)}{n^2} = O\left(\frac{\log n}{n^2}\right)
\end{aligned}$$

9.1 Modi...ed

a.1

V_1	V_2	V_3	V_4	$\sum_{i=1}^4 V_i w_i \cdot 6$	$C = \sum_{i=1}^4 V_i p_i$	optimun
0	0	0	0	0 · 6	0	
0	0	0	1	1 · 6	5	
0	0	1	0	2 · 6	2	
0	0	1	1	3 · 6	7	
0	1	0	0	3 · 6	3	
0	1	0	1	4 · 6	8	
0	1	1	0	5 · 6	5	
0	1	1	1	6 · 6	10	J
1	0	0	0	5 · 6	1	
1	0	0	1	6 · 6	2	
1	0	1	0	7 £ 6F	i	
1	0	1	1	8 £ 6F	i	
1	1	0	0	8 £ 6F	i	
1	1	0	1	9 £ 6F	i	
1	1	1	0	10 £ 6F	i	
1	1	1	1	11 £ 6F	i	

a.2

nnc	2	1	2	3	4	5	6	3
4		5	5	7	8	8	10	
3	6	0	2	3	3	5	5	7
2	4	0	0	3	3	3	3	5
1		0	0	0	0	1	1	

Solution pack: 4,3,2.

```

b.
/*****1*****/
Assumptions:
We have n objects and a knapsack with capacity c.
There are c processors labelled P1; P2; ; ; Pc
In the shared memory we have:
W=[wi]1∈n: Array with the weight of the objects
P=[pi]1∈n: Array with the price of the objects
F=[Fij]n∈c: Array with the values of F
*****/
KNAPSACK(label)
/* label is the processor id */
begin
for i=1 to c do
begin
F[i,label]= Max{F_value(i-1,label), F_value(i-1,label-W[i]) + P[i]};
end;
end.
function F_value(i,j)
begin
if (i == 0)then
begin
if (j >= 0) then return 0;
else return j - 1 ;
end;
else
begin
if (j == 0) then return 0;
else if (j < 0) then return j - 1 ;
else return F[i,j];
end;
end.
end.

```

$$\begin{aligned}
T_s^n(n; c) &= O(nc) \\
T_{p=c}(n; c) &= O(n) \\
S(n; c) &= \frac{T_s^n(n; c)}{T_p(n; c)} = \frac{O(nc)}{O(n)} = O(c) \\
W(n; c) &= P T_p(n; c) = cO(n) = O(nc) \\
E(n; c) &= \frac{S}{p} = \frac{O(c)}{c} = O(1)
\end{aligned}$$