HETEROGENEOUS PARALLELISM PROJECT FINAL REVIEW

NISHANTH NARENDRA – PES1UG19CS305

Multiplying an n x n matrix with a vector of length n

```
Mat_Vec(A, x)
    n = A.rows
    let b be a new vector of length n
    for i = 0 to n – 1
        b[i] = 0
        for j = 0 to n – 1
            b[i] += A[i][j] * x[j]
    return b
```

```
Mat_Vec(A, x)
    n = A.rows
    let b be a new vector of length n
    parallel for i = 0 to n – 1
        new dot_prod = 0
        parallel for new j = 0 to n – 1
            dot_prod += A[i][j] * x[j]
        b[i] = dot_prod
    return b
```

$$Serial\ run\ time = \Theta(n^2)$$

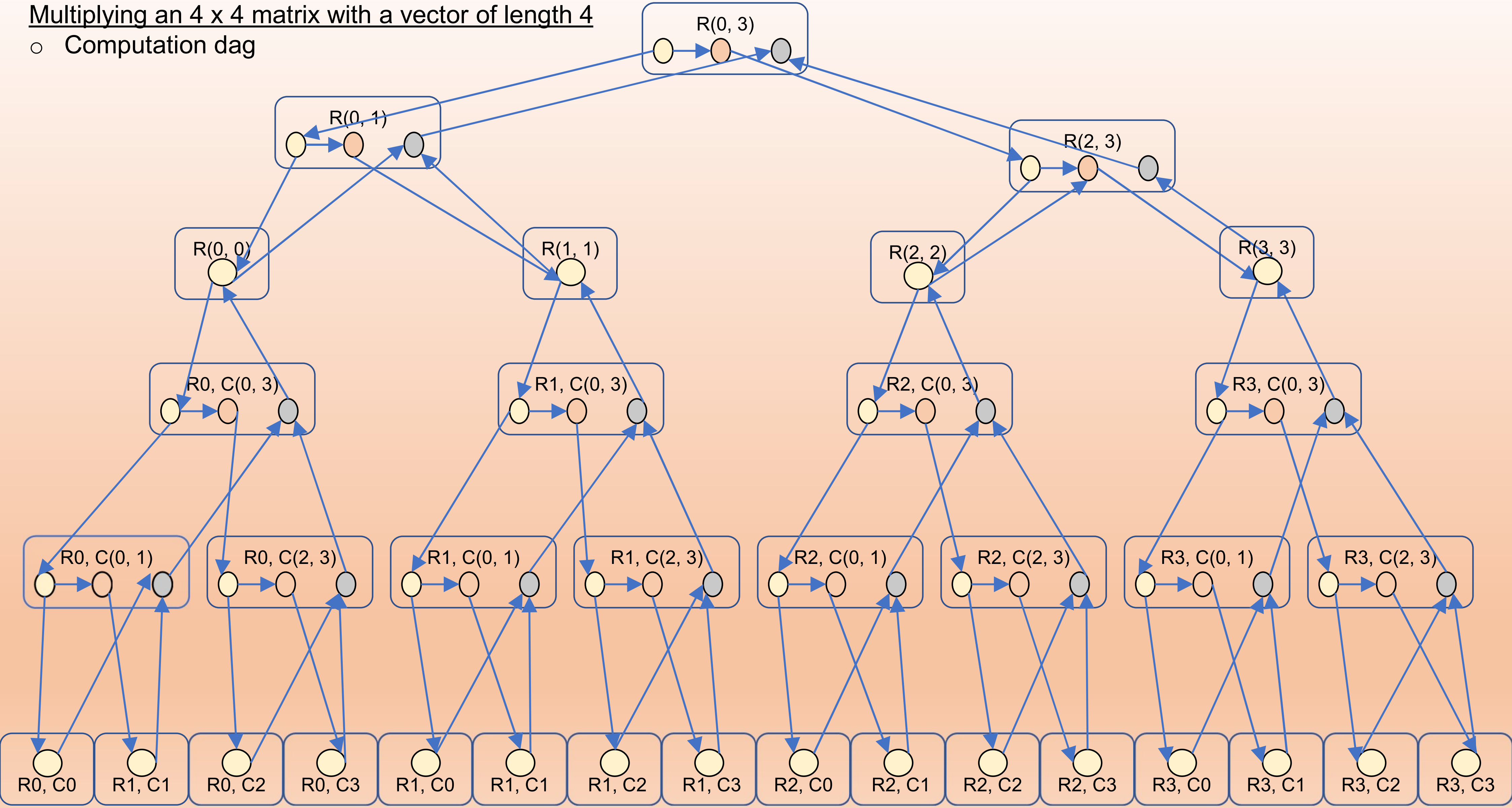
$$Work = T_1 = \Theta(n^2)$$

$$Span = T_\infty = \Theta(lg\, n)$$

$$Parallelism = \Theta\left(\frac{n^2}{lg\, n}\right)$$

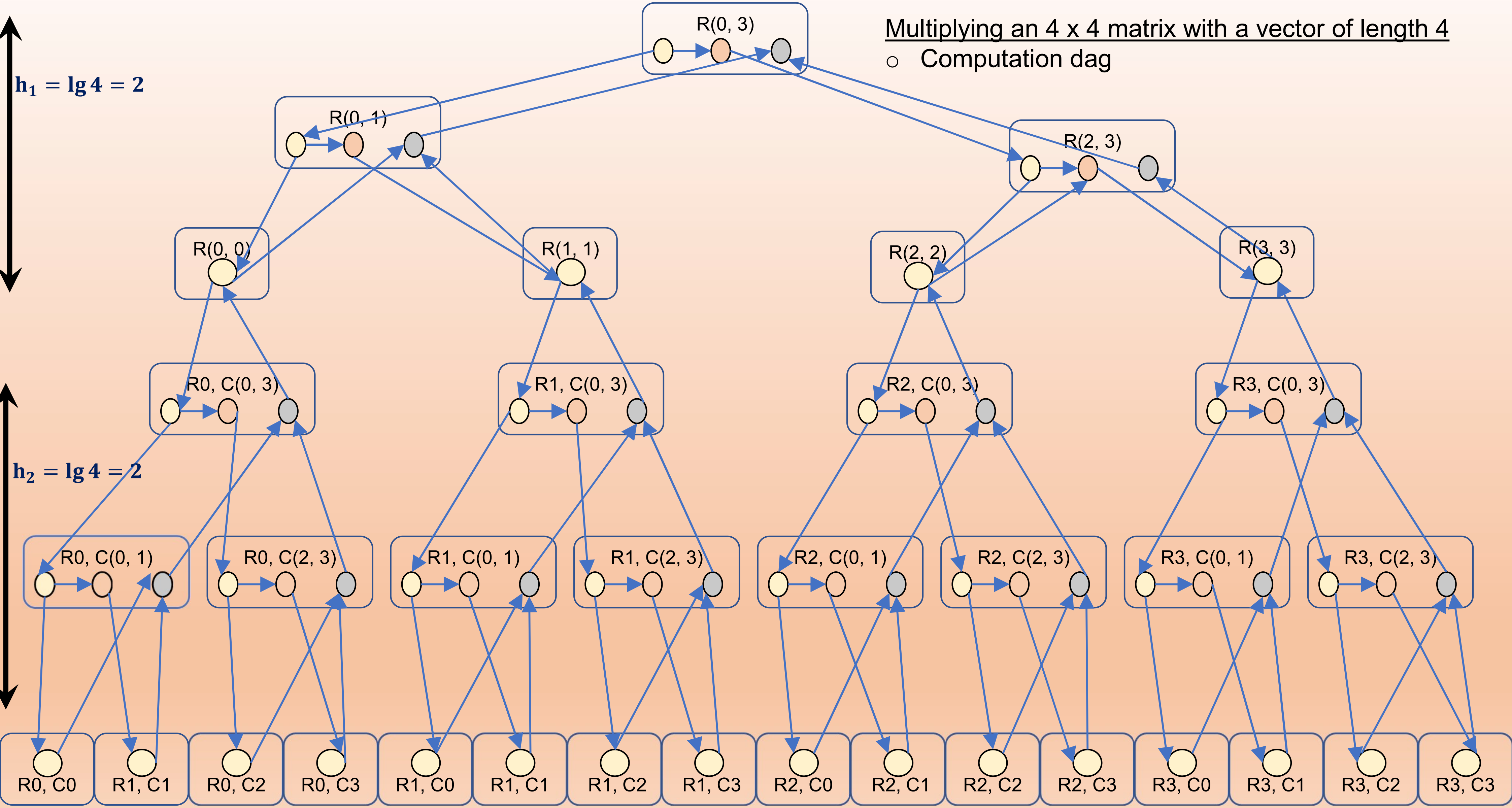
Multiplying an 4 x 4 matrix with a vector of length 4

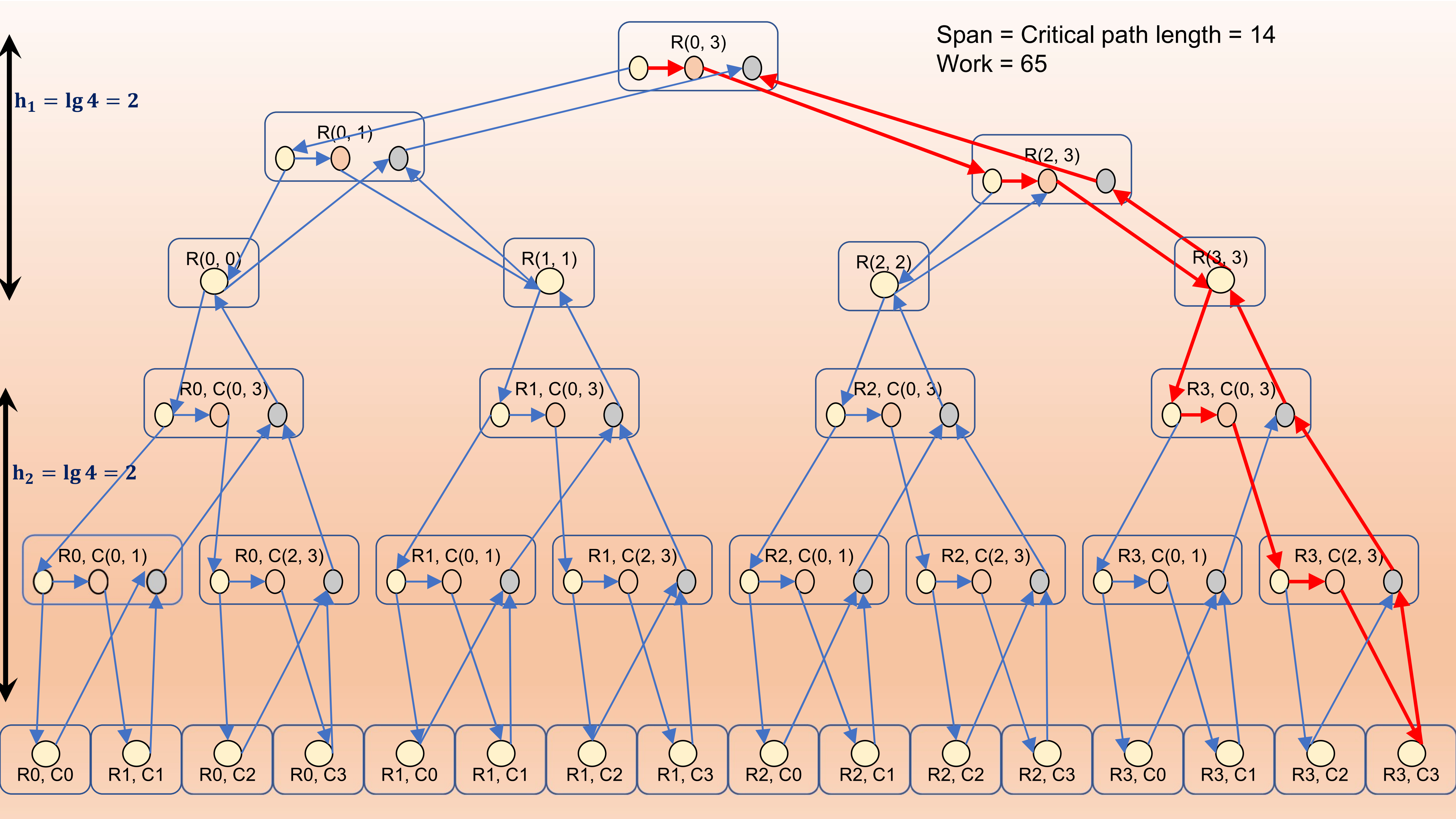
- Computation dag



Multiplying an 4 x 4 matrix with a vector of length 4

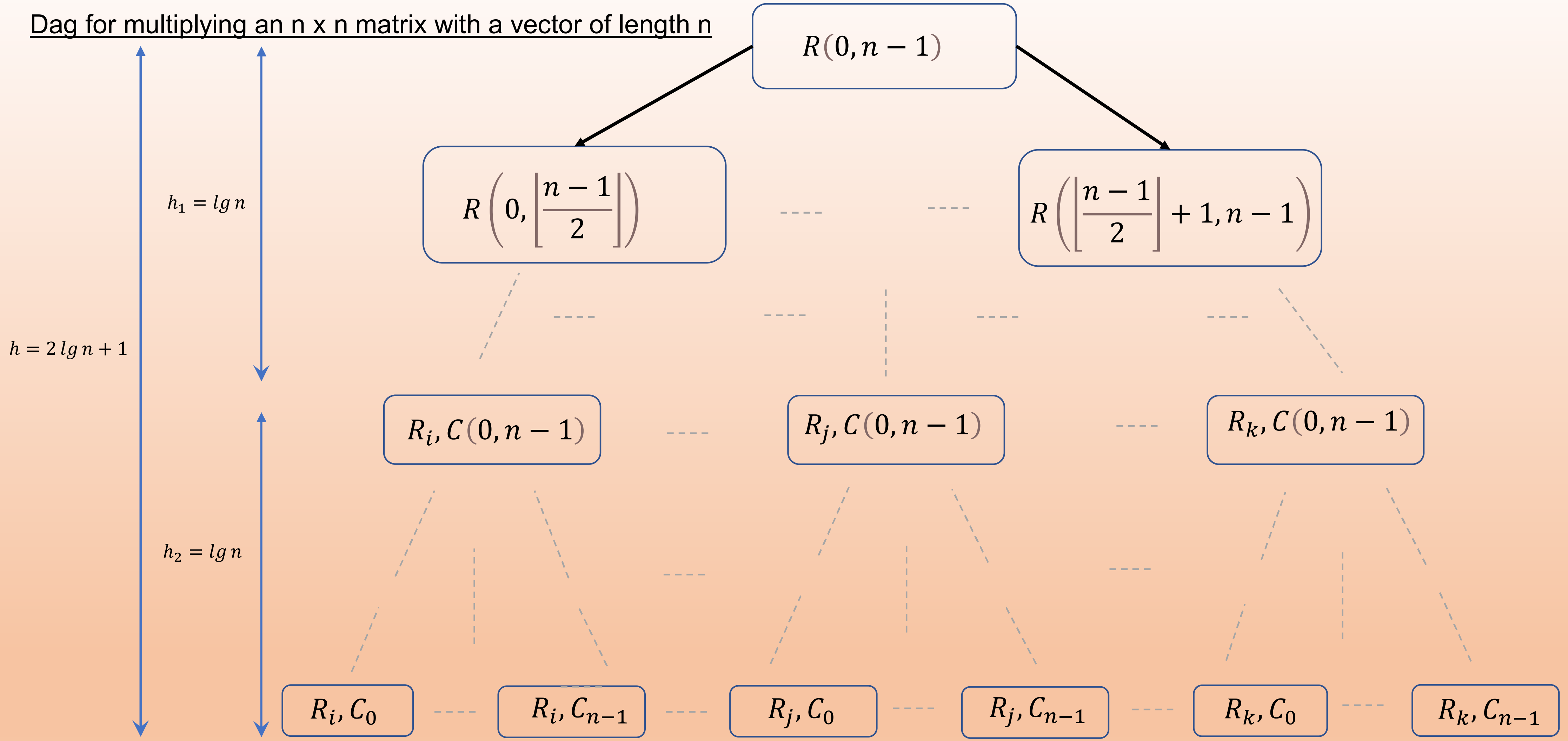
- Computation dag





Span = Critical path length = 14
Work = 65
$\mathbf{h_1 = lg\,4 = 2}$
$\mathbf{h_2 = lg\,4 = 2}$
R(0, 3)
R(0, 1)
R(2, 3)
R(0, 0)
R(1, 1)
R(2, 2)
R(3, 3)
R0, C(0, 3)
R1, C(0, 3)
R2, C(0, 3)
R3, C(0, 3)
R0, C(0, 1)
R0, C(2, 3)
R1, C(0, 1)
R1, C(2, 3)
R2, C(0, 1)
R2, C(2, 3)
R3, C(0, 1)
R3, C(2, 3)
R0, C0
R1, C1
R0, C2
R0, C3
R1, C0
R1, C1
R1, C2
R1, C3
R2, C0
R2, C1
R2, C2
R2, C3
R3, C0
R3, C1
R3, C2
R3, C3

Dag for multiplying an n x n matrix with a vector of length n



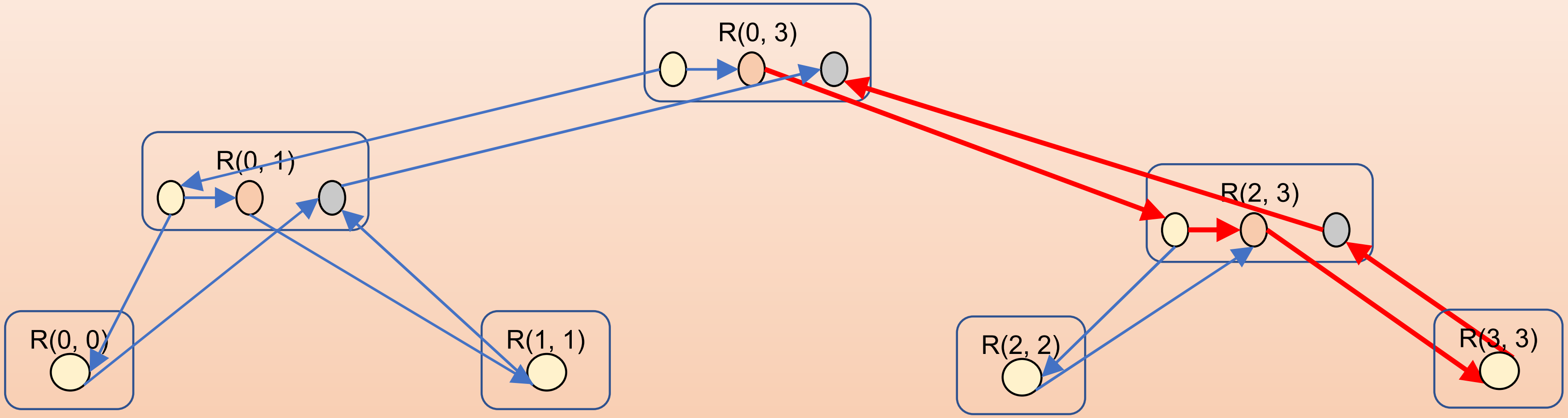
$Work = T_1 = \Theta(n^2)$

$Span = T_\infty = \Theta(lg\, n)$

$Parallelism = \Theta\left(\frac{n^2}{lg\, n}\right)$

Multiplying an 4 x 4 matrix with a vector of length 4

- Computation dag with the leaves **coarsened**



In general, for nxn mat-vec by **coarsening the base case** by performing the whole dot product in the leaves:

$$\text{Work} = T_1 = \Theta(n^2)$$

$$Span = T_\infty = \Theta(l\,g\,n) + \Theta(n) = \Theta(n)$$

$$Parallelism = \Theta(n)$$

Coarsening a parallel loop using grain size G

```
parallel for i = 0  to n -1 : grain_size = G
        A[i] += B[i]
```

In Cilk/Cilk++

```
#pragma cilk: grain_size = G
cilk_for (int i = 0;  i < n; ++i)
        A[i] += B[i]
```

$$T_1 = n \cdot t_{iter} + \left(\frac{n}{G} - 1\right) \cdot t_{spawn}$$

$$T_\infty = G \cdot t_{iter} + lg\left(\frac{n}{G}\right) \cdot t_{spawn}$$

$$For\ work\ I\ want:\ G \gg \frac{t_{spawn}}{t_{iter}}$$

$$For\ span\ I\ want\ :\ G\ small$$

All Pairs Shortest Paths problem – Parallel Floyd-Warshall

```
Floyd_Warshall(W)
    n = W.rows
    let D and P be new matrices of size n x n
    for i = 0 to n – 1
        for j = 0 to n – 1
            D[i][j] = W[i][j]
            if i == j or D[i][j] == ∞
                P[i][j] = -1
            else
                P[i][j] = i
    for k = 0 to n – 1
        for i = 0 to n – 1
            for j = 0 to n – 1
                if D[i][j] > D[i][k] + D[k][j]
                    D[i][j] = D[i][k] + D[k][j]
                    P[i][j] = P[k][j]
    return (D, P)
```

$$Serial\ run\ time = \Theta(V^3)$$

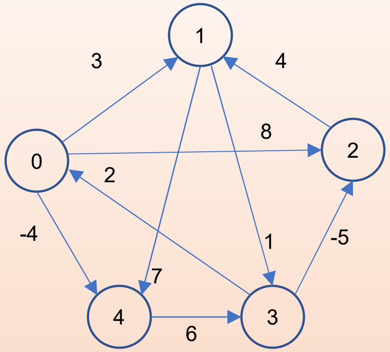
```
Parallel_Floyd_Warshall(W)
    n = W.rows
    let D and P be new matrices of size n x n
    parallel for i = 0 to n – 1
        parallel for new  j = 0 to n – 1
            D[i][j] = W[i][j]
            if i == j or D[i][j] == ∞
                P[i][j] = -1
            else
                P[i][j] = i
    for k = 0 to n – 1
        parallel for i = 0 to n – 1
            parallel for new j = 0 to n – 1
                if D[i][j] > D[i][k] + D[k][j]
                    D[i][j] = D[i][k] + D[k][j]
                    P[i][j] = P[k][j]
    return (D, P)
```

$$Work = \Theta(V^3)$$

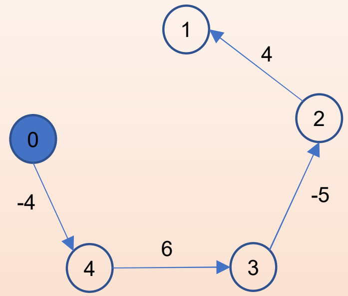
$$Span = \Theta(V\ lg\ V)$$

$$\text{Parallelism} = \frac{\text{Work}}{\text{Span}} = \frac{\Theta(\text{V}^3)}{\Theta(\text{V lg V})} = \Theta\left(\frac{\text{V}^2}{\text{lg V}}\right)$$

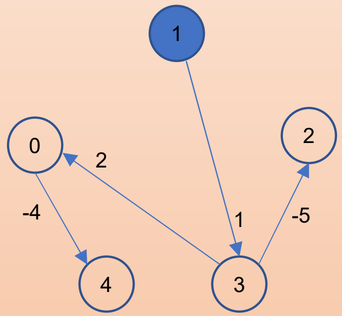
Floyd-Warshall – Example 1



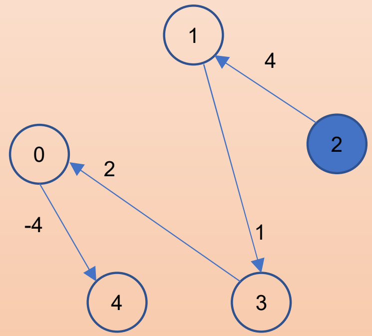
Source 0		
Destination	Shortest path weight	Shortest Path
0	0	<0>
1	1	<0, 4, 3, 2, 1>
2	-3	<0, 4, 3, 2>
3	2	<0, 4, 3>
4	-4	<0, 4>



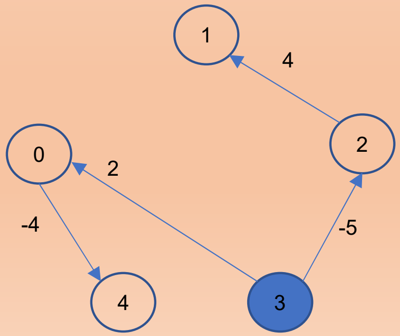
Source 1		
Destination	Shortest path weight	Shortest Path
0	0	<0>
1	1	<0, 4, 3, 2, 1>
2	-3	<0, 4, 3, 2>
3	2	<0, 4, 3>
4	-4	<0, 4>



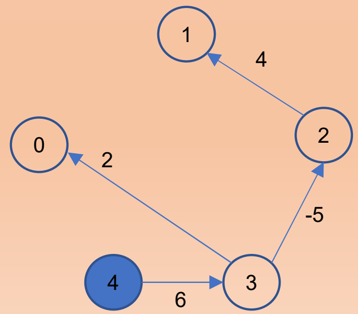
Source 2		
Destination	Shortest path weight	Shortest Path
0	7	<2, 1, 3, 0>
1	4	<2, 1>
2	0	<2>
3	5	<2, 1, 3>
4	3	<2, 1, 3, 0, 4>



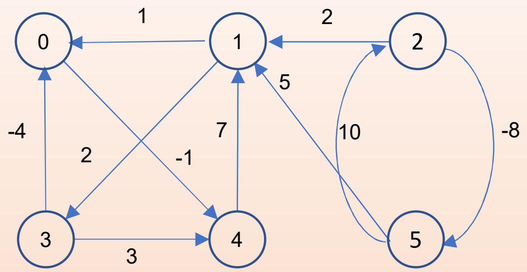
Source 3		
Destination	Shortest path weight	Shortest Path
0	2	<3, 0>
1	-1	<3, 2, 1>
2	-5	<3, 2>
3	0	<3>
4	-2	<3, 0, 4>



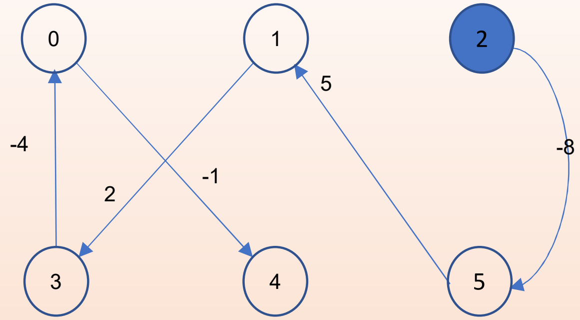
Source 4		
Destination	Shortest path weight	Shortest Path
0	8	<4, 3, 0>
1	5	<4, 3, 2, 1>
2	1	<4, 3, 2>
3	6	<4, 3>
4	-0	<4>



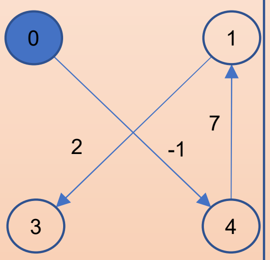
Floyd-Warshall – Example 2



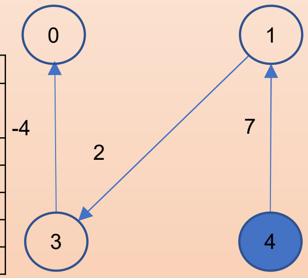
Source 0		
Destination	Shortest path weight	Shortest Path
0	0	<0>
1	1	<0, 4, 3, 2, 1>
2	-3	<0, 4, 3, 2>
3	2	<0, 4, 3>
4	-4	<0, 4>



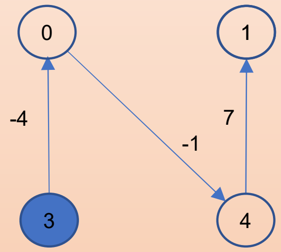
Source 0		
Destination	Shortest path weight	Shortest Path
0	0	<0>
1	1	<0, 4, 3, 2, 1>
2	-3	<0, 4, 3, 2>
3	2	<0, 4, 3>
4	-4	<0, 4>



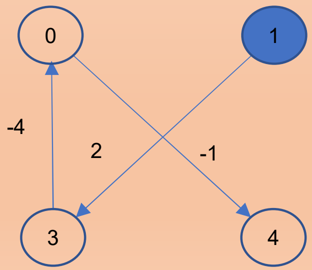
Source 0		
Destination	Shortest path weight	Shortest Path
0	0	<0>
1	1	<0, 4, 3, 2, 1>
2	-3	<0, 4, 3, 2>
3	2	<0, 4, 3>
4	-4	<0, 4>



Source 0		
Destination	Shortest path weight	Shortest Path
0	0	<0>
1	1	<0, 4, 3, 2, 1>
2	-3	<0, 4, 3, 2>
3	2	<0, 4, 3>
4	-4	<0, 4>



Source 0		
Destination	Shortest path weight	Shortest Path
0	0	<0>
1	1	<0, 4, 3, 2, 1>
2	-3	<0, 4, 3, 2>
3	2	<0, 4, 3>
4	-4	<0, 4>



Source 0		
Destination	Shortest path weight	Shortest Path
0	0	<0>
1	1	<0, 4, 3, 2, 1>
2	-3	<0, 4, 3, 2>
3	2	<0, 4, 3>
4	-4	<0, 4>