CSC 6220: Parallel Computing I: Programming
Homework 2
Fall 2023

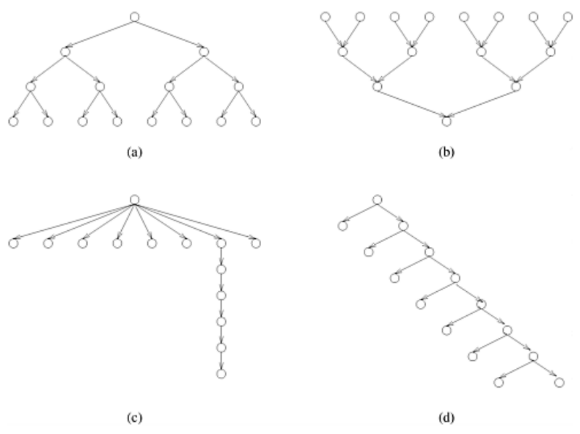
Student name: Kanchan Chopde

Problem 3.2,3.5,3.6,3.12 referred from Introduction to Parallel Computing, Second Edition by Ananth Grama et.al.

Problem 3.2

For the task graphs given in Figure 3.42, determine the following: 1. Maximum degree of concurrency. 2. Critical path length. 3. Maximum achievable speedup over one process assuming that an arbitrarily large number of processes is available. 4. The minimum number of processes needed to obtain the maximum possible speedup. 5. The maximum achievable speedup if the number of processes is limited to (a) 2, (b) 4, and (c) 8.

Figure 3.42. Task-dependency graphs for Problem 3.2.



Solution:

a)
1. Maximum degree of concurrency- 8
2. Critical path length- 4
3. Maximum achievable speedup over one process- 15/4 = 3.75
4. The minimum number of processes needed to obtain the maximum possible speedup- 8
5. The maximum achievable speedup if the number of processes is limited to

(a) 2- 15/8
(b) 4- 15/5 =3
(c) 8- 15/4 = 3.75

b)
1. Maximum degree of concurrency- 8
2. Critical path length- 4
3. Maximum achievable speedup over one process- 15/4 = 3.75
4. The minimum number of processes needed to obtain the maximum possible speedup- 8
5. The maximum achievable speedup if the number of processes is limited to

(a) 2- 15/8
(b) 4- 15/5 = 3
(c) 8- 15/4 = 3.75

c)
1. Maximum degree of concurrency- 8
2. Critical path length- 7
3. Maximum achievable speedup over one process- 14/7= 2
4. The minimum number of processes needed to obtain the maximum possible speedup- 3
5. The maximum achievable speedup if the number of processes is limited to
(a) 2- 7/4
(b) 4- 2
(c) 8- 2

d)
1. Maximum degree of concurrency -8
2. Critical path length - 8
3. Maximum achievable speedup over one process 15/8 = 1.875
4. The minimum number of processes needed to obtain the maximum possible speedup- 2
5. The maximum achievable speedup if the number of processes is limited to
(a) 2- 15/8=1.875
(b) 4- 15/8=1.875
(c) 8- 15/8=1.875

Problem 3.5

Consider LU factorization of a dense matrix shown in Algorithm 3.3. Figure 3.27 shows the decomposition of LU factorization into 14 tasks based on a two-dimensional partitioning of the matrix A into nine blocks Ai,j, 1 i, j 3. The blocks of A are modified into corresponding blocks of L and U as a result of factorization. The diagonal blocks of L are lower triangular submatrices with unit diagonals and the diagonal blocks of U are upper triangular submatrices. Task 1 factors the submatrix A1,1 using Algorithm 3.3. Tasks 2 and 3 implement the block versions of the loop on Lines 4–6 of Algorithm 3.3. Tasks 4 and 5 are the upper-triangular counterparts of tasks 2 and 3. The element version of LU factorization in Algorithm 3.3 does not show these steps because the diagonal entries of L are 1; however, a block version must compute a block-row of U as a product of the inverse of the corresponding diagonal block of L with the block-row of A. Tasks 6–9 implement the block version of the loops on Lines 7–11 of Algorithm 3.3. Thus, Tasks 1–9 correspond to the block version of the first iteration of the outermost loop of Algorithm 3.3. The remainder of the tasks complete the factorization of A. Draw a task-dependency graph corresponding to the decomposition shown in Figure 3.27

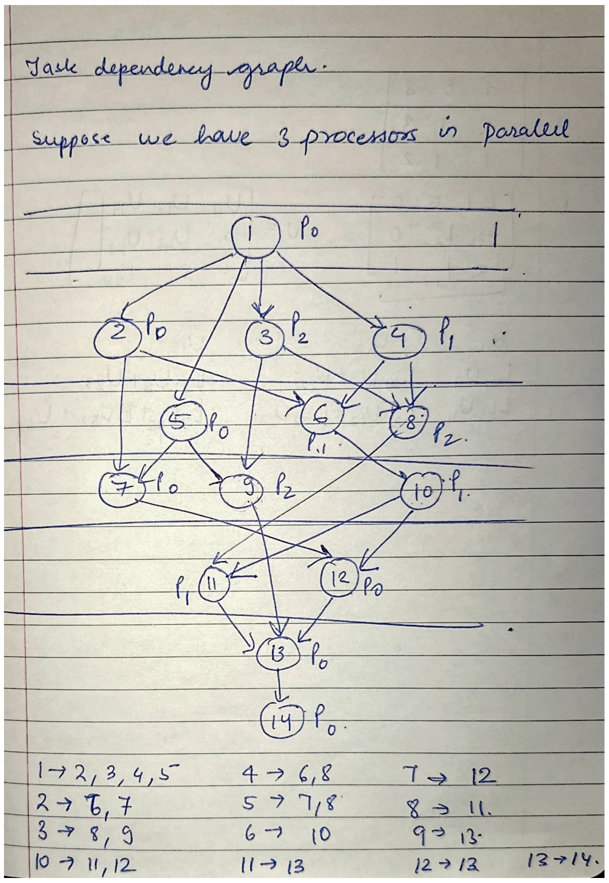
Figure 3.27. A decomposition of LU factorization into 14 tasks.

$$\begin{pmatrix} A_{1,1} & A_{1,2} & A_{1,3} \\ A_{2,1} & A_{2,2} & A_{2,3} \\ A_{3,1} & A_{3,2} & A_{3,3} \end{pmatrix} \rightarrow \begin{pmatrix} L_{1,1} & 0 & 0 \\ L_{2,1} & L_{2,2} & 0 \\ L_{3,1} & L_{3,2} & L_{3,3} \end{pmatrix} \cdot \begin{pmatrix} U_{1,1} & U_{1,2} & U_{1,3} \\ 0 & U_{2,2} & U_{2,3} \\ 0 & 0 & U_{3,3} \end{pmatrix}$$

1: $A_{1,1} \rightarrow L_{1,1}U_{1,1}$	6: $A_{2,2} = A_{2,2} - L_{2,1}U_{1,2}$	11: $L_{3,2} = A_{3,2}U_{2,2}^{-1}$
2: $L_{2,1} = A_{2,1}U_{1,1}^{-1}$	7: $A_{3,2} = A_{3,2} - L_{3,1}U_{1,2}$	12: $U_{2,3} = L_{2,2}^{-1}A_{2,3}$
3: $L_{3,1} = A_{3,1}U_{1,1}^{-1}$	8: $A_{2,3} = A_{2,3} - L_{2,1}U_{1,3}$	13: $A_{3,3} = A_{3,3} - L_{3,2}U_{2,3}$
4: $U_{1,2} = L_{1,1}^{-1}A_{1,2}$	9: $A_{3,3} = A_{3,3} - L_{3,1}U_{1,3}$	14: $A_{3,3} \rightarrow L_{3,3}U_{3,3}$
5: $U_{1,3} = L_{1,1}^{-1}A_{1,3}$	10: $A_{2,2} \rightarrow L_{2,2}U_{2,2}$	

Solution:
Tasks 2,3,4,5 depend on Task 1
Tasks 6,7 depends on Task 2

Tasks 8,9 depend on Task 3
Tasks 6,8 depend on Task 4
Task 7,9 depend on Task 5
Task 10 depends on Task 6
Task 2,5 depend on Task 7
Task 11 depend on Task 8
Task 13 depend on Task 9
Task 11,12 depend on Task 10
Task 13 depend on Task 11
Task 13 depend Task 12
Task 14 depend on Task 13
Task 14
So if we consider 3 processes p0, p1, p2 we have following graph:



Problem 3.6

Enumerate the critical paths in the decomposition of LU factorization shown in Figure 3.27.

Solution: is If we refer graph from problem 3.5 , critical path is the longest directed paths between start and end nodes. Possible critical paths can be as follows:
1) 1,2,6,10,11,13,14
2) 1,2,6,10,12,13,14
3) 1,4,6,10,12,13,14
4) 1,4,6,10,11,13,14

Problem 3.21

Consider seven tasks with running times of 1, 2, 3, 4, 5, 5, and 10 units, respectively. Assuming that it does not take any time to assign work to a process, compute the best- and worst-case speedup for a centralized scheme for dynamic mapping with two processes.

Solution:

Execution time by single processor: 1+2+3+4+5+5+10=30 units.

Best case speed up occurs when tasks are evenly distributed.
Process 1: Tasks with running times of 10 and 5 units (total 15 units).
Process 2: Tasks with running times of 5, 4, 3, 2, and 1 units (total 15 units).
Speed up= is execution time with a single processor/execution time with p processors

Best case speed up= 30/15= 2

Worst case speed up occurs if tasks are distributed to maximize make span.
Processor 1: Task with a running time of 10 units.
Processor 2: Tasks with running times of 5, 5, 4, 3, 2, 1 (20 units).

Worst case speed up= 30/20= 1.5