Step1: Convert Routing Sheet into a Machine Graph Matrix

Input: Routing Sheet

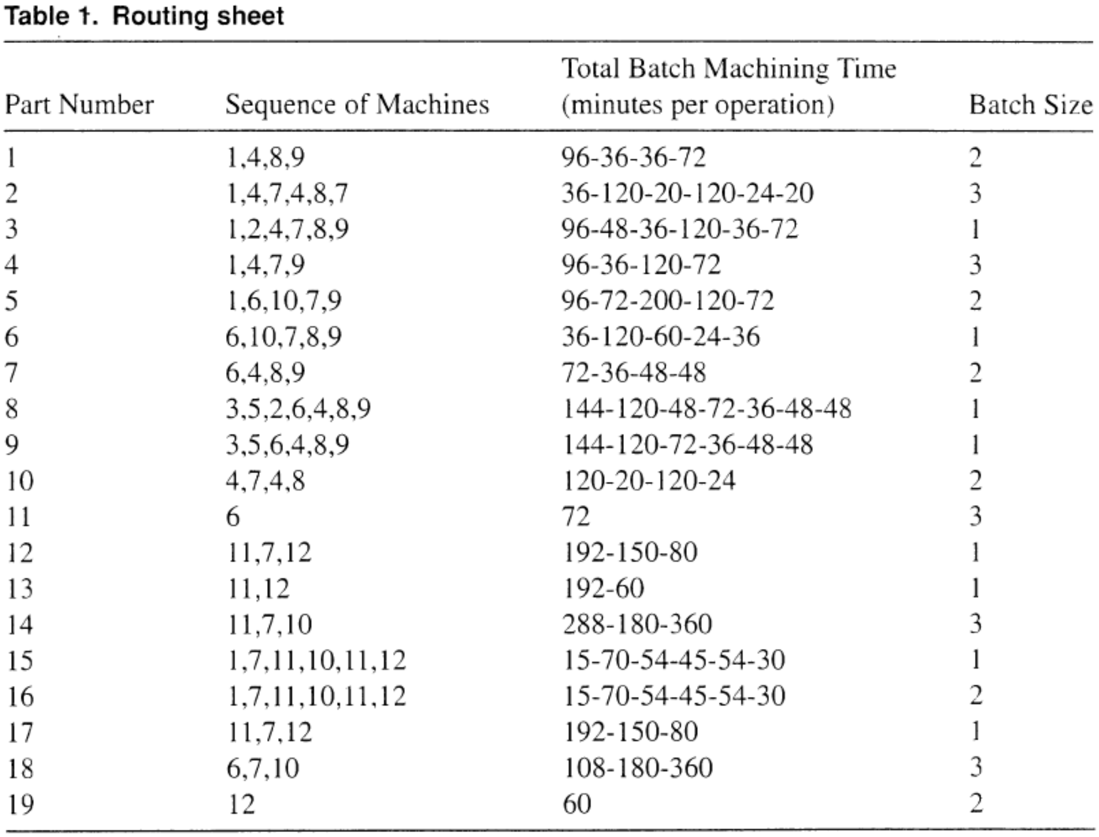
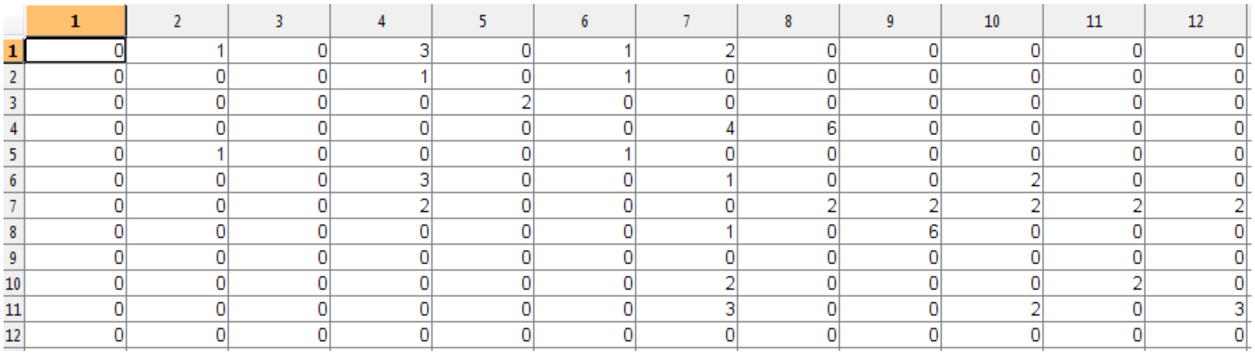


Table 1. Routing sheet

Part Number	Sequence of Machines	Total Batch Machining Time (minutes per operation)	Batch Size
1	1,4,8,9	96-36-36-72	2
2	1,4,7,4,8,7	36-120-20-120-24-20	3
3	1,2,4,7,8,9	96-48-36-120-36-72	1
4	1,4,7,9	96-36-120-72	3
5	1,6,10,7,9	96-72-200-120-72	2
6	6,10,7,8,9	36-120-60-24-36	1
7	6,4,8,9	72-36-48-48	2
8	3,5,2,6,4,8,9	144-120-48-72-36-48-48	1
9	3,5,6,4,8,9	144-120-72-36-48-48	1
10	4,7,4,8	120-20-120-24	2
11	6	72	3
12	11,7,12	192-150-80	1
13	11,12	192-60	1
14	11,7,10	288-180-360	3
15	1,7,11,10,11,12	15-70-54-45-54-30	1
16	1,7,11,10,11,12	15-70-54-45-54-30	2
17	11,7,12	192-150-80	1
18	6,7,10	108-180-360	3
19	12	60	2

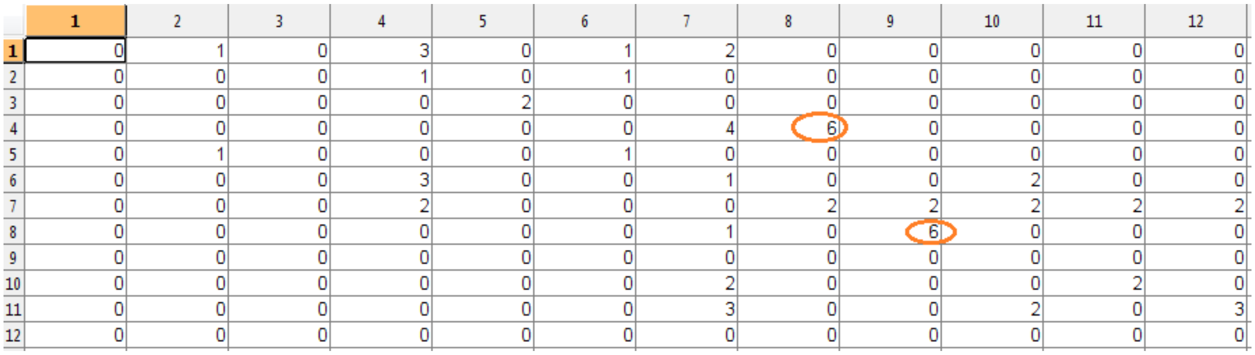
Output: Machine Graph Matrix



	1	2	3	4	5	6	7	8	9	10	11	12
1	0	1	0	3	0	1	2	0	0	0	0	0
2	0	0	0	1	0	1	0	0	0	0	0	0
3	0	0	0	0	2	0	0	0	0	0	0	0
4	0	0	0	0	0	0	4	6	0	0	0	0
5	0	1	0	0	0	1	0	0	0	0	0	0
6	0	0	0	3	0	0	1	0	0	2	0	0
7	0	0	0	2	0	0	0	2	2	2	2	2
8	0	0	0	0	0	0	1	0	6	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	2	0	0	0	2	0
11	0	0	0	0	0	0	3	0	0	2	0	3
12	0	0	0	0	0	0	0	0	0	0	0	0

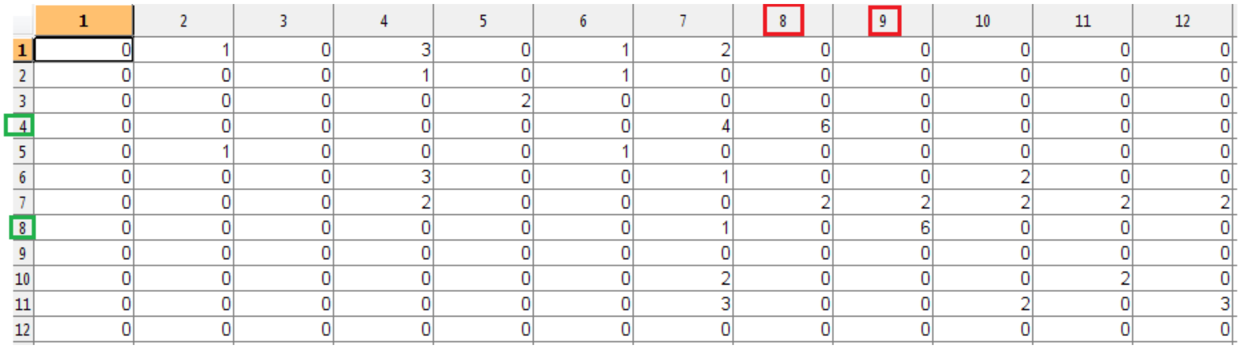
Step2: Iteratively start to form Cells using Machine Graph Matrix until all values in the matrix are 0

- Iteration Step 1 Start:
  - Get the most frequent routing pairs (4,8) & (8,9)



	1	2	3	4	5	6	7	8	9	10	11	12
1	0	1	0	3	0	1	2	0	0	0	0	0
2	0	0	0	1	0	1	0	0	0	0	0	0
3	0	0	0	0	2	0	0	0	0	0	0	0
4	0	0	0	0	0	0	4	6	0	0	0	0
5	0	1	0	0	0	1	0	0	0	0	0	0
6	0	0	0	3	0	0	1	0	0	2	0	0
7	0	0	0	2	0	0	0	2	2	2	2	2
8	0	0	0	0	0	0	1	0	6	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	2	0	0	0	2	0
11	0	0	0	0	0	0	3	0	0	2	0	3
12	0	0	0	0	0	0	0	0	0	0	0	0

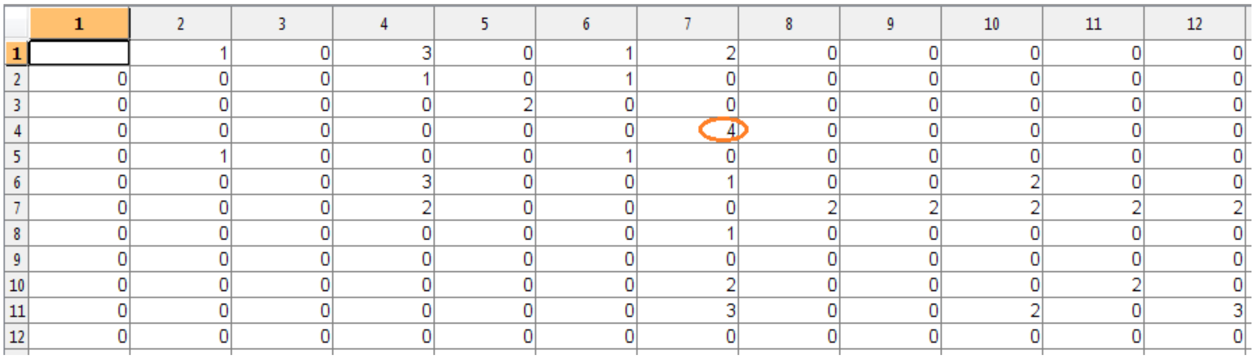
  - It is self-explanatory green rectangle is the starting machine and red rectangle is the ending machine.
  - For all the routing pairs, utilizing the ending machine, continue the routing flow by performing the search using them as starting machine until it leads to a dead-end (Row with all zeros).
  - Once a routing path is completed in a dead-end, mark it as a cell and assign the values in the matrix to 0



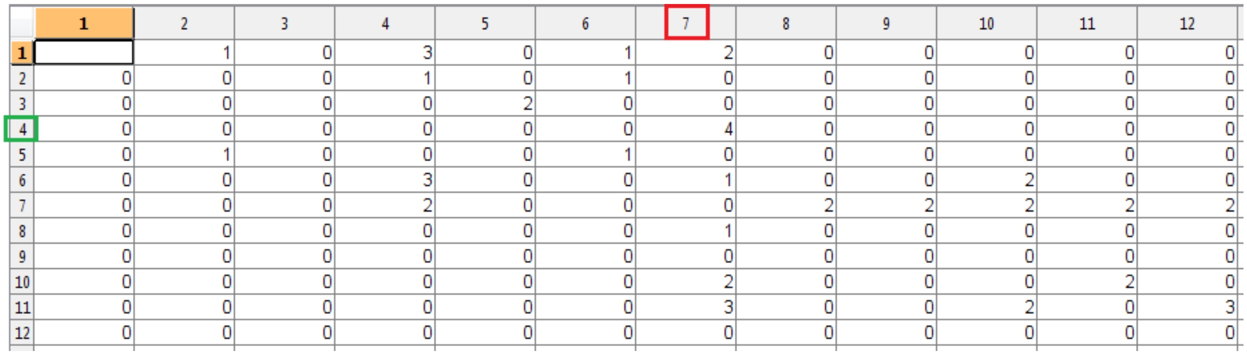
	1	2	3	4	5	6	7	8	9	10	11	12
1	0	1	0	3	0	1	2	0	0	0	0	0
2	0	0	0	1	0	1	0	0	0	0	0	0
3	0	0	0	0	2	0	0	0	0	0	0	0
4	0	0	0	0	0	0	4	6	0	0	0	0
5	0	1	0	0	0	1	0	0	0	0	0	0
6	0	0	0	3	0	0	1	0	0	2	0	0
7	0	0	0	2	0	0	0	2	2	2	2	2
8	0	0	0	0	0	0	1	0	6	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	2	0	0	0	2	0
11	0	0	0	0	0	0	3	0	0	2	0	3
12	0	0	0	0	0	0	0	0	0	0	0	0

- Iteration Step 1 Output:
  - **Cell1 : 4 -> 8 -> 9**
  - **Cell2 : 8 -> 9**

- Iteration Step 2 Start:
  - Get the most frequent routing pairs (4,7)



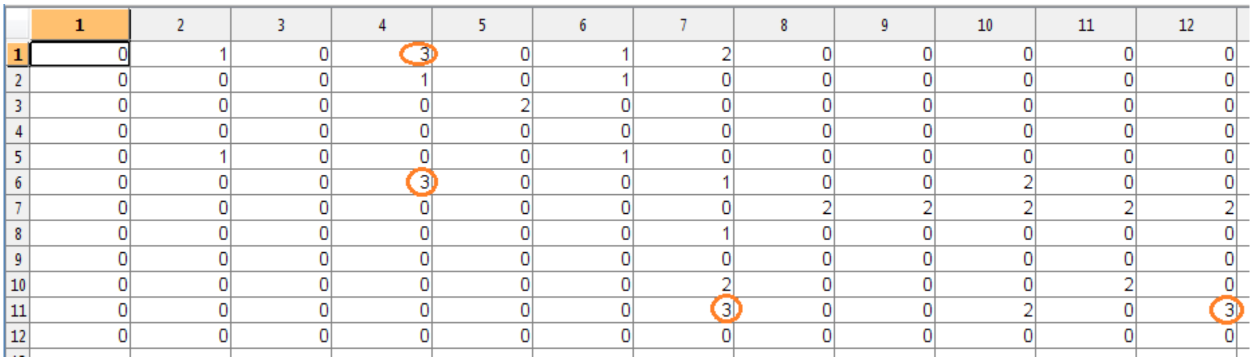
	1	2	3	4	5	6	7	8	9	10	11	12
1		1	0	3	0	1	2	0	0	0	0	0
2	0	0	0	1	0	1	0	0	0	0	0	0
3	0	0	0	0	2	0	0	0	0	0	0	0
4	0	0	0	0	0	0	4	0	0	0	0	0
5	0	1	0	0	0	1	0	0	0	0	0	0
6	0	0	0	3	0	0	1	0	0	2	0	0
7	0	0	0	2	0	0	0	2	2	2	2	2
8	0	0	0	0	0	0	1	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	2	0	0	0	2	0
11	0	0	0	0	0	0	3	0	0	2	0	3
12	0	0	0	0	0	0	0	0	0	0	0	0



	1	2	3	4	5	6	7	8	9	10	11	12
1		1	0	3	0	1	2	0	0	0	0	0
2	0	0	0	1	0	1	0	0	0	0	0	0
3	0	0	0	0	2	0	0	0	0	0	0	0
4	0	0	0	0	0	0	4	0	0	0	0	0
5	0	1	0	0	0	1	0	0	0	0	0	0
6	0	0	0	3	0	0	1	0	0	2	0	0
7	0	0	0	2	0	0	0	2	2	2	2	2
8	0	0	0	0	0	0	1	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	2	0	0	0	2	0
11	0	0	0	0	0	0	3	0	0	2	0	3
12	0	0	0	0	0	0	0	0	0	0	0	0

  - Loops are not allowed. So end the search when search hits dead-end or forms a loop.
- Iteration Step 2 Output:
  - **Cell3 : 4 -> 7 -> 4**

- Iteration Step 3 Start:
  - Get the most frequent routing pairs (1,4), (6,4), (11,7) & (11,12)

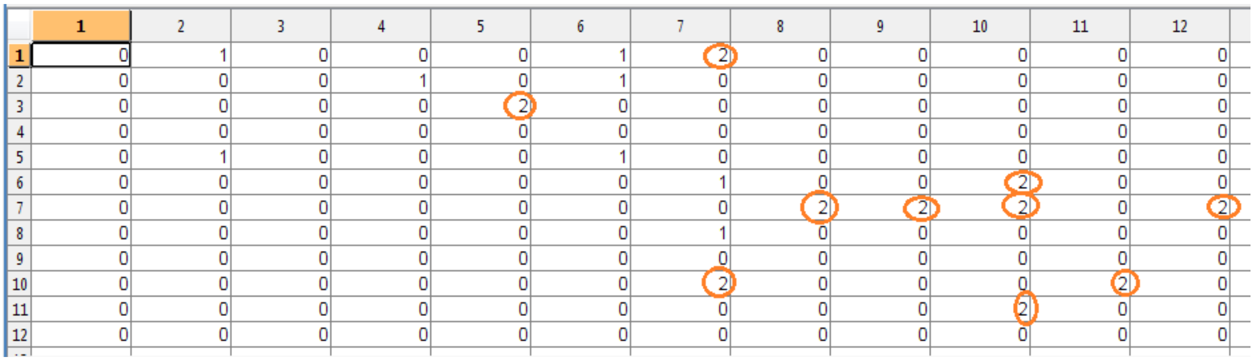


	1	2	3	4	5	6	7	8	9	10	11	12
1	0	1	0	3	0	1	2	0	0	0	0	0
2	0	0	0	1	0	1	0	0	0	0	0	0
3	0	0	0	0	2	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0
5	0	1	0	0	0	1	0	0	0	0	0	0
6	0	0	0	3	0	0	1	0	0	2	0	0
7	0	0	0	0	0	0	0	2	2	2	2	2
8	0	0	0	0	0	0	1	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	2	0	0	0	2	0
11	0	0	0	0	0	0	3	0	0	2	0	3
12	0	0	0	0	0	0	0	0	0	0	0	0

  - End the search when search hits dead-end or forms a loop.

- Iteration Step 3 Output
  - **Cell4 : 1 -> 4**
  - **Cell5 : 6-> 4**
  - **Cell6 : 11 -> 7 -> 11**
  - **Cell7 : 11 -> 12**

- Iteration Step 4 Start:
  - Get the most frequent routing pairs: (3, 5) , (1, 7), (10, 7), (7, 8), (7, 9) , (6,10) , (7, 10) , (11,10), (10, 11) & (7, 12)

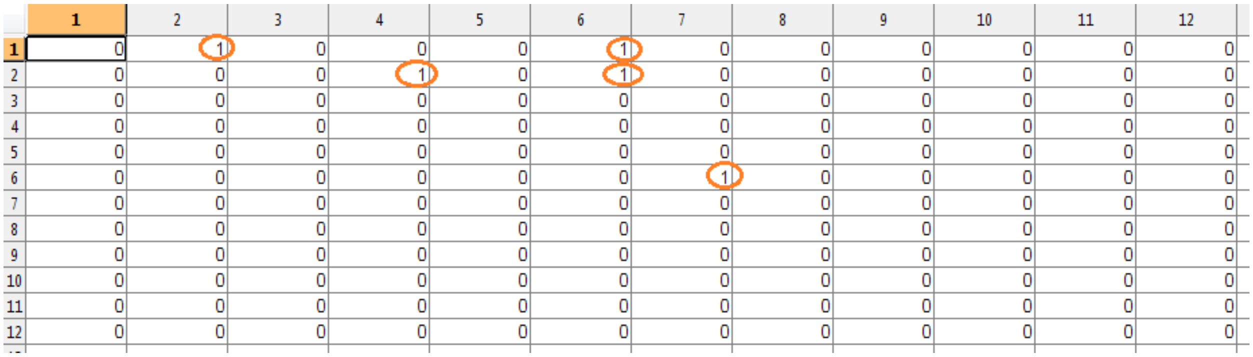


	1	2	3	4	5	6	7	8	9	10	11	12
1	0	1	0	0	0	1	2	0	0	0	0	0
2	0	0	0	1	0	1	0	0	0	0	0	0
3	0	0	0	0	2	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0
5	0	1	0	0	0	1	0	0	0	0	0	0
6	0	0	0	0	0	0	1	0	0	2	0	0
7	0	0	0	0	0	0	0	2	2	2	0	2
8	0	0	0	0	0	0	1	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	2	0	0	0	2	0
11	0	0	0	0	0	0	0	0	0	2	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0

- End the search when search hits dead-end or forms a loop or has a fork (2 or more options).

- Iteration Step 4 Output
  - **Cell8 : 3 -> 5->2, 6**
  - **Cell9 : 1-> 7, 8, 9, 10, 12**
  - **Cell10 : 10 -> 7**
  - **Cell11 : 7-> 8 ->7**
  - **Cell12 : 7-> 9**
  - **Cell13 : 6-> 10 ->11 ->10**
  - **Cell14 : 7-> 10**
  - **Cell15 : 11 -> 10**
  - **Cell16 : 10 -> 11**
  - **Cell17 : 7-> 12**

- Iteration Step 5 Start:
  - Get the most frequent routing pairs: (1, 2), (2, 4), (1, 6), (2, 6) & (6, 7)



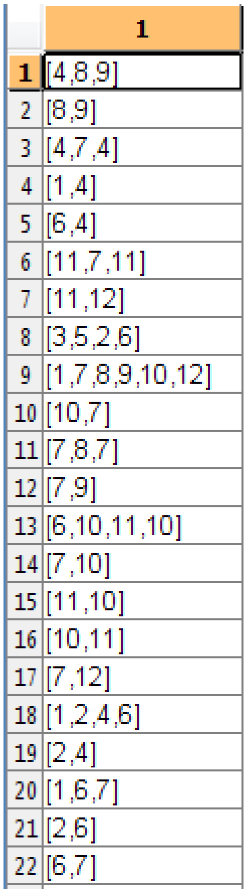
	1	2	3	4	5	6	7	8	9	10	11	12
1	0	1	0	0	0	1	0	0	0	0	0	0
2	0	0	0	1	0	1	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	1	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0

  - End the search when search hits dead-end or forms a loop or has a fork (2 or more options).

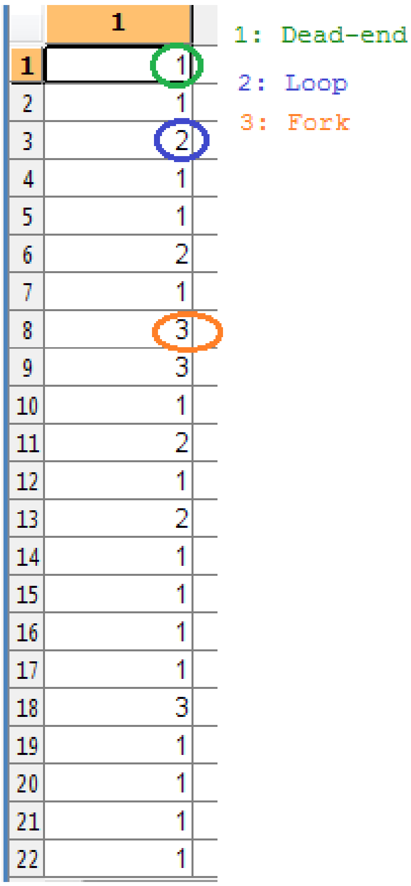
- Iteration Step 5 Output:
  - **Cell18 : 1 -> 2, 4, 6**
  - **Cell19 : 2 -> 4**
  - **Cell20 : 1 -> 6 ->7**

- **Cell21 : 2 -> 6**
- **Cell22 : 6 -> 7**

Summary of the Cells with reason code for the end of search path



	1
1	[4,8,9]
2	[8,9]
3	[4,7,4]
4	[1,4]
5	[6,4]
6	[11,7,11]
7	[11,12]
8	[3,5,2,6]
9	[1,7,8,9,10,12]
10	[10,7]
11	[7,8,7]
12	[7,9]
13	[6,10,11,10]
14	[7,10]
15	[11,10]
16	[10,11]
17	[7,12]
18	[1,2,4,6]
19	[2,4]
20	[1,6,7]
21	[2,6]
22	[6,7]



	1
1	1
2	1
3	2
4	1
5	1
6	2
7	1
8	3
9	3
10	1
11	2
12	1
13	2
14	1
15	1
16	1
17	1
18	3
19	1
20	1
21	1
22	1

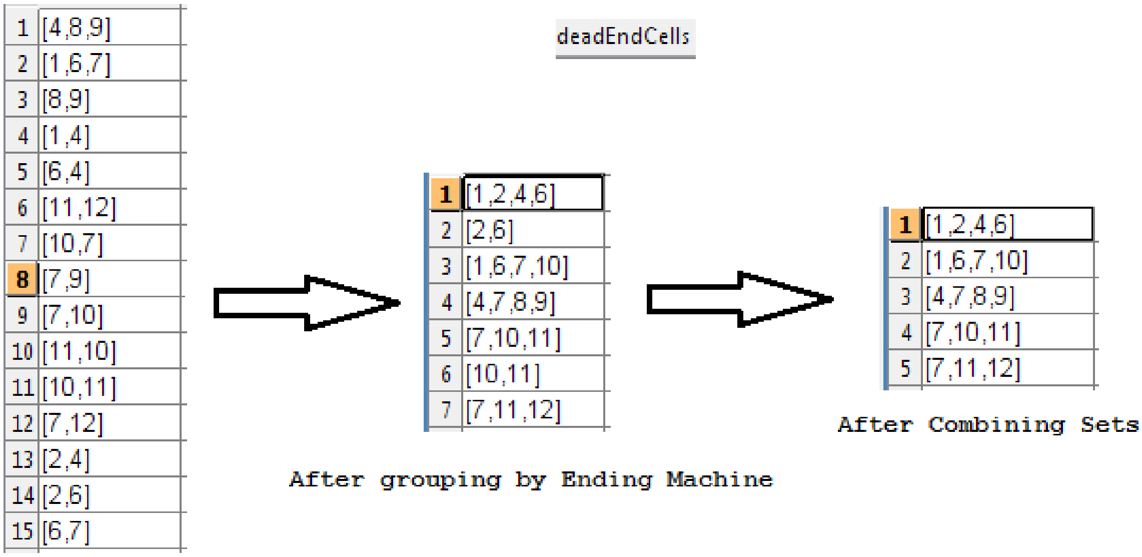
1: Dead-end

2: Loop

3: Fork

Processing the cells

- For cells which ended with dead-end have natural ending. So, they will have first priority for forming machine cells.
- Group the cells which have same ending machine
- Then combine any Sub-set Cells



1	[4,8,9]
2	[1,6,7]
3	[8,9]
4	[1,4]
5	[6,4]
6	[11,12]
7	[10,7]
8	[7,9]
9	[7,10]
10	[11,10]
11	[10,11]
12	[7,12]
13	[2,4]
14	[2,6]
15	[6,7]

Original Set

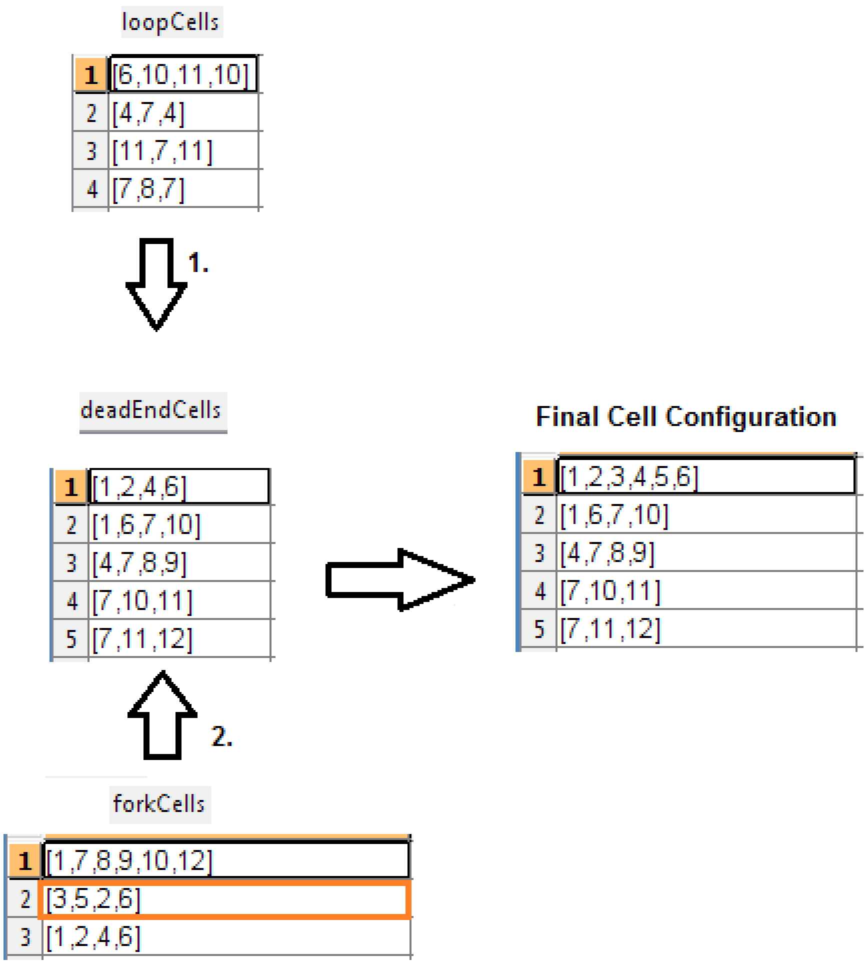
1	[1,2,4,6]
2	[2,6]
3	[1,6,7,10]
4	[4,7,8,9]
5	[7,10,11]
6	[10,11]
7	[7,11,12]

After grouping by Ending Machine

1	[1,2,4,6]
2	[1,6,7,10]
3	[4,7,8,9]
4	[7,10,11]
5	[7,11,12]

After Combining Sets

- Dead-end cell formation output:
  - Cell1: 1, 2, 4,6
  - Cell2: 1,6,7,10
  - Cell3: 4, 7, 8, 9
  - Cell4: 7, 10, 11, 12
- Are there still machines not part of the above Cells?
  - Yes:
    - Use cells derived from loops (1), followed by forks (2) to cover the unused machines.
    - Based on number of common machines, either merge the cells into existing dead-end cells or create a new cell.
  - No:
    - End the cell formation process



loopCells

1	[6,10,11,10]
2	[4,7,4]
3	[11,7,11]
4	[7,8,7]

deadEndCells

1	[1,2,4,6]
2	[1,6,7,10]
3	[4,7,8,9]
4	[7,10,11]
5	[7,11,12]

forkCells

1	[1,7,8,9,10,12]
2	[3,5,2,6]
3	[1,2,4,6]

Final Cell Configuration

1	[1,2,3,4,5,6]
2	[1,6,7,10]
3	[4,7,8,9]
4	[7,10,11]
5	[7,11,12]

Final Output

- **Cell1: 1, 2, 3, 4, 5, 6**
- **Cell 2: 1, 6, 7, 10**
- **Cell 3: 4, 7, 8, 9**
- **Cell 4: 7, 10, 11**
- **Cell 5: 7, 11, 12**