Heuristic Scheduling: List Scheduling for MLRC

Dr. Chandan Karfa

Department of Computer Science and Engineering

भारतीय प्रौद्योगिकी संस्थान गुवाहाटी
INDIAN INSTITUTE OF TECHNOLOGY GUWAHATI

Multiprocessor Scheduling and Hu's algorithm

- Assumptions:
  - A1: All operations have the same type – a multiprocessor is executing the operations
  - A2: All operations have unit delay
  - A3: Sequence graph is Tree
    - Implications: There is no parallel paths.
    - have single paths from each vertex to the sink with monotonically unit-wise decreasing labels (path length from the sink node)
- The problem is in P with A1, A2 and A3
  - Greedy strategy
  - Exact solution

Hu's Algorithm (MLRC)

HU (*G(V,E), a*) {

Label the vertices // *a* = resource constraint (scalar)
// *label* = length of longest path passing through the vertex

$l = 1$

repeat {

U = unscheduled vertices in V whose predecessors have been already scheduled (or have no predecessors)

Select $S \subseteq U$ such that $|S| \leq a$ and labels in *S* are *maximal*

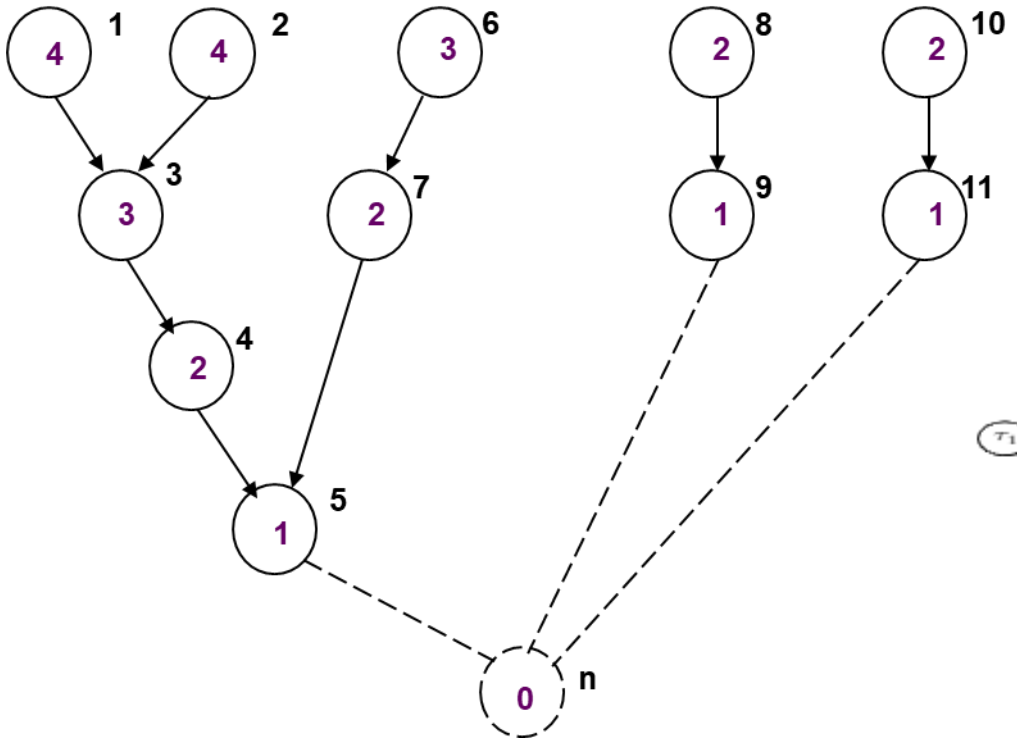
Schedule the *S* operations at step l by setting

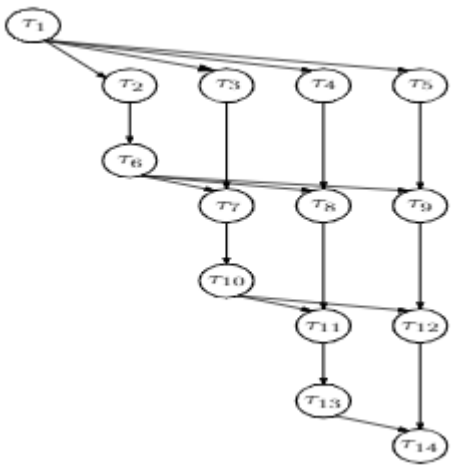
$t_i = l, \quad \forall v_i \in S;$

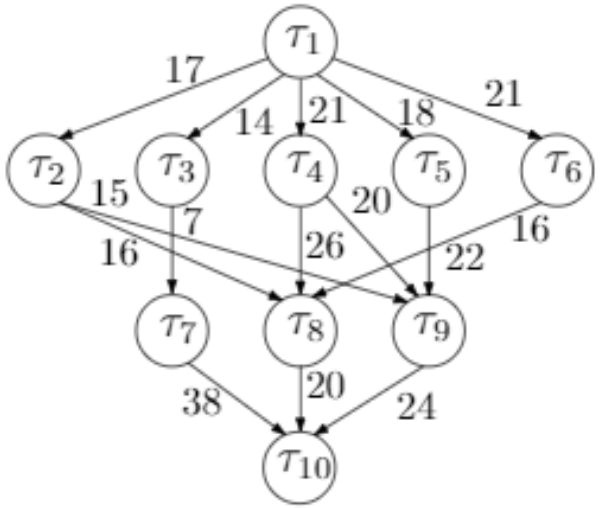
$l = l + 1;$

} until v_n is scheduled.

}


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


τ_1
τ_2
τ_3
τ_4
τ_5
τ_6
τ_7
τ_8
τ_9
τ_{10}
τ_{11}
τ_{12}
τ_{13}
τ_{14}


τ_1
17
14
21
18
21
τ_2
τ_3
τ_4
τ_5
τ_6
15
7
20
16
16
26
22
τ_7
τ_8
τ_9
38
20
24
τ_{10}

List Scheduling

- Extend the idea of HU's algorithm to several operators
- Greedy algorithm for ML-RC and MR-LC
  - Does NOT guarantee optimum solution
- Similar to Hu's algorithm
  - Operation selection decided by criticality
  - *O(n)* time complexity
- Considers a more general case
  - Resource constraints with different resource types
  - Multi-cycle operations
  - Pipelined operations
- Priority list heuristics
  - Longest path to sink

List Scheduling Algorithms

- Algorithm 1: Minimize latency under resource constraint (ML-RC)
  - Resource constraint represented by vector ***a*** (indexed by resource type)
    - Example: two types of resources, MULT (a_1=1), ADD (a_2=2)
- Algorithm 2: Minimize resources under latency constraint (MR-LC)
  - Latency constraint is given and resource constraint vector ***a*** to be minimized

Define:

- The candidate operations $U_{l,k}$
  - those operations of type k whose predecessors have already been scheduled early enough (completed at step l):

    $U_{l,k} = \{ v_i \subseteq V: type(v_i) = k$ and $t_j + d_j \leq l$, for all j: $(v_j, v_i) \subseteq E$
- The unfinished operations $T_{l,k}$
  - those operations of type k that started at earlier cycles but whose execution has not finished at step l *(multi-cycle opearations):*

    $T_{l,k} = \{ v_i \subseteq V: type(v_i) = k$ and $t_i + d_i > l \}$
- Priority list
  - List operators according to some heuristic urgency measure
  - Common priority list: labeled by position on the longest path in decreasing order

The candidate operations $U_{l,k}$

- The candidate operations U_{lk} are those operations of type k whose predecessors have already been scheduled early enough so that the corresponding operations are completed at step l.

$$U_{l,k} = \{v_i \in V : T(v_i) = k \text{ and } t_j + d_j \leq l \quad \forall j : (v_j, v_i) \in E\} \text{ for any resource type } k = 1, 2, \ldots, n_{res}.$$

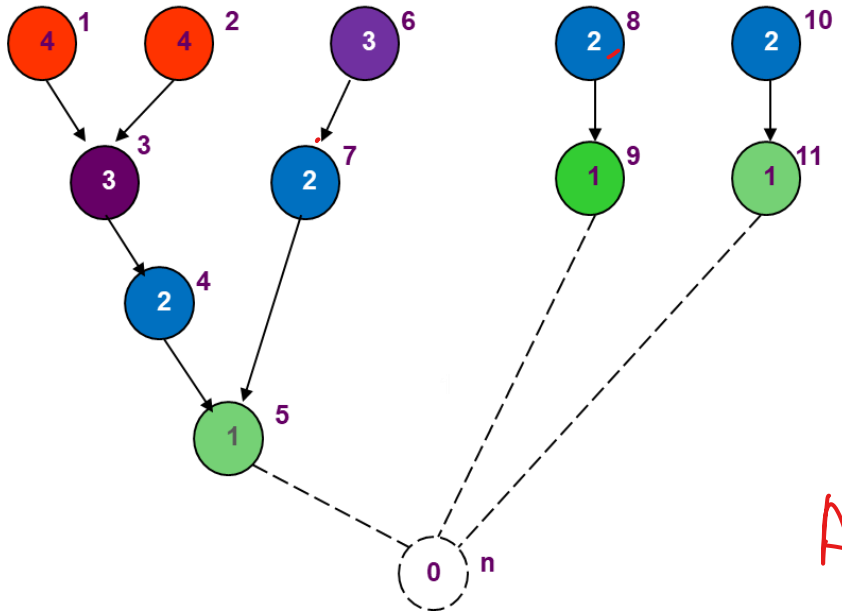
The unfinished operations $T_{l,k}$

- The unfinished operations T_{lk} are those operations of type *k* that started at earlier cycles and whose execution is not finished at step l.

$$T_{l,k} = \{v_i \in V : T(v_i) = k \text{ and } t_i + d_i > l\}$$

Priority List

- A *priority list* of the operations is used in choosing among the operations, based on some heuristic urgency measure.
- A common priority list is to label the vertices with weights of their longest path to the sink and to rank them in decreasing order. The most urgent operations are scheduled first.



List Scheduling Algorithm 1: ML-RC

Minimize latency under resource constraint

LIST_L (G(V,E), **a**) { // resource constraints specified by vector **a**

l = 1

repeat {

for each resource type k {

$U_{l,k}$ = candidate operations available in step l

$T_{l,k}$ = unfinished operations (in progress)

Select $S_k \subseteq U_{l,k}$ such that $|S_k| + |T_{l,k}| \leq a_k$

Schedule the S_k operations at step l

}

$l = l + 1$

} until v_n is scheduled

}

Note: If for all operators *i*, $d_i = 1$ (unit delay), the set $T_{l,k}$ is empty

Hu's Algorithm (MLRC)

HU (*G(V,E), a*) {

Label the vertices // *a* = resource constraint (scalar)

// *label* = length of longest path passing through the vertex

$l = 1$

repeat {

U = unscheduled vertices in V whose predecessors have been already scheduled (or have no predecessors)

Select $S \subseteq U$ such that $|S| \leq a$ and labels in *S* are *maximal*

Schedule the *S* operations at step l by setting

$t_i = l, \quad \forall v_i \in S;$

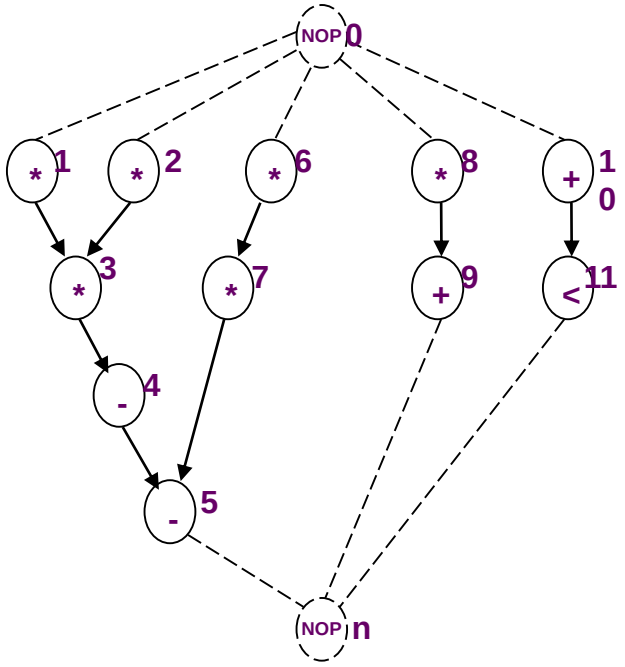
$l = l + 1;$

} until v_n is scheduled.

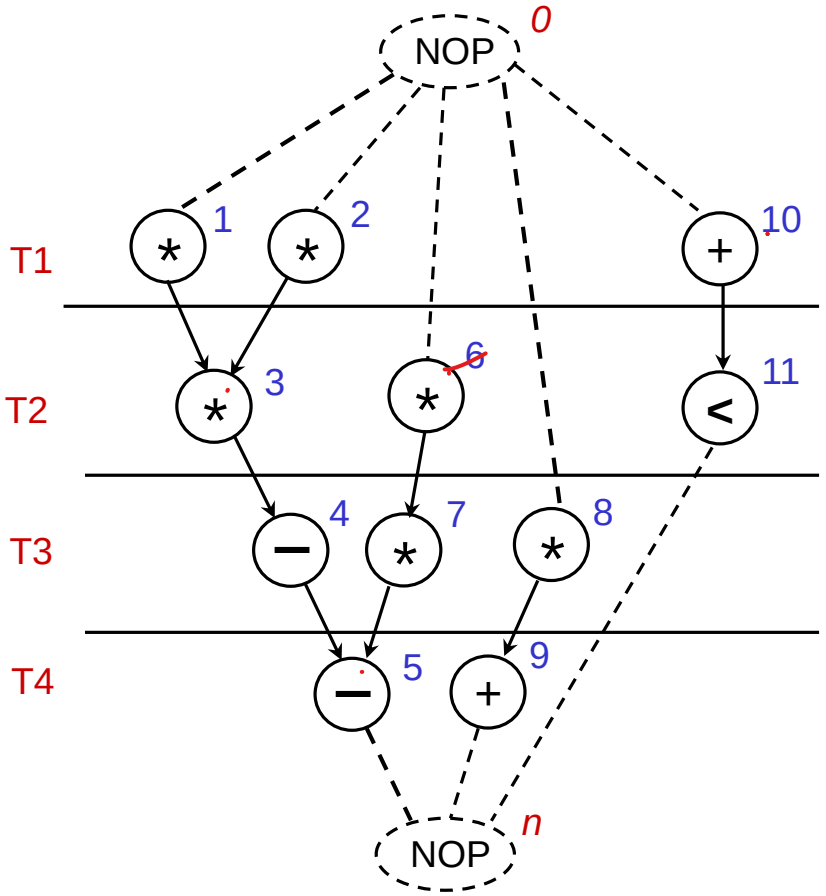
}

List Scheduling ML-RC – Example 1 (a=[2,2])

Minimize latency under resource constraint (with ***d* = 1**)

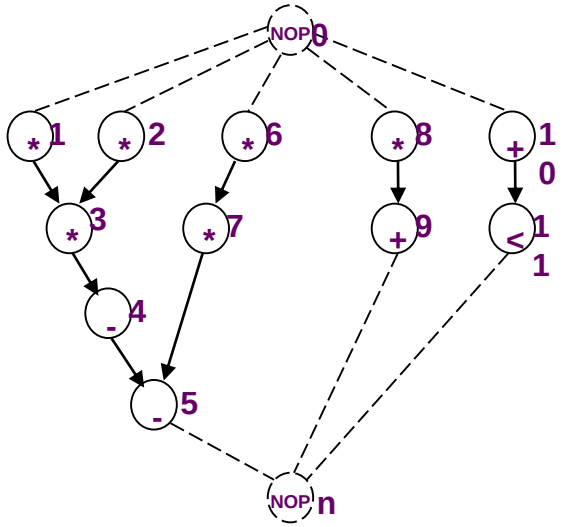


- Assumptions
  - All operations have unit delay (d_i=1)
  - Resource constraints:
    MULT: a_1 = 2, ALU: a_2 = 2
- Step 1:
  - $U_{1,1} = \{v_1, v_2, v_6, v_8\}$, select $\{v_1, v_2\}$
  - $U_{1,2} = \{v_{10}\}$, select + schedule
- Step 2:
  - $U_{2,1} = \{v_3, v_6, v_8\}$, select $\{v_3, v_6\}$
  - $U_{2,2} = \{v_{11}\}$, select + schedule
- Step 3:
  - $U_{3,1} = \{v_7, v_8\}$, select + schedule
  - $U_{3,2} = \{v_4\}$, select + schedule
- Step 4:
  - $U_{4,2} = \{v_5, v_9\}$, select + schedule



List Scheduling ML-RC – Example 2a (a = [3,1])

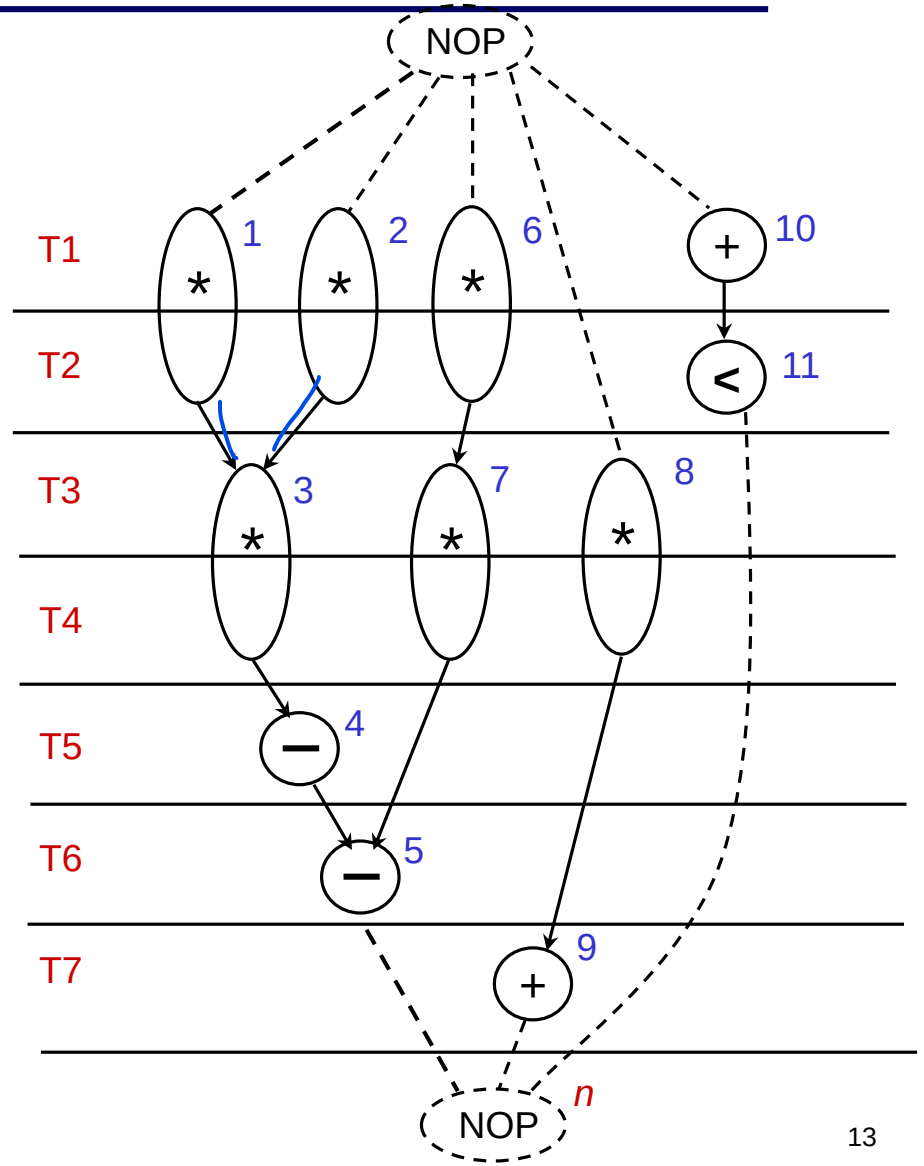
Minimize latency under resource constraint (with $d_1=2$, $d_2=1$)



- Assumptions
  - Operations have different delay:
    $del_{MULT} = 2$, $del_{ALU} = 1$
  - Resource constraints:
    MULT: $a_1 = 3$, ALU: $a_2 = 1$

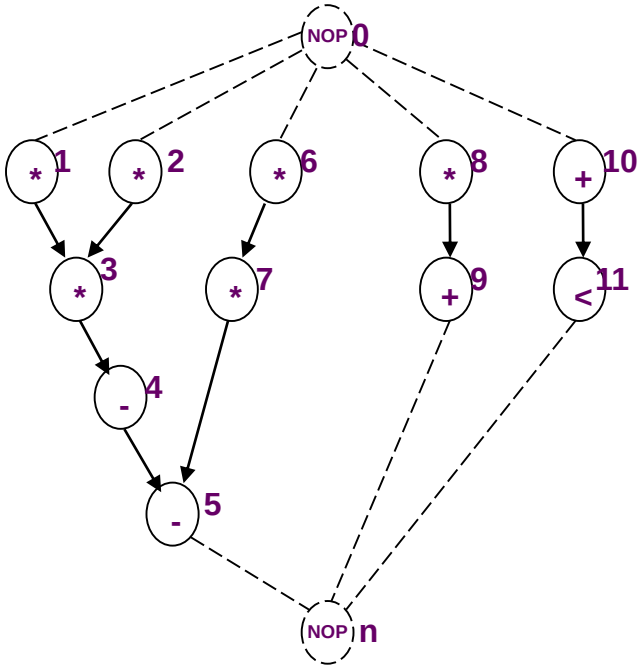
MUTL	ALU	start time
U= $\{v_1, v_2, v_6\}$	v_{10}	1
T= $\{v_1, v_2, v_6\}$	v_{11}	2
U= $\{v_3, v_7, v_8\}$	--	3
T= $\{v_3, v_7, v_8\}$	--	4
--	v_4	5
--	v_5	6
--	v_9	7

Latency L = 8



List Scheduling ML-RC – Example 2b (Pipelined)

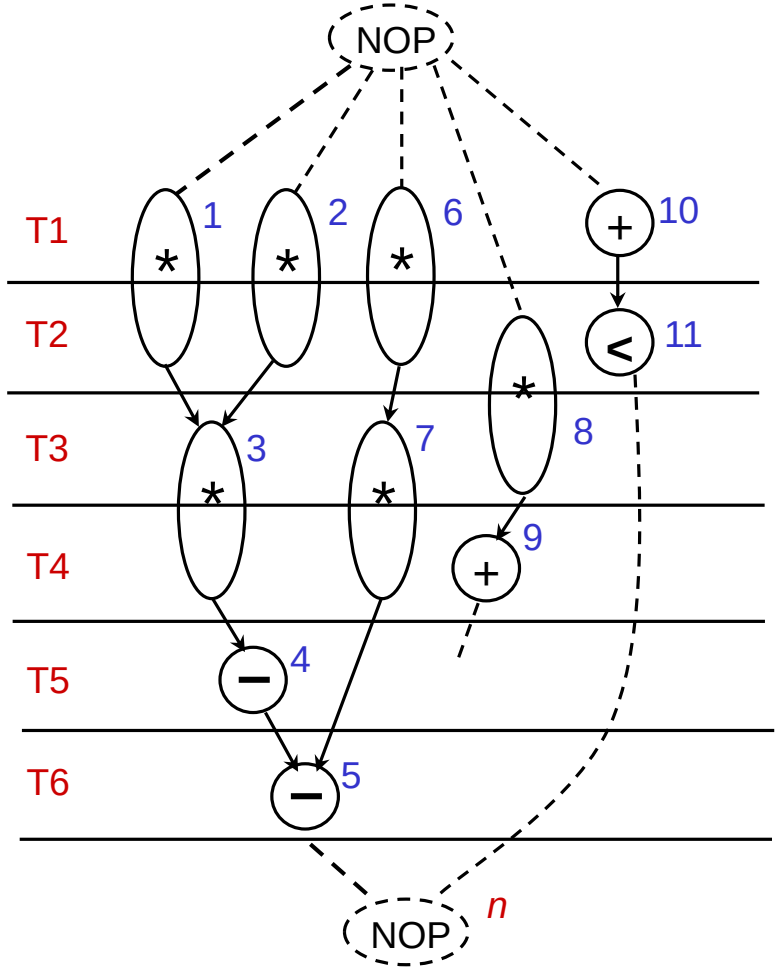
Minimize latency under resource constraint (a_1=3 *MULTs*, a_2=3 *ALUs*)



- Assumptions
  - Multipliers are pipelined
  - Sharing between first and second pipeline stage allowed for different multipliers

MULT	ALU	start time
$\{v_1, v_2, v_6\}$	v_{10}	1
v_8	v_{11}	2
$\{v_3, v_7\}$	--	3
--	v_9	4
--	v_4	5
--	v_5	6

- *L=7*, Compare to multi-cycle case



Thank You