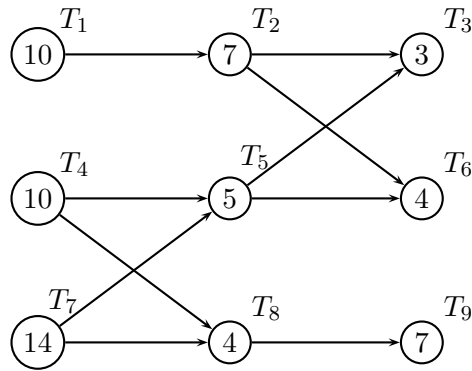


The List Processing Algorithm

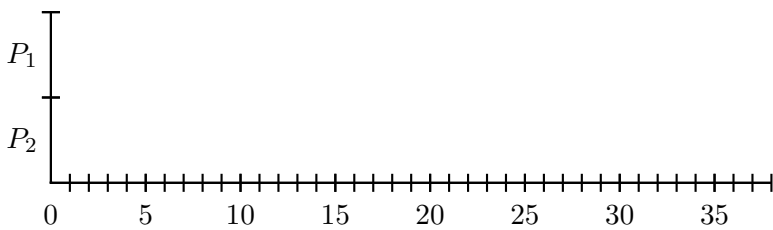
Goal: Schedule tasks in a preference list (called a priority list in the book) while obeying the order requirement directed graph. Does not guarantee minimal completion time. Does not guarantee preference list order.

1. draw schedule axes and scheduling table
2. start at time 0
3. record current time
4. identify tasks finishing at current time
5. list ready tasks (remaining from previous time and from newly finished tasks)
6. for each idle processor (top down; P_1, P_2, \dots)
 - a. select ready task that appears earliest in the preference list
 - b. add task to schedule graph
7. advance current time to next time that a task finishes on any processor
8. repeat from step 3

Example: Schedule the following tasks on two processors using the List Processing Algorithm with the preference list: $T_1, T_2, T_3, T_4, T_5, T_6, T_7, T_8, T_9$.

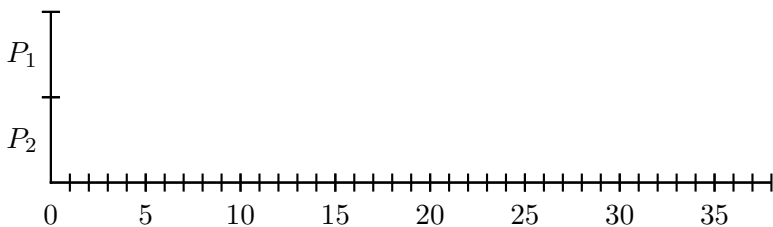


We start by drawing an empty schedule and scheduling table:



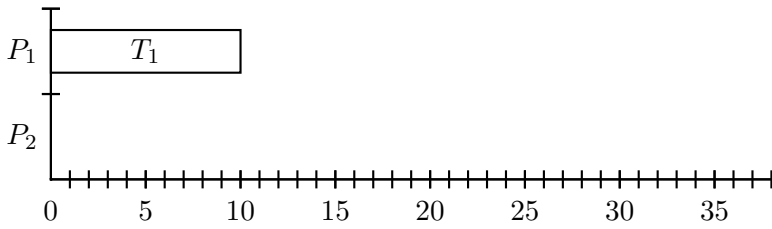
time	finish	ready

At time 0, no tasks are finishing and there are three ready tasks.



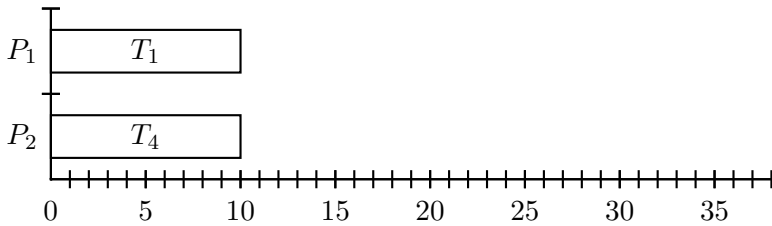
time	finish	ready
0		T_1, T_4, T_7

Of the three ready tasks, T_1 appears first in our preference list ($T_1, T_2, T_3, T_4, T_5, T_6, T_7, T_8, T_9$). Schedule T_1 on the first idle processor.



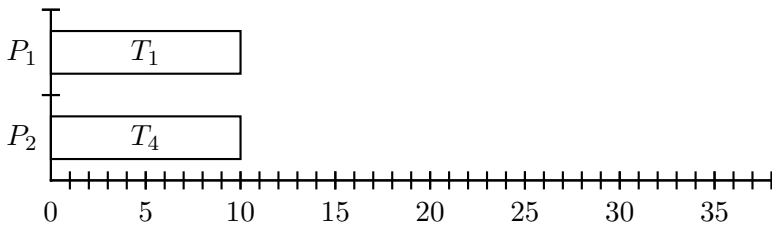
time	finish	ready
0		T_1, T_4, T_7

Our other processor is also idle and we have more ready tasks. T_4 is the next ready task in our preference list. Schedule T_4 on the next idle processor.



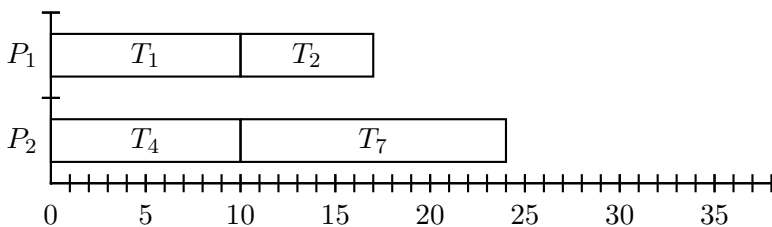
time	finish	ready
0		T_1, T_4, T_7

We have filled all of our idle processors, advance to the next time that a task finishes (time 10). Identify finished tasks (T_1 and T_4), copy down the old ready task (T_7). T_1 points to T_2 which becomes ready. T_4 points to T_5 and T_8 , however both of those depend on T_7 finishing and T_7 is not listed in our finish column, thus they are not yet ready.



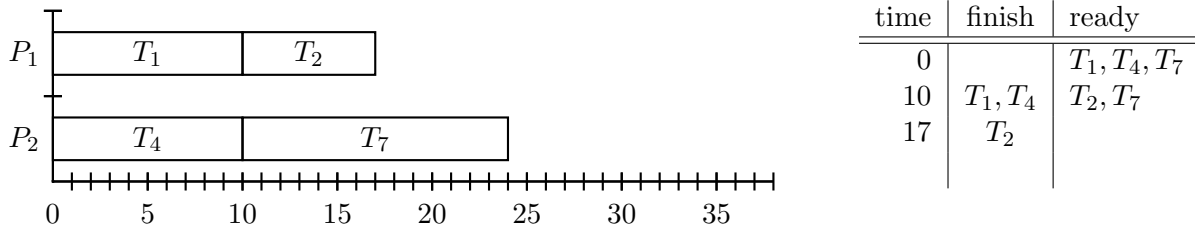
time	finish	ready
0		T_1, T_4, T_7
10	T_1, T_4	T_7, T_2

Schedule our ready tasks on the available processors, being careful to schedule T_2 first onto P_1 , then T_7 onto P_2 .

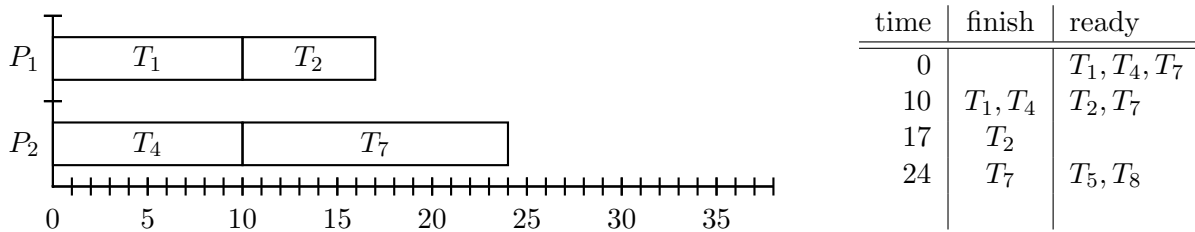


time	finish	ready
0		T_1, T_4, T_7
10	T_1, T_4	T_7, T_2

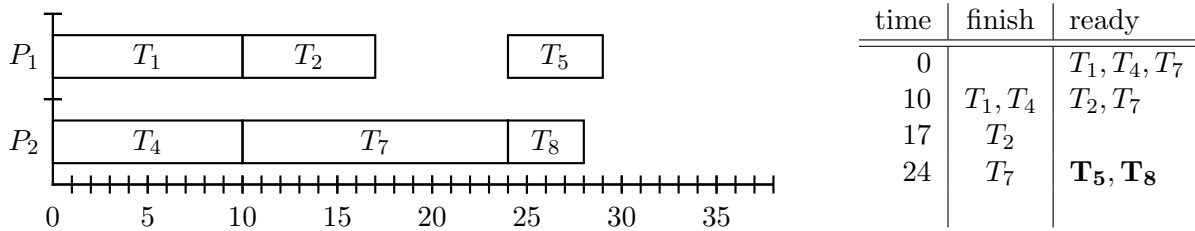
All processors are now working at time 10, advance time to time 17 where T_2 finishes. Both tasks at time 10 were scheduled, so we do not have any tasks to copy down. T_2 points to T_3 and T_6 , however both T_3 and T_6 depend on T_5 and T_5 is not yet finished. Therefore we have no ready tasks at this time.



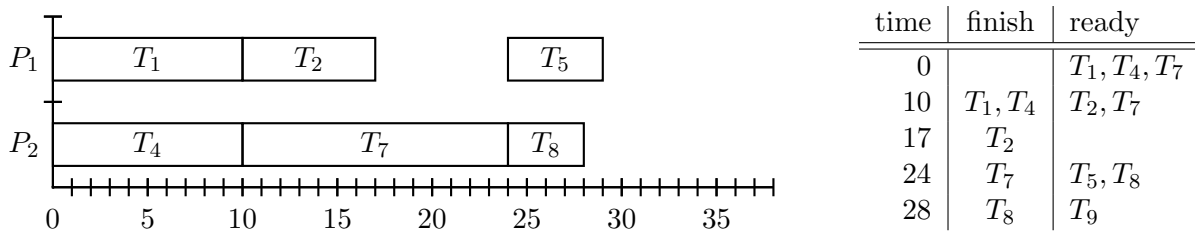
Since we have no ready tasks we must proceed without assigning any task to the next time that a task finishes. Task T_7 finishes at time 24. T_7 points to T_5 and T_8 . These both depend on T_2 , but T_2 is already in our finished column! We add T_5 and T_8 to our list of ready tasks.



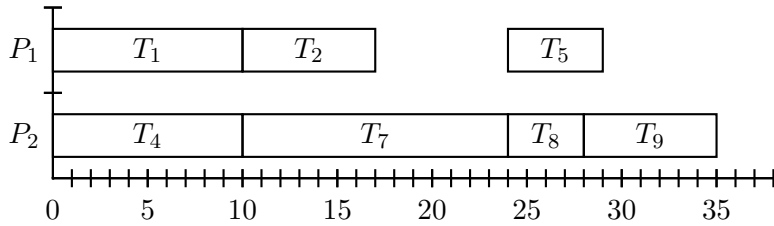
At this time both of our processors are idle so we can schedule both of these tasks. Notice that we must start T_5 exactly at time 24. We can not go back in time by scheduling it earlier! Again, T_5 gets scheduled first because it is earlier in our preference list.



Task T_8 finishes at time 28 making T_9 ready.

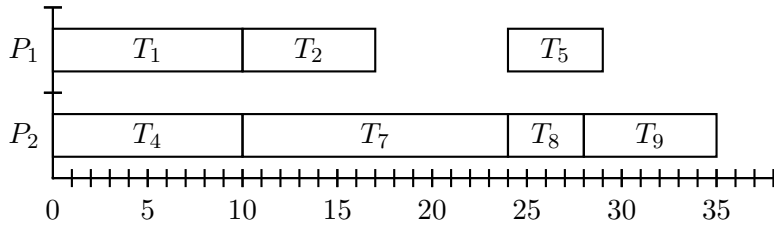


Schedule T_9 .



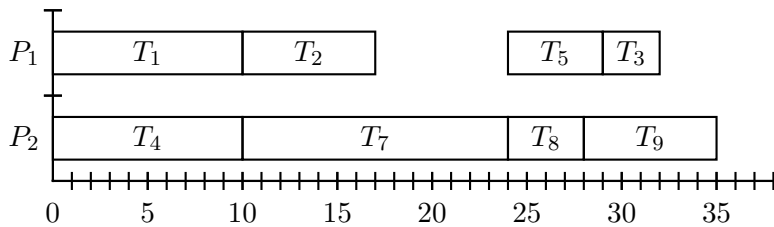
time	finish	ready
0		T_1, T_4, T_7
10	T_1, T_4	T_2, T_7
17	T_2	
24	T_7	T_5, T_8
28	T_8	T_9

Task T_5 finishes at time 29 making T_3 and T_6 ready.



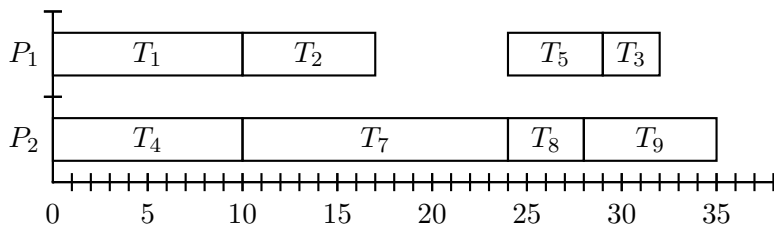
time	finish	ready
0		T_1, T_4, T_7
10	T_1, T_4	T_2, T_7
17	T_2	
24	T_7	T_5, T_8
28	T_8	T_9
29	T_5	T_3, T_6

Schedule T_3 (appears earlier in preference list!).



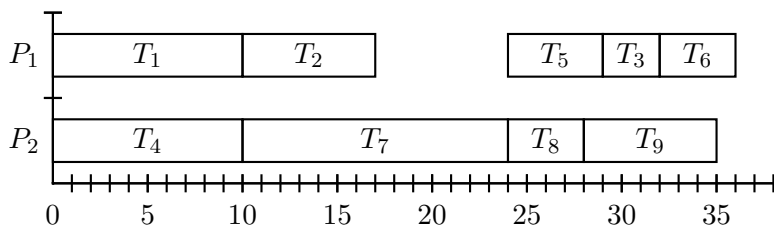
time	finish	ready
0		T_1, T_4, T_7
10	T_1, T_4	T_2, T_7
17	T_2	
24	T_7	T_5, T_8
28	T_8	T_9
29	T_5	T_3, T_6

Task T_3 finishes at time 32. Carry T_6 down.



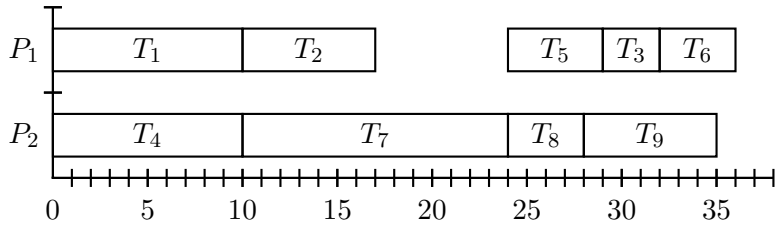
time	finish	ready
0		T_1, T_4, T_7
10	T_1, T_4	T_2, T_7
17	T_2	
24	T_7	T_5, T_8
28	T_8	T_9
29	T_5	T_3, T_6
32	T_3	T_6

Schedule T_6 .



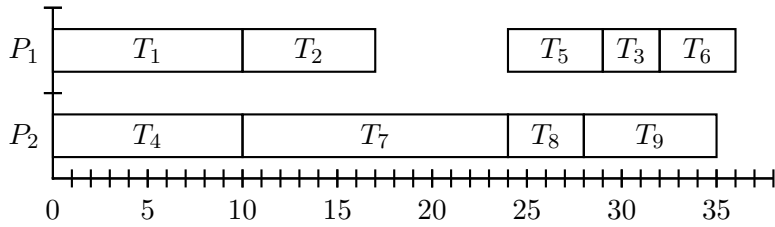
time	finish	ready
0		T_1, T_4, T_7
10	T_1, T_4	T_2, T_7
17	T_2	
24	T_7	T_5, T_8
28	T_8	T_9
29	T_5	T_3, T_6
32	T_3	T_6

At time 35 T_9 finishes. No further ready tasks.



time	finish	ready
0		T_1, T_4, T_7
10	T_1, T_4	T_2, T_7
17	T_2	
24	T_7	T_5, T_8
28	T_8	T_9
29	T_5	T_3, T_6
32	T_3	T_6
35	T_9	

At time 35 T_6 finishes. No further ready tasks and all processors idle. We are finished at time 36.



time	finish	ready
0		T_1, T_4, T_7
10	T_1, T_4	T_2, T_7
17	T_2	
24	T_7	T_5, T_8
28	T_8	T_9
29	T_5	T_3, T_6
32	T_3	T_6
35	T_9	
36	T_6	done