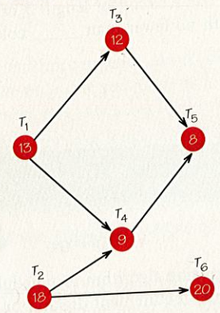
(c) Does either list lead to a completion time that equals the length of the critical path?
(d) Show that no list can ever lead to a completion time equal to the length of the critical path (providing the schedule uses two processors).

7. (a) Use the following order-requirement digraph to schedule the 6 tasks $T_1, T_2, T_3, T_4, T_5, T_6$ on two processors with the priority lists:
(i) $T_1, T_2, T_3, T_4, T_5, T_6$
(ii) $T_1, T_6, T_3, T_5, T_4, T_2$



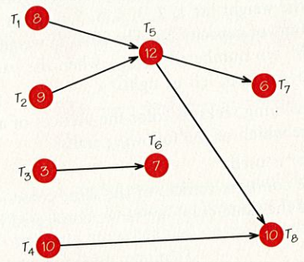
(b) Are either of the schedules produced from these lists optimal? If not, can you find a priority list that will result in an optimal schedule?
(c) Find the critical path and its length. Explain why no schedule has earliest completion time equal to the length of the critical path.

8. (a) Repeat Exercise 7, but interchange the task times of tasks T_2 and T_6.
(b) How does the completion time for an optimum schedule for this situation compare with the optimum schedule for Exercise 7?

9. (a) If one adds a new directed edge to an order-requirement digraph D, can the critical path in the the new order-requirement digraph D' have longer length?
(b) If one adds a new directed edge to an order-requirement digraph D, can the critical path in the the new order-requirement digraph D' have shorter length?

◆ **10.** Discuss scheduling problems for which it is not reasonable to assume that once a processor starts a task, it will always complete that task, before it works on any other task. Give examples for which this approach would be reasonable.

11. For the accompanying order-requirement digraph, apply the list-processing algorithm, using three processors for lists (a) through (c). How do the completion times obtained compare with the length of the critical path?

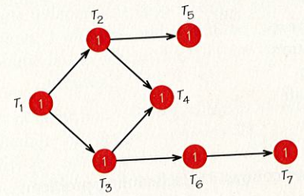


(a) $T_1, T_2, T_3, T_4, T_5, T_6, T_7, T_8$
(b) $T_1, T_3, T_5, T_7, T_2, T_4, T_6, T_8$
(c) $T_8, T_6, T_4, T_2, T_1, T_3, T_5, T_7$

12. (a) Can you find an order-requirement digraph with four tasks for which every priority list used to schedule the tasks on two machines assigns task T_4 to machine 1 at time 0?
(b) Can you choose the order-requirement digraph in part (a) so that machine 2 stays idle for all lists from time 0 to time 3?

13. Can you give examples of scheduling problems for which it seems reasonable to assume that all the task times are the same?

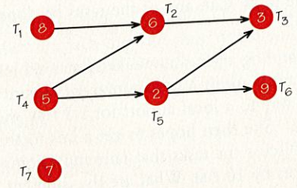
14. Use the list-processing algorithm to schedule the tasks in the following order-requirement digraph on



(a) two processors using the list $T_1, \ldots, T_7$.
(b) two processors using the list $T_1, T_2, T_3, T_4, T_6, T_5, T_7$.
(c) Is either of the schedules that you obtain optimal?

15. Can you find a list that gives rise to the optimal schedule shown in Figure 3.14 for the order-requirement digraph in Figure 3.12?

16. Consider the following order-requirement digraph:



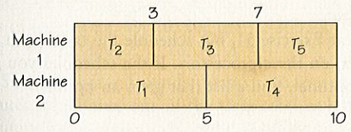
(a) Find the critical path(s).

(b) Schedule these tasks on one processor using the critical-path scheduling method.
(c) Schedule these tasks on one processor using the priority list obtained by listing the tasks in order of decreasing time.
(d) Does either of these schedules have idle time? How do their completion times compare?
(e) If two different schedules have the same completion time, what criteria can be used to say one schedule is superior to the other?
(f) Schedule these tasks on two processors using the order-requirement digraph shown and the priority list from part (b).
(g) Does the schedule produced in part (f) finish in half the time that the schedule in part (b) did, which might be expected, since the number of processors has doubled?
(h) Schedule the tasks on (i) one processor and (ii) two processors (using the decreasing-time list), assuming that each task time has been reduced by one. Do the changes in completion time agree with your expectations?

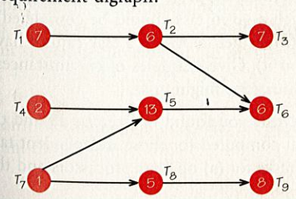
17. (a) Can all the processors being used to schedule tasks be simultaneously idle at a time before the completion time of a collection of tasks scheduled using the list-processing algorithm?
(b) Explain why the list-processing algorithm cannot give rise to the schedule below, regardless of what priority list was used to schedule the tasks on the two processors.

Machine 1	T_1	T_4	T_6
Machine 2	T_2		T_7
Machine 3	T_3	T_5	

(c) Construct an order-requirement digraph and a priority list that will give rise to the following schedule on two processors.

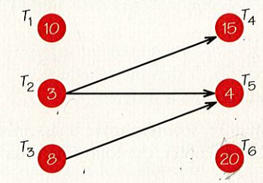


18. To prepare a meal quickly involves carrying out the tasks shown (time lengths in minutes) in the following order-requirement digraph:



(a) If Mike prepares the meal alone, how long will it take?
(b) If Mike can talk Mary into helping him prepare the meal, how long will it take them if the tasks are scheduled using the list $T_5, T_9, T_1, T_3, T_2, T_6, T_8, T_4, T_7$ and the list-processing algorithm?
(c) If Mike can talk Mary and Jack into helping him prepare the meal, how long will it take if the tasks are scheduled using the same list as in part (b)?
(d) What would be a reasonable set of criteria for choosing a priority list in this situation?

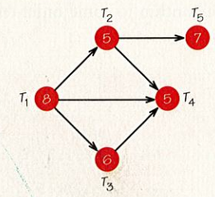
19. (a) Making use of the order-requirement digraph below, determine at time 0 which tasks are ready.



(b) What is special about tasks T_1 and T_6?
(c) What is the critical path, and what is its length?
(d) Schedule the tasks on three processors with the priority list $T_1, \ldots, T_6$.
(e) Is the schedule found in part (d) optimal?
(f) Schedule the tasks on three processors using the priority list $T_6, \ldots, T_1$.
(g) Is the schedule found in part (f) optimal?
(h) Can you find a priority list that yields an optimal schedule?

20. (a) In Exercise 19, what priority list would be used if you applied the critical-path scheduling method?
(b) Use this priority list to schedule the tasks on three processors. Is this schedule optimal?
(c) How does this schedule compare with the schedules that you found using the lists in Exercise 19?

21. Consider the order-requirement digraph below. Suppose one plans to schedule these tasks on two identical processors.



(a) How many different priority lists can be used to schedule the tasks?
(b) Can all these priority lists lead to different schedules? If not, why not?
(c) Can an optimal schedule have no idle time? Can you give two different reasons why an optimal schedule must have some idle time?