Chapter 3
Planning and Scheduling

Chapter Outline

Introduction
Section 3.1 Scheduling Tasks
Section 3.2 Critical-Path Schedules
Section 3.3 Independent Tasks
Section 3.4 Bin Packing
Section 3.5 Resolving Conflicts via Coloring

Chapter Summary

When scheduling tasks for processors (which can be either machines or people), we can consider two types of problems. In the first type, we have a fixed number of processors, each with unlimited time. In the second, we have an unlimited number of processors, each with the same fixed time to work.

The *list-processing algorithm* is an approach to solving the first type of problem. We assume we are given an *order-requirement digraph* for the tasks and a priority list for the tasks that can be independent of the order requirements. Additionally, we need two assumptions on the processors: (1) a processor works on an assigned task without interruption until the task is completed; and (2) no processor is ever voluntarily idle. Simply stated, the algorithm assigns the highest priority task currently ready to the lowest numbered processor currently idle. The schedule obtained by this algorithm will be optimal if its overall completion time equals the length of the critical path in the order-requirement digraph. The list-processing algorithm does not always produce an optimum schedule. Moreover, it is subject to several paradoxical quirks, which can occur when attempts are made to shorten the overall completion time.

Other scheduling methods can be used. *Critical-path scheduling* works first on those tasks that occur early in long paths of the order-requirement digraph. If the tasks being scheduled are independent (can be done in any order), the *decreasing-list algorithm* can be employed (schedule longer tasks first). Unfortunately, the scheduling problem is of the same character as the TSP. No computationally efficient method is known that will always produce an optimal schedule.

The second type of problem is a particular instance of the *bin-packing problem*: What is the least number of containers (of the same fixed capacity) needed to pack a given list of items (of various sizes, each less than the capacity of the bins). For this problem, we must again employ heuristic methods. *Next-fit* places items in a container until the next one won't fit and then moves on to a new container. *First-fit* keeps an ordered list of containers and places the next item in the first container in which it fits. *Worst-fit* is similar to first-fit, but places the next item in the container with the most leftover space.

There are many situations in which a schedule or a configuration must be produced that avoids conflicts. Examples include legislative bodies, in which representatives serve on several committees. If one or more members serve on two particular committees, then meetings of those committees must be scheduled at different times. Similarly, final examinations in a university are best arranged in a way that no student has a conflict between two exams scheduled at the same time. Since the number of slots for examinations is limited, this requirement produces a difficult scheduling problem. What is the minimum number of slots required to accommodate all of the examinations without conflicts?

Graph theory—in particular, *vertex coloring*—can be of use in solving such problems. Construct a graph in which vertices are joined by an edge if they are incompatible; e.g., two courses would be joined if there is at least one student in both of the classes. We then color the vertices in such a way that adjacent vertices (those joined by an edge) have different colors. The minimum number of colors required to accomplish this is called the *chromatic number* of the graph, and it yields the solution to the problem. Unfortunately, no algorithm for finding the chromatic number of graphs consisting of a large number of vertices is currently known.

Skill Objectives

1. State the assumptions for the scheduling model.

2. Compute the lower bound on the completion time for a list of independent tasks on a given number of processors.

3. Describe the list-processing algorithm.

4. Apply the list-processing algorithm to schedule independent tasks on identical processors.

5. For a given list of independent tasks, compare the total task time using the list-processing algorithm for both the non-sorted list and also a decreasing-time list.

6. When given an order-requirement digraph, apply the list-processing algorithm to schedule a list of tasks subject to the digraph.

7. Explain how a bin-packing problem differs from a scheduling problem.

8. Given an application, determine whether its solution is found by the list-processing algorithm or by one of the bin-packing algorithms.

9. Discuss advantages and disadvantages of the next-fit, bin-packing algorithm.

10. Solve a bin-packing problem by the non-sorted, next-fit algorithm.

11. Solve a bin-packing problem by the decreasing-time, next-fit algorithm.

12. Discuss advantages and disadvantages of the first-fit, bin-packing algorithm.

13. Apply the non-sorted, first-fit algorithm to a bin-packing problem.

14. Apply the decreasing-time, first-fit algorithm to a bin-packing problem.

15. Discuss advantages and disadvantages of the worst-fit, bin-packing algorithm.

16. Find the solution to a bin-packing problem by the non-sorted, best-fit algorithm.

17. Find the solution to a bin-packing problem by the decreasing-time, worst-fit algorithm.

18. List two examples of bin-packing problems.

19. Color the vertices of a graph so that adjacent vertices have different colors.

20. Determine the chromatic number of a graph.

21. Apply vertex coloring to produce schedules that avoid conflicts.

Teaching Tips

1. When applying the list-processing algorithm, some students find it helpful to construct a grid both above and below the rectangular diagram of processors so that task times can be diagrammed more accurately.

2. An emphasis on the fact that the list-processing algorithm requires a task to be placed on the lowest-numbered available processor at a given time can avoid a decision making problem some students face when more than one processor is available at the same time.

3. Working with an order-requirement digraph in conjunction with the list-processing algorithm poses a difficult process for some students. It may require a great deal of time and several examples. It's often helpful to work with students through at least one homework problem of this type. In addition, listing the task numbers in the rectangles on the scheduling diagram sometimes aids students in keeping on track.

4. Although the three bin-packing algorithms appear to be stated quite simply, many students have difficulty following their instructions. Careful and thorough explanation of these directions can be of help to students. Working through one example using all six versions of the bin-packing algorithms can pay learning dividends.

5. Students often question the need for the next-fit algorithm, since it often leaves bins only partially filled. This is most suitable when processors cannot be kept waiting, even if they are only partially full, as, for example, with delivery trucks being filled with perishable items.

6. It may be helpful to introduce the concept of bin-packing as simply a variant of the scheduling problem in which the processors are of fixed capacity with the number of processors variant. This offers the advantage of looking at the problem from a different perspective.

7. The problem of whether or not to introduce idle time on a processor when tasks are ordered according to a digraph creates difficulty for some students. It can be helpful to have them keep a list of tasks that are being put on hold temporarily because their predecessors have not yet been completed. Then, as a processor becomes ready for a task, the students should check the list before going on to new tasks, to see if any waiting task can now be scheduled. The only situation in which idle time can appear on the schedule before all of the tasks have been assigned is when no tasks were on hold and no new tasks can be scheduled.

Research Paper

Have students research the "knapsack problem". In this scenario you are given a set of items, each with a cost and a value. The task is to determine the number of each item to include in the knapsack so that the total cost is less than some given cost and one has the total value as large as possible. Students should compare this problem and its applications to the ones presented in this chapter.

Have students research the contributions of R. L. Brooks and what is known as Brooks' Theorem. This theorem pertains to an upper bound on the chromatic number of a graph.

Collaborative Learning

Scheduling

Two people are camping out and wish to cook a simple supper consisting of soup and hamburgers. In order to prepare the supper, several tasks must be performed.

***T*1:** Go to the store to buy matches – 10 minutes

***T*2:** Collect firewood– 8 minutes

***T*3:** Light the fire– 6 minutes

***T*4:** Get water from a well– 12 minutes

***T*5:** Cook soup (requires constant stirring)– 15 minutes

***T*6:** Make hamburger patties – 9 minutes

***T*7:** Cook hamburgers– 7 minutes

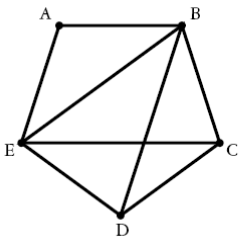
Each task requires only one of the campers and once a task is begun it cannot be interrupted.

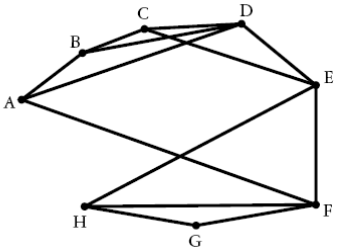
Some of the tasks can be done simultaneously, but some cannot be done until others are completed. What is the shortest amount of time it will take for all of the tasks to be completed?

How should the work be divided between the two campers?

Vertex Coloring

Color the vertices of the following graphs in such a way that adjacent vertices have different colors. What is the minimum number of colors required for each graph?





Conflict Resolution

A man wants to set up an aquarium in his home to contain five different types of fish—A, B, C, D, and E. However, some of these fish will eat other ones and thus cannot share a tank. The chart below displays the incompatibilities. (An X indicates that the two species cannot share a tank.)

What is the minimum number of tanks required, and which fish go into which tank?

	A	B	C	D	E
A		X		X	
B	X		X		X
C		X		X	
D	X		X		
E		X			

Solutions

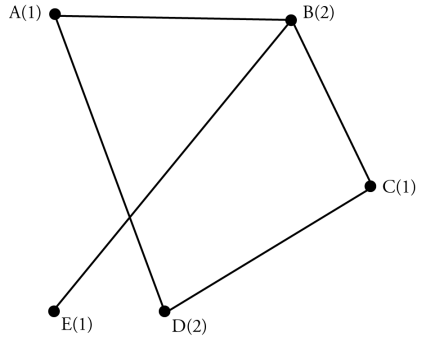
Skills Check:

1. c	2. c	3. b	4. b	5. c	6. b	7. c	8. b	9. b	10. c
11. b	12. b	13. c	14. a	15. b	16. a	17. c	18. a	19. a	20. c

Cooperative Learning:

Conflict Resolution:

Set up the graph corresponding to the conflict matrix:



Exercises:

1. (a) Scheduling final examinations, the times for course being offered, cleaning of classrooms, etc.

(b) Scheduling the buses, the selling of tickets, cleaning, etc.

(c) Scheduling regular staff and supervisory staff, deliveries of items for sale, etc.

(d) Scheduling meal times, shopping, etc.

(e) Scheduling staff, maintenance of backup equipment, etc.

(f) Scheduling nurses, doctors, operating rooms, medical imaging, etc.

(g) Scheduling maintenance and repair of hoses and equipment, personnel to respond to alarms.

(h) Scheduling store personnel, inventory work, shelving new books, and restoring books to proper order on the shelves.

(i) Scheduling ground and squad car patrols around the clock, officers on surveillance assignments, general work involving nonemergency interaction with the public; arrangements must be made regarding who is on call for emergency duty.

2. Tasks include purchasing food, preparing the food for cooking, cooking the food, setting the table, cleaning the house, etc. The processors involved include the parents and three children, the stove (which cooks the turkey), etc. Since the human processors can work in parallel, some of these tasks can be done simultaneously. Some of the human processors can also work simultaneously with the stove.

3. Jocelyn must perhaps launder her clothes, arrange care for her cat, pack, arrange for a taxi to the airport, and get to the airport. Unless she can get a friend to help her with some of these tasks, she must do all the tasks herself. She can launder her clothes during the time she arranges for a taxi, but most of the tasks cannot be done simultaneously.

4. (a) The critical path has length 32.

(b) (1) Processor 1: T_1, T_3, T_5, idle 30 to 44.
Processor 2: T_2, T_4, T_6, T_7, idle 36 to 38.

(2) Processor 1: T_2, T_5, T_6, T_7.
Processor 2: T_1, T_3, T_4, idle 33 to 41.

(c) No.

(d) The sum of the task time divided by 2 is 37. Hence, no schedule can finish earlier than time 37.

5. (a) i. Processor 1: T_1 from 0 to 13, T_3 from 13 to 25, T_6 from 25 to 45.
Processor 2: T_2 from 0 to 18, T_4 from 18 to 27, T_5 from 27 to 35, idle from 35 to 45.

ii. Processor 1: T_1 from 0 to 13, T_3 from 13 to 25, T_4 from 25 to 34, T_5 from 34 to 42.
Processor 2: T_2 from 0 to 18, T_6 from 18 to 38, idle from 38 to 42.

(b) The schedule produced in (ii) is optimal, because the sum of the task times is 80 and no set of tasks can be arranged that will feasibly sum to 40 on each processor.

(c) The critical path is T_2, T_6, and it has length 38. No schedule can be completed by time 38 on two processors because the sum of the task times divided by 2 is 40.

6. (a) i. Processor 1: T_1, T_3, T_6. Processor 2: T_2, T_4, T_5, idle 37 to 43.

ii. Processor 1: T_1, T_3, T_4, T_5. Processor 2: T_2, T_6, idle 38 to 42. The second is optimal. The critical path is T_2, T_6 and has length 38. No schedule on two machines finishes by 38 because the total time for all the tasks is 80. Thus, on two machines no earlier time than $\frac{80}{2} = 40$ is possible.

(b) The optimum completion time before and after the two task times are switched is the same.

7. (a) Processor 1: T_1, T_2, T_3, T_5, T_7.
Processor 2: Idle 0 to 2, T_4, T_6, idle 4 to 5.

(b) Processor 1: T_1, T_2, T_3, T_6, T_7.
Processor 2: Idle 0 to 2, T_4, T_5, idle 4 to 5.

(c) Yes.

(d) No.

(e) T_3 and T_5.

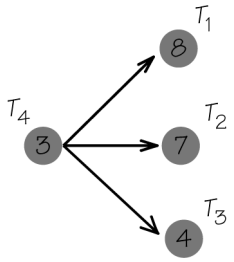
8. Hospital emergency room activities and police officer activities may have to be interrupted to start new tasks. For manufacturing machines in standard production runs, the assumption is reasonable.

9. (a) Processor 1: T_1, T_6, idle 15 to 21, T_7, idle 27 to 31.
Processor 2: T_2, T_5, T_8.
Processor 3: T_3, T_4, idle from 13 to 31.

(b) Processor 1: T_1, T_6, idle 15 to 21, T_7, idle 27 to 31.
Processor 2: T_3, T_4, idle from 13 to 21, T_8.
Processor 3: T_2, T_5, idle from 21 to 31.

(c) Processor 1: T_4, idle 10 to 11, T_6, idle 18 to 21, T_8.
Processor 2: T_2, T_5, T_7, idle 27 to 31.
Processor 3: T_1, T_3, idle 11 to 31.

10. (a) The order-requirement diagraph has only one task T_4 ready at time 0. Thus, no matter what list one starts with, one will skip over any tasks that are prior to T_4 in the list until one reaches T_4 because any other task would not be ready at time 0. Of course those lists that start with T_4 would assign T_4 to Machine 1 at time 0 because it is ready at this time.



(b) So by part (a) we always assign T_4 to Machine 1 at time 0, which keeps that machine busy until time 3. Now, can we assign any task to machine 2 at any time between 0 and 3? No, because all of those tasks must wait until T_4 is done before they can begin. So Machine 2 stays idle from time 0 to time 3. Thus the same order-requirement digraph used for (a) works here.

11. Examples include inserting identical mirror systems on different models of cars and vaccinating different children against polio.

12. (a) Processor 1: $T_1,\ T_2,\ T_4,\ T_6,\ T_7$.
Processor 2: idle 0 to 1, $T_3,\ T_5,$ idle 3 to 5.

(b) Processor 1: $T_1,\ T_2,\ T_4,\ T_5$.
Processor 2: idle 0 to 1, $T_3,\ T_6,\ T_7$.

(c) The schedule in (b) is optimal.

13. $T_1,\ T_2,\ T_3,\ T_4,\ T_8,\ T_9,\ T_{10},\ T_{11},\ T_5,\ T_6,\ T_7,\ T_{12}$.

14. (a) The critical path, which has length 17, is $T_1,\ T_2,\ T_3$.

(b) $T_1,\ T_4,\ T_5,\ T_2,\ T_6,\ T_7,\ T_3$ is the list to be used. The one processor would have the tasks scheduled on it: $T_1,\ T_4,\ T_5,\ T_2,\ T_6,\ T_7,\ T_3$.

(c) $T_6,\ T_1,\ T_7,\ T_2,\ T_4,\ T_3,\ T_5$ would be the list. The resulting schedule on one processor would be: $T_1,\ T_7,\ T_4,\ T_2,\ T_5,\ T_6,\ T_3$.

(d) No idle time. Their completion times are the same.

(e) Earlier completion of tasks giving rise to cash payments.

(f) The required schedule is Processor 1: $T_1,\ T_6,\ T_3$; Processor 2: $T_4,\ T_5,\ T_2,\ T_7$.

(g) The completion time does halve $(40 \to 20)$. As the number of processors goes up, the completion time may decrease, but at some point the length of the critical path will govern the completion time rather than the number of processors.

(h) (i) Completion time goes down by 7.
(ii) Completion time is 19 for two processors using the decreasing time list.

15. (a) No. Consider the tasks that begin after the stretch where all machines are idle. Pick one of these tasks T and say machine 1 was the machine that it was given to. This task was ready for machine 1 just prior to when it began T because no task was just being completed on any other machine at this time because they were all idle. Thus, T should have begun earlier on machine 1.

(b) This schedule cannot arise using the list-processing algorithm, because T_2 should have been scheduled at time 0.

(c) Use the digraph with no edges and the list: $T_2,\ T_1,\ T_3,\ T_4,\ T_5$.

16. (a) 55 minutes.

(b) Mike: $T_1,\ T_2,\ T_3,\ T_8,\ T_9$.
Mary: $T_4,\ T_7,\ T_5,\ T_6,$ idle 22–33.

(c) Mike: $T_1,\ T_2,\ T_3,$ idle 20–21.
Mary: $T_4,\ T_5,\ T_6$.
Jack: $T_7,\ T_8,\ T_9,$ idle 14–21.

(d) Tasks that, when completed, result in hot food for eating should be completed as close to the end as possible.

17. (a) $T_1,\ T_2,\ T_3,$ and T_6 are ready at time 0.

(b) No tasks require that T_1 and T_6 be done before these other tasks can begin.

(c) The critical path consists only of T_6 and has length 20.

(d) Processor 1: $T_1,\ T_6$: Processor 2: $T_2,\ T_4,$ idle from 18 to 30: Processor 3: $T_3,\ T_5,$ idle from 12 to 30.

(e) No.

(f) Processor 1: $T_6,$ idle from 20 to 22: Processor 2: $T_3,\ T_5,\ T_1$: Processor 3: $T_2,\ T_4,$ idle from 18 to 22.

(g) Yes.

(h) Another list leading to the same optimal schedule is $T_6,\ T_3,\ T_2,\ T_4,\ T_5,\ T_1$.

18. (a) $T_6,\ T_2,\ T_4,\ T_3,\ T_1,\ T_5$.

(b) Processor 1: $T_6,$ idle 20–22;
Processor 2: $T_2,\ T_4,\ T_5$;
Processor 3: $T_3,\ T_1,$ idle 18–22.
This schedule is optimal for 3 processors.

(c) It is no better than the best schedule found there.

19. (a) 5! = 120

(b) No. Whatever list is used, T_1 must be assigned to the first machine at time 0 because it is the only task ready at time 0.

(c) No. First, while Processor 1 works on T_1. Processor 2 must be idle. Second, the task times are integers with sum 31. If there are two processors, one of the processors must have idle time since when 2 divides 31, there is a remainder of 1.

(d) No.

20. (a) $120 - 24 = 96$.

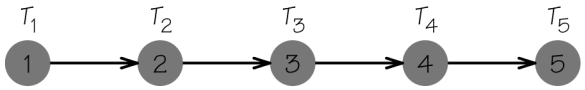
(b) No, because Task 1 is a predecessor of all the other tasks.

(c) Answers will vary.

(d) Answers will vary.

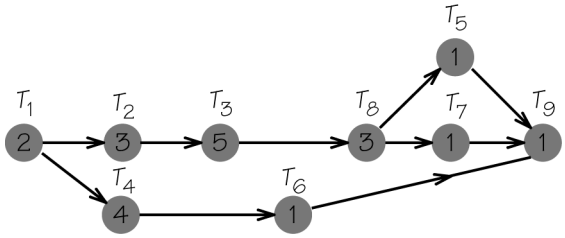
(e) The list $T_1,\ T_3,\ T_2,\ T_5,\ T_4$ will yield an optimal schedule.

21. Using the order-requirement digraph shown and any list with one or more processors yields the same schedule:



22. If one uses the digraph with no edges and schedules three tasks whose completion times are 6, 8, and 14 on one machine, each of the 6 possible scheduling lists will yield a different schedule.

23. (a) One reasonable possibility is (time in min):



The earliest completion time is 15.

(b) The decreasing-time list is $T_3,\ T_4,\ T_2,\ T_8,\ T_1,\ T_5,\ T_6,\ T_7,\ T_9$. The schedule is Processor 1: $T_1,\ T_4,\ T_6,$ idle 7 to 10, $T_8,\ T_5,\ T_9$; Processor 2: idle 0 to 2, $T_2, T_3,$ idle 10 to 13, $T_7,$ idle 14 to 15.

24. Not if the number of processors is the same. If the number of processors is different, two schedules can end at the same time and have different amounts of idle time.

25. No. At time 11, T_4 should been assigned to Machine 1 because Machine 1 was free at this time and T_4 was ready.

26. No. If T_3 could be scheduled at time 12 on Robot 1, the order requirement digraph could not have required that this task await the completion of T_2. In this case, this task could have been started at time 8 on Robot 1.

27. (a) Task times: $T_1 = 3,\ T_2 = 3,\ T_3 = 2,\ T_4 = 3,\ T_5 = 3,\ T_6 = 4,\ T_7 = 5,\ T_8 = 3,\ T_9 = 2,\ T_{10} = 1,$ $T_{11} = 1,$ and $T_{12} = 3$. This schedule would be produced from the list: $T_1,\ T_3,\ T_2,\ T_5,\ T_4,$ $T_6,\ T_7,\ T_8,\ T_{11},\ T_{12},\ T_9,\ T_{10}$.

(b) Task times: $T_1 = 3,\ T_2 = 3,\ T_3 = 3,\ T_4 = 2,\ T_5 = 2,\ T_6 = 4,\ T_7 = 3,\ T_8 = 5,\ T_9 = 8,\ T_{10} = 4,$ $T_{11} = 7,\ T_{12} = 9,$ and $T_{13} = 3$. This schedule would be produced from the list: $T_1,\ T_5,\ T_7,$ $T_4,\ T_3,\ T_6,\ T_{11},\ T_8,\ T_{12},\ T_9,\ T_2, T_{10}, T_{13}$.

28. Tasks could be rearranged in order of increasing time on each machine; tasks could be reordered so those that, when completed, resulted in a cash payment were completed as early as possible.

29. (a) (i) Processor 1: T_1, T_3, T_5, T_7, idle from 16 to 20; Processor 2: T_2, T_4, T_6, T_8.
(ii) Processor 1: T_8, T_5, T_4, T_1; Processor 2: T_7, T_6, T_3, T_2.

(b) The schedule in (ii) is optimal.

30. (a) (i) Processor 1: T_1, T_4, T_7, idle 12–15; Processor 2: T_2, T_5, T_8;
Processor 3: T_3, T_6, idle 9–15.
(ii) Processor 1: T_8, T_3, T_2; Processor 2: T_7, T_4, T_1, idle 12–13;
Processor 3: T_6, T_5, idle 11–13.

(b) Neither of these schedules are optimal. An optimal schedule is achieved when the list T_8, T_7, T_2, T_3, T_6, T_5, T_4 is used. The idea for finding an optimal schedule is based on the fact that the total task time is 36.

31. Such criteria include decreasing length of the times of the tasks, order of size of financial gains when each task is finished, and increasing length of the times of the tasks.

32. Examples of scheduling projects with due dates include construction projects, military procurement projects, and plane departures from the gates at an airport. An example of a situation where there might be release times would be for planes to enter a runway holding area or for a construction project where materials are delivered to a site but cannot arrive too early lest this get in the way of current activities because there is little storage space at the site.

33. (a) Machine 1: T_1, T_6, T_{10}, idle from 8 to 9; Machine 2: T_3, T_4, T_{11}, T_{12}; Machine 3: T_2, T_7, idle from 8 to 9; Machine 4: T_5, T_8, T_9, idle from 8 to 9.

(b) Machine 1: T_1, T_8, T_{10}; Machine 2: T_5, T_6, T_2, T_{13}; Machine 3: T_7, T_{12}; Machine 4: T_4, T_{11}, idle from 9 to 12; Machine 5: T_3, T_9, idle from 11 to 12.

The schedule in part (a) had to have idle time because the total task time was 33, and 33 is not exactly divisible by 4. The schedule in part (b) had to have idle time because the total task time was 56, and 56 is not exactly divisible by 5.

34. (a) (i) Processor 1: T_1, T_2, T_5, T_6.
Processor 2: T_3, T_7, T_4, idle 30–56.
(ii) Processor 1: T_1, T_4, T_7, T_6.
Processor 2: T_3, T_2, T_5, idle 31–55.

The decreasing time list is: T_6, T_5, T_4, T_1, T_7, T_2, T_3, yielding the schedule:
Processor 1: T_1, T_4, T_7, T_6.
Processor 2: T_2, T_3, T_5, idle 31–55.

(b) These schedules are not optimal since the list T_3, T_2, T_1, T_7, T_5, T_4, T_6, yields the schedule:
Processor 1: T_3, T_1, T_5, T_6.
Processor 2: T_2, T_7, T_4, idle 32–54.
The completion time is 54.

(c) The critical path method gives rise to the list: T_1, T_2, T_3, T_5, T_4, T_7, T_6. This list gives rise to the schedule:
Processor 1: T_1, T_4, T_7, T_6.
Processor 2: T_2, T_3, T_5, idle 31–55.

35. (a) List for (i) yields (with items coded by task time): Machine 1: 12, 9, 15, idle from 36 to 50; Machine 2: 7, 10, 13, 20. List for (ii) yields: Machine 1: 12, 13, 20; Machine 2: 7, 9, 15, 10, idle from 41 to 45. List for (iii) yields: Machine 1: 20, 12, 9, idle from 41 to 45; Machine 2: 15, 13, 10, 7.

(b) These schedules complete earlier than those where precedence constraints hold. An optimal schedule is possible, however: Machine 1: 20, 10, 13; Machine 2: 12, 15, 9, 7. The associated list is: T_6, T_1, T_5, T_7, T_2, T_4, T_3.

(c) The critical path list is T_6, T_5, T_4, T_1, T_7, T_2, T_3 using the first processor. It finishes at time 41 and is idle until 45, when processor 2 finishes.

36. (a) With this list the order of the tasks on the processors is: Machine 1: 8, 17, 16, 5, 3, 7, 2, 1, idle 59 to 61; Machine 2: 11, 14, 9, 2, 1, 18, 6. The completion time is 61.

(b) The list is: 18, 17, 16, 14, 11, 9, 8, 7, 6, 5, 3, 2, 2, 1, 1. On two processors we get the schedule: Machine 1: 18, 14, 11, 7, 6, 2, 2; Machine 2: 17, 16, 9, 8, 5, 3, 1, 1. The completion time is 60 on both machines.

(c) The answer in (b) is one optimal schedule

(d) Using the original list we get the schedule: Machine 1: 19, 20, 1, 2, 3, 5, 11, 17, 18, 2, 16; Machine 2: 19, 20, 1, 2, 3, 5, 11, 18, 17, 16, 2. Both machines finish at time 114. The decreasing list is: 20, 20, 19, 19, 18, 18, 17, 17, 16, 16, 11, 11, 5, 5, 3, 3, 2, 2, 2, 2, 1, 1. Machine 1: 20, 19, 18, 17, 16, 11, 5, 3, 2, 2, 1; Machine 2: 20, 19, 18, 17, 16, 11, 5, 3, 2, 2, 1. Both machines finish at time 114. Both lists lead to optimal schedules.

37. (a) The tasks are scheduled on the machines as follows: Processor 1: 12, 13, 45, 34, 63, 43, 16, idle 226 to 298; Processor 2: 23, 24, 23, 53, 25, 74, 76; Processor 3: 32, 23, 14, 21, 18, 47, 23, 43, 16, idle 237 to 298.

(b) The tasks are scheduled on the machines as follows: Processor 1: 12, 24, 14, 34, 25, 23, 16, 16, 76; Processor 2: 23, 23, 21, 63, 43, idle 173 to 240; Processor 3: 32, 23, 53, 74, idle 182 to 240; Processor 4: 13, 45, 18, 47, 43, idle 166 to 240.

(c) The decreasing-time list is 76, 74, 63, 53, 47, 45, 43, 43, 34, 32, 25, 24, 23, 23, 23, 23, 21, 18, 16, 16, 14, 13, 12. The tasks are scheduled on three machines as follows: Processor 1: 76, 45, 43, 24, 23, 18, 16, 13; Processor 2: 74, 47, 34, 32, 23, 21, 14, 12, idle 257 to 258; Processor 3: 63, 53, 43, 25, 23, 23, 16, idle 246 to 248. The tasks are scheduled on four machines as follows: Processor 1: 76, 43, 24, 23, 16, idle 182 to 194; Processor 2: 74, 43, 25, 23, 16, 13; Processor 3: 63, 45, 32, 23, 18, 12, idle 193 to 194; Processor 4: 53, 47, 34, 23, 21, 14, idle 192 to 194.

(d) The new decreasing time list is 84, 82, 71, 61, 55, 45, 43, 43, 34, 32, 25, 24, 23, 23, 23, 23, 21, 18, 16, 16, 14, 13, 12. The tasks are scheduled as follows: Processor 1: 84, 45, 43, 25, 23, 23, 16, 12; Processor 2: 82, 55, 34, 32, 23, 18, 14, 13; Processor 3: 71, 61, 43, 24, 23, 21, 16, idle 259 to 271.

38. T_2, T_3, T_1, T_4, T_5; decreasing time list is T_4, T_5, T_3, T_2, T_1. Machine 1: T_4, T_2, T_1; Machine 2: T_5, T_3, idle 28 – 32. The completion time is 32.

39. Examples include jobs in a videotape copying shop, data entry tasks in a computer system, and scheduling nonemergency operations in an operating room. These situations may have tasks with different priorities, but there is no physical reason for the tasks not to be independent, as would be the case with putting on a roof before a house had walls erected.

40. An algorithm that will give an optimal answer every time is the list-processing algorithm using any list. The total time to complete all the tasks will be nk. When the number of machines is smaller than the number of tasks, the completion time will depend on whether the number of processors exactly divides n. An example where all the tasks times might be equal would be to fill a collection of identical boxes with identical objects.

41. Each task heads a path of length equal to the time to do that task.

42. Case I: Using 10-foot horizontal shelves and 6-foot vertical boards, the list of decreasing board lengths is 10, 10, 10, 10, 6, 6, 2, 2, 2, 2.

First-fit-decreasing:
BIN 1: 10;
BIN 2: 10;
BIN 3: 10;
BIN 4: 10;
BIN 5: 6, 2, 2;
BIN 6: 6, 2, 2.

Next-fit-decreasing:
BIN 1: 10;
BIN 2: 10;
BIN 3: 10;
BIN 4: 10;
BIN 5: 6;
BIN 6: 6;
BIN 7: 2, 2, 2, 2.

Worst-fit-decreasing:
BIN 1: 10;
BIN 2: 10;
BIN 3: 10;
BIN 4: 10;
BIN 5: 6, 2, 2;
BIN 6: 6, 2, 2.

Case II: Using boards that add up to 10- and 6-foot lengths, the list of decreasing board lengths is 7, 5, 5, 4, 4, 3, 3, 3, 3, 3, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2.

First-fit-decreasing:
BIN 1: 7, 3;
BIN 2: 5, 5;
BIN 3: 4, 4, 2;
BIN 4: 3, 3, 3;
BIN 5: 3, 2, 2, 2;
BIN 6: 2, 2, 2, 2, 2;
BIN 7: 2.

Next-fit-decreasing:
BIN 1: 7;
BIN 2: 5, 5;
BIN 3: 4, 4;
BIN 4: 3, 3, 3;
BIN 5: 3, 3, 2, 2;
BIN 6: 2, 2, 2, 2, 2;
BIN 7: 2, 2, 2.

Worst-fit-decreasing:
BIN 1: 7, 3;
BIN 2: 5, 5;
BIN 3: 4, 4, 2;
BIN 4: 3, 3, 3;
BIN 5: 3, 2, 2, 2;
BIN 6: 2, 2, 2, 2, 2;
BIN 7: 2.

By using the 10- and 6-foot boards, you need to buy 1 less board. Also, the system will be easier to make structurally sound with single pieces of wood. You are encouraged to try other combinations of board lengths.

43. The times to photocopy the manuscripts, in decreasing order, are 120, 96, 96, 88, 80, 76, 64, 64, 60, 60, 56, 48, 40, 32. Packing these in bins of size 120 yields Bin 1: 120; Bin 2: 96; Bin 3: 96; Bin 4: 88, 32; Bin 5: 80, 40; Bin 6: 76; Bin 7: 64, 56; Bin 8: 64, 48; Bin 9: 60, 60. Nine photocopy machines are needed to finish within 2 minutes using FFD. The number of bins would not change, but the placement of the items in the bins would differ for worst-fit decreasing.

44. The bins have capacity 135.
First-fit uses 5 bins.
Bin 1: 80, 50;
Bin 2: 90, 20;
Bin 3: 130;
Bin 4: 60, 30, 30;
Bin 5: 90, 40.
The decreasing list is: 130, 90, 90, 80, 60, 50, 40, 30, 30, 20.
First-fit decreasing uses 5 bins.
Bin 1: 130;
Bin 2: 90, 40;
Bin 3: 90, 30;
Bin 4: 80, 50;
Bin 5: 60, 30, 20.
Either of these packings is optimal.

Using first-fit on the list: 60, 50, 40, 40, 60, 90, 90, 50, 20, 30, 30, 50 yields (bin capacity 135):
Bin 1: 60, 50, 20;
Bin 2: 40, 40, 50;
Bin 3: 60, 30, 30;
Bin 4: 90;
Bin 5: 90;
Bin 6: 50.
The decreasing list is: 90, 90, 60, 60, 50, 50, 50, 40, 40, 30, 30, 20.
Using first-fit on this list yields:
Bin 1: 90, 40;
Bin 2: 90, 40;
Bin 3: 60, 60;
Bin 4: 50, 50, 30;
Bin 5: 50, 30, 20.
The decreasing time solution is optimal and uses one fewer bin than first-fit applied to the original list.

45. (a) Using the next-fit algorithm, the bins are filled as follows: Bin 1: 12, 15; Bin 2: 16, 12; Bin 3: 9, 11, 15; Bin 4: 17, 12; Bin 5: 14, 17; Bin 6: 18; Bin 7: 19; Bin 8: 21; Bin 9: 31; Bin 10: 7, 21; Bin 11: 9, 23; Bin 12: 24; Bin 13: 15, 16; Bin 14: 12, 9, 8; Bin 15: 27; Bin 16: 22; Bin 17: 18.

(b) The decreasing list is 31, 27, 24, 23, 22, 21, 21, 19, 18, 18, 17, 17, 16, 16, 15, 15, 15, 14, 12, 12, 12, 12, 11, 9, 9, 9, 8, 7. The next-fit decreasing schedule is Bin 1: 31; Bin 2: 27; Bin 3: 24; Bin 4: 23; Bin 5: 22; Bin 6: 21; Bin 7: 21; Bin 8: 19; Bin 9: 18, 18; Bin 10: 17, 17; Bin 11: 16, 16; Bin 12: 15, 15; Bin 13: 15, 14; Bin 14: 12, 12, 12; Bin 15: 12, 11, 9; Bin 16: 9, 9, 8, 7.

(c) The worst-fit schedule using the original list is Bin 1: 12, 15, 9; Bin 2: 16, 12; Bin 3: 11, 15; Bin 4: 17, 12; Bin 5: 14, 17; Bin 6: 18, 7; Bin 7: 19, 9; Bin 8: 21, 15; Bin 9: 31; Bin 10: 21, 9; Bin 11: 23, 8; Bin 12: 24; Bin 13: 16, 12; Bin 14: 27; Bin 15: 22; Bin 16: 18.

(d) The worst-fit decreasing schedule would be Bin 1: 31; Bin 2: 27, 9; Bin 3: 24, 12; Bin 4: 23, 12; Bin 5: 22, 14; Bin 6: 21, 15; Bin 7: 21, 15; Bin 8: 19, 17; Bin 9: 18, 18; Bin 10: 17, 16; Bin 11: 16, 15; Bin 12: 12, 12, 11; Bin 13: 9, 9, 8, 7.

46. (a) (Next-fit):
Bin 1: 100, 120, 60, 90, 110;
Bin 2: 45, 30, 70, 60, 50, 40, 25, 65, 25, 55;
Bin 3: 35, 45, 60, 75, 30, 120, 100;
Bin 4: 60, 90 85.

(b) The decreasing list is: 120, 120, 110, 100, 100, 90, 90, 85, 75, 70, 65, 60, 60, 60, 60, 55, 50, 45, 45, 40, 35, 30, 30, 25, 25.
(Next-fit decreasing):
Bin 1: 120, 120, 110, 100;
Bin 2: 100, 90, 90, 85, 75;
Bin 3: 70, 65, 60, 60, 60, 60, 55, 50;
Bin 4: 45, 45, 40, 35, 30, 30, 25, 25.

(c) (First-fit): same solution as next-fit.

(d) (First-fit decreasing):
Bin 1: 120, 120, 110, 100, 30;
Bin 2: 100, 90, 90, 85, 75, 40;
Bin 3: 70, 65, 60, 60, 60, 60, 55, 50;
Bin 4: 45, 45, 35, 30, 25, 25.

47. The bins have a capacity of 120. (First fit): Bin 1: 63, 32, 11; Bin 2: 19, 24, 64; Bin 3: 87, 27; Bin 4: 36, 42; Bin 5: 63. This schedule would take five station breaks; however, the total time for the breaks is under 8 minutes. The decreasing list is 87, 64, 63, 63, 42, 36, 32, 27, 24, 19, 11. (First-fit decreasing): Bin 1: 87, 32; Bin 2: 64, 42, 11; Bin 3: 63, 36, 19; Bin 4: 63, 27, 24. This solution uses only four station breaks.

48. Bin 1: 8, 1;
Bin 2: 7, 2;
Bin 3: 9;
Bin 4: 5;
Bin 5: 7, 3;
Bin 6: 6, 4.

49. (a) There are theoretical results that show that best fit "usually" performs better than worst fit.

(b) Try 8, 7, 5, 3, 3, 2 in bins of capacity 14.

50. Bin 1: 6, 3, 1;
Bin 2: 9, 1;
Bin 3: 5, 2, 2, 1;
Bin 4: 8, 2;
Bin 5: 9;
Bin 6: 7, 3;
Bin 7: 5, 4;
Bin 8: 7, 2;
Bin 9: 6, 3;
Bin 10: 8, 2;
Bin 11: 7, 3;
Bin 12: 6, 4;
Bin 13: 5, 5;
Bin 14: 7, 2;
Bin 15: 2, 3, 5;
Bin 16: 6, 2, 2;
Bin 17: 7, 3;
Bin 18: 4, 6.

Continued on next page

50. continued
The decreasing list is: 9, 9, 8, 8, 7, 7, 7, 7, 7, 6, 6, 6, 6, 6, 5, 5, 5, 5, 5, 4, 4, 4, 3, 3, 3, 3, 3, 3, 3, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 1, 1.
Bin 1: 9, 1;
Bin 2: 9, 1;
Bin 3: 8, 2;
Bin 4: 8, 2;
Bin 5: 7, 3;
Bin 6: 7, 3;
Bin 7: 7, 3;
Bin 8: 7, 3;
Bin 9: 7, 3;
Bin 10: 6, 4;
Bin 11: 6, 4;
Bin 12: 6, 4;
Bin 13: 6, 3, 1;
Bin 14: 6, 2, 2;
Bin 15: 5, 5;
Bin 16: 5, 5;
Bin 17: 5, 2, 2;
Bin 18: 2, 2, 2.

51. The total performance time exceeds what will fit on four disks. Using FFD, one can fit the music on five disks.

52. Consider the list 8, 8, 5, 3, 5, 5, 5 to be packed in bins of capacity 11. Using first-fit, we get:
Bin 1: 8, 3;
Bin 2: 8;
Bin 3: 5, 5;
Bin 4: 5, 5.

When worst-fit is used, we get:
Bin 1: 8;
Bin 2: 8;
Bin 3: 5, 3;
Bin 4: 5, 5;
Bin 5: 5.
The list 8, 8, 5, 3, 5, 5, 5, 8, 8, 5, 5, 2 yields 7 bins when first-fit is used, 8 bins when worst-fit is used, and 9 bins when next-fit is used.

53. For problems with few weights to be packed, small integer weights, and a small integer as bin capacity, this method can work well. However, when these special conditions are not met, it is very time-consuming to carry out this method. For example, imagine trying to use this method for 2000 random real numbers of the form *.xyz*, where *x*, *y*, and *z* are decimal digits and the bin capacity is 1.

54. The amount of time available to record on each side of a record can be thought of as a bin capacity. *N* records would mean that *2N* bins would be available. However, since the movements should be configured consecutively, the algorithms we have developed might not yield acceptable solutions. Put somewhat differently, we are packing bins but with extra constraints on the packings. Tapes present similar problems to records since they involve taping in two directions on the same tape. In this sense (they do not have "two sides"), compact disks are closer to the model we have been using.

55. It makes sense to leave bins open as more items arrive to be packed if the cost of having many bins open at once is reasonable and there is room to have many partially-filled bins open without incurring great inconvenience or cost. One such example might be a company that has room for many identical trucks to park as they are loaded with goods to be delivered. There may be complex cost trade-offs between sending off fewer trucks because we wait to pack as much into each truck as possible and sending out more partially-filled trucks.

56. If one has to pack trucks of the same capacity using a single loading dock, it might be natural to send a truck on its way when the next item would not fit into the space left in the partially-filled truck. Any situation where keeping many bins up simultaneously is costly or storing an item temporarily is costly will encourage using an approach where when the next item will not fit into the currently open bin that bin is closed.

57. (a) The schedule with four secretaries is as follows: Processor 1: 25, 36, 15, 15, 19, 15, 27; Processor 2: 18, 32, 18, 31, 30, 18; Processor 3: 13, 30, 17, 12, 18, 16, 16, 16, 14; Processor 4: 19, 12, 25, 26, 18, 12, 24, 9.

The schedule with five secretaries is as follows: Processor 1: 25, 25, 31, 12, 16, 14; Processor 2: 18, 12, 17, 12, 15, 30, 9; Processor 3: 13, 32, 26, 16, 15, 18; Processor 4: 19, 36, 18, 19, 24; Processor 5: 30, 18, 15, 18, 16, 27.

(b) The decreasing time list is 36, 32, 31, 30, 30, 27, 26, 25, 25, 24, 19, 19, 18, 18, 18, 18, 18, 17, 16, 16, 16, 15, 15, 15, 14, 13, 12, 12, 12, 9.

The schedule using this list on four processors would be Processor 1: 36, 25, 19, 18, 17, 16, 13, 9; Processor 2: 32, 26, 25, 18, 16, 15, 12; Processor 3: 31, 27, 24, 18, 16, 15, 12, 12; Processor 4: 30, 30, 19, 18, 18, 15, 14.

The schedule using this list on five processors would be Processor 1: 36, 24, 18, 16, 14, 12; Processor 2: 32, 25, 18, 18, 15, 12; Processor 3: 31, 25, 19, 18, 15, 9; Processor 4: 30, 27, 18, 17, 15, 13; Processor 5: 30, 26, 19, 16, 16, 12.

(c) The five-processor decreasing-time schedule is optimal (time 120), but the four-processor decreasing-time schedule is not. One can see this, since when the task of length 17 scheduled on processor 1 and the task of length 18 on processor 3 are interchanged, the completion time is reduced to 154 from 155 for the four-processor, decreasing-time schedule.

(d) As a bin-packing problem, each bin will have a capacity of 60. Using the decreasing list, we obtain the following packings: (First-fit decreasing): Bin 1: 36, 24; Bin 2: 32, 27; Bin 3: 31, 26; Bin 4: 30, 30; Bin 5: 25, 25, 9; Bin 6: 19, 19, 18; Bin 7: 18, 18, 18; Bin 8: 18, 17, 16; Bin 9: 16, 16, 15, 13; Bin 10: 15, 15, 14, 12; Bin 11: 12, 12.

(e) NFD uses 13 bins. Bin 1: 36; Bin 2: 32; Bin 3: 31; Bin 4: 30, 30; Bin 5: 27, 26; Bin 6: 25, 25; Bin 7: 24, 19; Bin 8: 19, 18, 18; Bin 9: 18, 18, 18; Bin 10: 17, 16, 16; Bin 11: 16, 15, 15; Bin 12: 15, 14, 13, 12; Bin 13: 12, 12, 9. WFD uses 11 bins. Bin 1: 36, 24; Bin 2: 32, 26; Bin 3: 31, 27; Bin 4: 30, 30; Bin 5: 25, 25; Bin 6: 19, 19, 18; Bin 7: 18, 18, 18; Bin 8: 18, 17, 16; Bin 9: 16, 16, 15, 13; Bin 10: 15, 15, 14, 12; Bin 11: 12, 12, 9.

(f) An optimal packing with 10 bins exists. Bin 1: 36, 24; Bin 2: 32, 16, 12; Bin 3: 31, 17, 12; Bin 4: 30, 30; Bin 5: 27, 18, 15; Bin 6: 26, 18, 16; Bin 7: 25, 19, 16; Bin 8: 25, 19, 15; Bin 9: 18, 18, 12, 9; Bin 10: 18, 15, 14, 13.

58. First-fit

(a) First-fit, capacity 9:
Bin 1: 8, 1;
Bin 2: 5, 3, 1;
Bin 3: 4, 3, 2;
Bin 4: 7, 2;
Bin 5: 8, 1;
Bin 6: 8, 1;
Bin 7: 6, 3;
Bin 8: 5, 2, 2;
Bin 9: 3, 5, 1;
Bin 10: 4, 2, 3;
Bin 11: 6, 2;
Bin 12: 5, 4;
Bin 13: 6, 2;
Bin 14: 7;
Bin 15: 7;
Bin 16: 8;
Bin 17: 6;
Bin 18: 5, 4;
Bin 19: 6;
Bin 20: 4, 5;
Bin 21: 7;
Bin 22: 4.

(b) First-fit, capacity 10
Bin 1: 8, 2;
Bin 2: 5, 3, 1, 1;
Bin 3: 4, 3, 3;
Bin 4: 7, 2, 1;
Bin 5: 8, 2;
Bin 6: 8, 2;
Bin 7: 6, 3, 1;
Bin 8: 5, 5;
Bin 9: 4, 2, 3, 1;
Bin 10: 6, 4;
Bin 11: 5, 2, 2;
Bin 12: 6, 4;
Bin 13: 7;
Bin 14: 7;
Bin 15: 8;
Bin 16: 6, 4;
Bin 17: 5, 5;
Bin 18: 6, 4;
Bin 19: 7.

Continued on next page

(c) First-fit, capacity 11
Bin 1: 8, 3;
Bin 2: 5, 4, 2;
Bin 3: 3, 6, 1, 1;
Bin 4: 7, 3, 1;
Bin 5: 8, 2, 1;
Bin 6: 8, 2, 1;
Bin 7: 5, 3, 2;
Bin 8: 5, 4, 2;
Bin 9: 6, 3, 2;
Bin 10: 5, 4, 2;
Bin 11: 6, 5;
Bin 12: 7, 4;
Bin 13: 7, 4;
Bin 14: 8;
Bin 15: 6, 5;
Bin 16: 6, 4;
Bin 17: 7.

(d) First-fit, capacity 12
Bin 1: 8, 3, 1;
Bin 2: 5, 4, 3;
Bin 3: 7, 5;
Bin 4: 8, 3, 1;
Bin 5: 8, 2, 2;
Bin 6: 6, 2, 1, 3;
Bin 7: 5, 2, 4, 1;
Bin 8: 2, 6, 3, 1;
Bin 9: 5, 4, 2;
Bin 10: 6, 6;
Bin 11: 7, 5;
Bin 12: 7, 4;
Bin 13: 8, 4;
Bin 14: 6, 5;
Bin 15: 7, 2;
Bin 16: 4.

58. continued

First-fit decreasing

The decreasing list is: 8, 8, 8, 8, 7, 7, 7, 7, 6, 6, 6, 6, 6, 5, 5, 5, 5, 5, 5, 4, 4, 4, 4, 4, 4, 3, 3, 3, 3, 3, 2, 2, 2, 2, 2, 2, 2, 1, 1, 1, 1, 1.

(a) First-fit decreasing, capacity 9
Bin 1: 8, 1;
Bin 2: 8, 1;
Bin 3: 8, 1:
Bin 4: 8, 1;
Bin 5: 7, 2;
Bin 6: 7, 2;
Bin 7: 7, 2;
Bin 8: 7, 2;
Bin 9: 6, 3;
Bin 10: 6, 3;
Bin 11: 6, 3;
Bin 12: 6, 3;
Bin 13: 6, 3;
Bin 14: 5, 4;
Bin 15: 5, 4;
Bin 16: 5, 4;
Bin 17: 5, 4;
Bin 18: 5, 4;
Bin 19: 5, 4;
Bin 20: 2, 2, 2, 1.

(b) First-fit decreasing, capacity 10
Bin 1: 8, 2;
Bin 2: 8, 2;
Bin 3: 8, 2;
Bin 4: 8, 2;
Bin 5: 7, 3;
Bin 6: 7, 3;
Bin 7: 7, 3;
Bin 8: 7, 3;
Bin 9: 6, 4;
Bin 10: 6, 4;
Bin 11: 6, 4;
Bin 12: 6, 4;
Bin 13: 6, 4;
Bin 14: 5, 5;
Bin 15: 5, 5;
Bin 16: 5, 5;
Bin 17: 4, 3, 2, 1;
Bin 18: 2, 2, 1, 1, 1, 1.

(c) First-fit decreasing, capacity 11
Bin 1: 8, 3;
Bin 2: 8, 3;
Bin 3: 8, 3;
Bin 4: 8, 3;
Bin 5: 7, 4;
Bin 6: 7, 4;
Bin 7: 7, 4;
Bin 8: 7, 4;
Bin 9: 6, 5;
Bin 10: 6, 5;
Bin 11: 6, 5;
Bin 12: 6, 5;
Bin 13: 6, 5;
Bin 14: 5, 4, 2;
Bin 15: 4, 3, 2, 2;
Bin 16: 2, 2, 2, 2, 1, 1, 1;
Bin 17: 1, 1.

(d) First-fit decreasing, capacity 12
Bin 1: 8, 4;
Bin 2: 8, 4;
Bin 3: 8, 4;
Bin 4: 8, 4;
Bin 5: 7, 5;
Bin 6: 7, 5;
Bin 7: 7, 5;
Bin 8: 7, 5;
Bin 9: 6, 6;
Bin 10: 6, 6;
Bin 11: 6, 5, 1;
Bin 12: 5, 4, 3;
Bin 13: 4, 3, 3, 2;
Bin 14: 3, 3, 2, 2, 2;
Bin 15: 2, 2, 2, 1, 1, 1, 1.

All of the solutions using the decreasing list, with capacity 9–12 are optimal.

59. (a) Packing boxes of the same height into crates: packing want ads into a newspaper page.

(b) We assume, without loss of generality, $p \geq q$. One heuristic, similar to first-fit, orders the rectangles $p \times q$ as in a dictionary (i.e., $p \times q$ listed prior to $r \times s$ if $p > r$ or $p = r$ and $q \geq s$). It then puts the rectangles in place in layers in a first-fit manner; that is, do not put a rectangle into a second layer until all positions on the first layer are filled. However, extra room in the first layer is "wasted."

(c) The problem of packing rectangles of width 1 in an $m \times 1$ rectangle is a special case of the two-dimensional problem, equivalent to the bin-packing problem we have discussed.

(d) Two 1×10 rectangles cannot be packed into a 5×4 rectangle, even though there would be an area of 20 in this rectangle.

60. One example would be scheduling advertisements in station breaks, where the breaks can have two different lengths. Variants of the algorithms for bins of one size can be developed; however, one must develop a way to address the fact that an item might fit in a bin of larger size, but not of smaller size.

61. There is an example of a bin-packing problem for which a given list takes a certain number of bins, and when an item is deleted from the list, more bins are required. In this example, the deleted item is not first in the list.

62. The answer to Exercise 54 mentions one such "paradoxical" situation. Another possible paradox would be: a given set of weights requires a certain number of bins; when these weights are all decreased by, say, 1, more bins are required. Still another possible paradox would be: a given set of weights requires, say, N bins of capacity W. When the capacity is increased to, say, $W + 1$, more than N bins are required. Problems such as these make nice research projects.

63. (a) Graphs (a), (d), (e), and (f) can be colored with three colors, but graphs (b) and (c) cannot.

(b) Graphs (a), (b), (d), (e), and (f) can be colored with four colors, but graph (c) cannot.

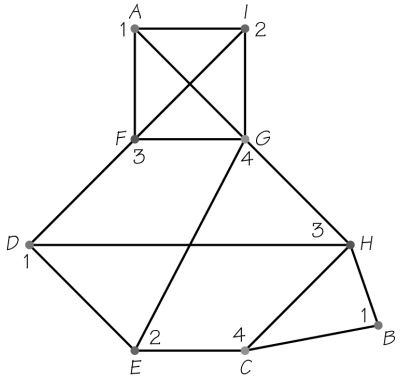
(c) The chromatic number for graphs (a) through (f) are, respectively, 3, 4, 5, 2, 3, and 2.

64. (a) The only graphs shown whose vertices can be colored with two colors are (b), (c), and (e).

(b) The only graphs shown whose vertices can be colored with three colors are (b), (c), (e), and (h).

(c) The chromatic number is 4 for (a), 2 for (b), 2 for (c), 4 for (d), 2 for (e), 6 for (f), 5 for (g), and 3 for (h).

65. (a) Construct the graph shown below:



The vertices of this graph can be colored with no fewer than four colors (1, 2, 3, 4 are used to denote the colors in the figure). Hence, four tanks can be used to display the fish.

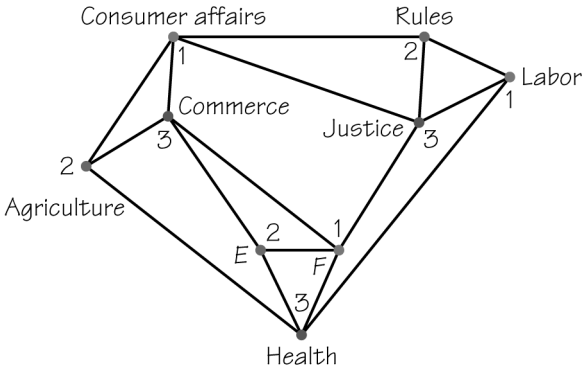
(b) The coloring in (a) shows that one can display two types of fish in three of the tanks, and three types of fish in one tank. Since 4 does not divide 9, one cannot do better.

66. (a) 5

(b) Yes, because 5 divides 10.

(c) Equalizing numbers in each enclosure may make observing this enclosure more interesting for the public. If the enclosures are not equal in size, equalization may mean that a small enclosure gets too many animals.

67. (a) The graph for this situation is shown below. The vertices can be labeled with the colors 1, 2, 3 as shown.



(b) Since the vertices can be colored with three colors (and no fewer), the minimum number of time slots for scheduling the committees is three.

(c) The committees can be scheduled in three rooms during each time slot. This might be significant if there were only three rooms that had microphone systems.

68. (a) The chromatic number is 4. There is a coloring using each color twice and one color three times. There is also a coloring using one color 4 times, two colors 2 times, and one color 1 time.

(b) There is a coloring with 5 colors, where each color is used twice.

(c) There is a coloring with 3 colors, where each color is used six times.

(d) There is a coloring with 4 colors, but the colors cannot be used equally often.

69. (a) Three time slots. To solve this problem, draw a graph by joining the vertices representing two committees if there is no x in the row and column of the table for these two committees.

(b) It is possible to three-color this graph so that each of the three colors is used three times. This means that one needs three rooms to arrange the scheduling of the nine committees.

70. (a) Three frequencies.

(b) Each frequency can be used twice.

71. Start at any vertex of the tree and label this vertex with color 1; color any vertex attached by an edge to this vertex with color 2. Continue to color the vertices in the tree in this manner, alternating the use of colors. If some vertex were attached to both a vertex colored 1 and another vertex colored 2, at some stage this would imply the graph had a circuit (of odd length), which is not possible, since trees have no circuits of any length.

72. Yes. The graph with the n vertices each joined to the others requires n colors.

73. The edge-coloring numbers for graphs (a) through (f) of Exercise 63 are, respectively, 6, 8, 6, 3, 3, and 4. The minimum edge-coloring number of any graph is either the maximal valence of any vertex in the graph or one more than the maximal valence. (This fact was discovered by the Russian mathematician Vizing.)

74. There are applications of edge colorings of graphs to designing timetables. Also, because one can find out information about vertex colorings for certain graphs by studying edge colorings, one can extend the applicability of the edge-coloring concept.

75. (a) Graph (a) four colors; graph (b) two colors; graph (c) four colors; graph (d) four colors; graph (e) two colors; graph (f) three colors.

(b) Coloring the maps of countries in an atlas would be one application of face colorings of graphs.

76. The minimum number of play groups is 3 because the chromatic number for this graph is 3. Since there are 7 children, the only number of play groups which could be set up which is conflict free would be if each child formed his/her own play group! With 3 play groups one can form two groups of size 2 and one group of size 3.

77. 3

78. Removing a single edge will not change the chromatic number of this graph.

Word Search Solution