Chapter 3
Planning and Scheduling

For All Practical Purposes: Effective Teaching

- Courses such as the one you are teaching can assist students in becoming lifetime learners by encouraging them to be active participants in the learning experience. In order for students to maximize their understanding of your lecture, encourage them to read the section or sections that are to be covered the night before.
- In a course such as this, it is helpful to begin the class with an overview of the topics to be covered along with a brief discussion as to their applications in the real world.

Chapter Briefing

In this chapter, you will firstly examine situations where you will schedule tasks. This will be done in different ways depending on a priority list, order requirements, and the number of machines that will process the tasks. *Order-requirement digraphs* are helpful in visually determining which tasks have precedence while considering the priority list. Laying the tasks out on machines (which looks like a brick wall) is a very helpful technique to determining the *completion time*. *Critical paths* will be revisited in order to determine optimal schedules. Unfortunately, no computationally efficient method is known that will always produce an optimal schedule.

The second type of problem is the *bin-packing problem*. In this application one needs to determine the least number of containers (of the same fixed capacity) needed to pack a given list of items of various sizes. A natural condition is that each item cannot have a capacity greater than that of the bins. Three algorithms will be discussed. They are *next-fit*, *first-fit*, and *worst-fit*.

A final type of problem situation applies to scenarios in which a schedule or a configuration must be produced that avoids conflicts. Naturally one will seek an optimal solution to minimize the number of meetings or other physical objects such as cages.

Vertex coloring of a graph can be of use in solving such problems. 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.

Being well prepared for class discussion with examples and a clear understanding of the different methods presented in this chapter are essential in order to help students focus on the main topics presented. In order to facilitate your preparation, the **Chapter Topics to the Point** has been broken down into **Scheduling Tasks**, **Bin Packing**, and **Coloring**. Examples with solutions for these topics that do not appear in the text nor study guide are included in the *Teaching Guide*. You should feel free to use these examples in class, if needed.

The last section of this chapter of *The Teaching Guide for the First-Time Instructor* is **Solutions to Student Study Guide ✎ Questions**. These are the complete solutions to the eight questions included in the *Student Study Guide*. Students only have the answers to these questions, not the solutions.

Chapter Topics to the Point

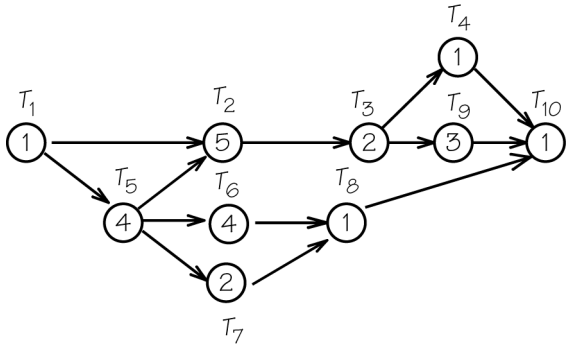
Scheduling Tasks

The **machine-scheduling problem** is to decide how a collection of tasks can be handled by a certain number of **processors** as quickly as possible. We have to respect both order requirements among the tasks and a priority list. The **list-processing algorithm** chooses a task for an available processor by running through the priority list in order, until it finds the **ready task**. The list-processing algorithm considers the following.

- Times of the tasks
- Number of processors
- Order-requirement digraph
- Ordering of the tasks on the list

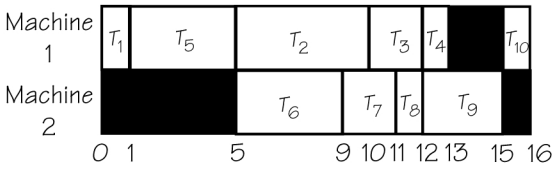
Example

Schedule the tasks in the digraph on two processors with priority list $T_1, T_2, T_3, T_4, T_5, T_6, T_7, T_8, T_9, T_{10}$. What is the completion time?



Solution

T_1 is first. No task can be started until T_1 is completed. T_2 cannot be scheduled until T_5 is completed. So T_5 is scheduled on machine 1 and machine 2 remains idle. Since T_2, T_3, and T_4 have priority next, they are scheduled on machine 1. T_6 must wait for completion of T_5. So, T_6 is scheduled on machine 2 since it is ready. T_7, T_8, and T_9 follow on machine 2. Since machine 1 is ready, the final task, T_{10}, is scheduled on machine 1. The completion time is 16. The following is the schedule.



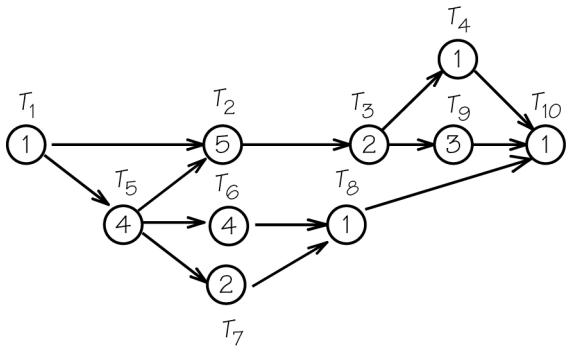
Teaching Tip

Point out to students that the algorithm for assigning tasks to machines assigns the lowest-numbered free processor the first task on the priority list that is ready.

Different priority lists and a different number of machines can lead to different schedules and completion times.

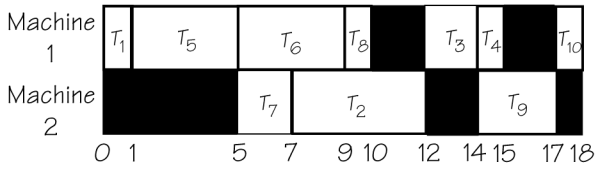
Example

Schedule the tasks in the digraph on two processors with priority list T_1, T_4, T_5, T_6, T_7, T_8, T_9, T_{10}, T_2, T_3. What is the completion time?



Solution

T_1 is first. No task can be started until T_1 is completed. T_4 cannot be scheduled until yet. So T_5 is scheduled on machine 1 and machine 2 remains idle. T_6 and T_7 must wait for completion of T_5. So, T_6 is scheduled on machine 1 and T_7 is scheduled on machine 2. T_8 is scheduled on machine 1 following and T_2 is scheduled on machine 2 since the necessary tasks have been completed. T_3, will go on machine 1 followed by T_4. T_9 is scheduled on machine 2 after the completion of T_3. Since T_{10} requires the completion of T_9 it is scheduled on machine 1. The completion time is 18. The following is the schedule.

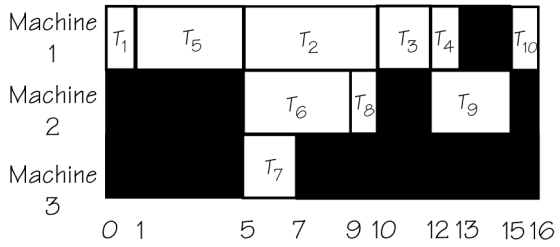


Example

Using the same digraph, schedule the tasks on three processors with priority list T_1, T_2, T_3, T_4, T_5, T_6, T_7, T_8, T_9, T_{10}. What is the completion time?

Solution

T_1 is first. T_2 cannot be scheduled until T_5 is completed. So T_5 is scheduled on machine 1, and machines 2 and 3 remain idle. Since T_2, T_3, and T_4 have priority next, they are scheduled on machine 1. T_6 and T_7 must wait for completion of T_5. So, T_6 and T_7 are scheduled on machine 2 and 3, respectively. T_8 is scheduled on machine 2, and T_9 is also scheduled on machine 2 after T_3 has finished. Since machine 1 is ready, the final task, T_{10}, is scheduled on machine 1 after T_4 is completed. The completion time is 16. The following is the schedule.



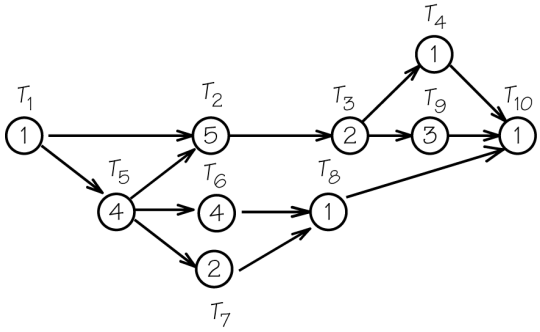
☝Teaching Tip

Relay to students that changes such as a different priority list of number of machines may actually increase the completion time from a previous schedule. This should motivate them to think about a way to know that we have an optimal schedule.

A schedule is **optimal** if it has the earliest possible total completion time. For example, a critical path in the order-requirement digraph may determine the earliest completion time. The **critical-path scheduling** algorithm schedules first the tasks in a critical path. If there is a tie, choose the task with the lower number.

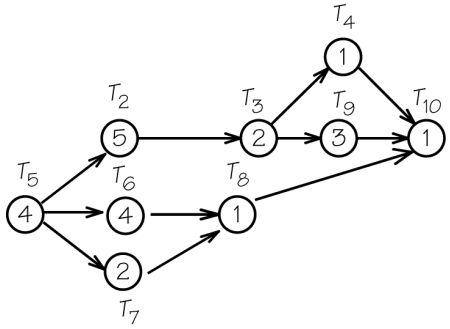
Example

Determine the earliest completion time for the order-requirement digraph. Use critical-path scheduling to construct a priority list for the tasks.

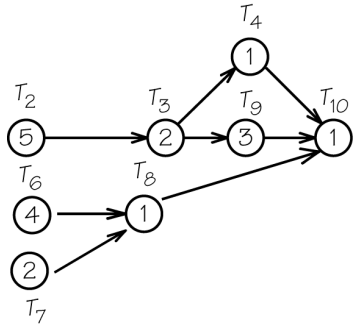


Solution

$T_1, T_5, T_2, T_3, T_9, T_{10},$ is the longest path in the digraph, and no scheduling can be completed in less time than the length of this path. The length of this path is 16. T_1 is at the head of a critical path. When you remove T_1 and its arrows, T_5 is the head of the remaining critical path $T_5, T_2, T_3, T_9, T_{10}$.

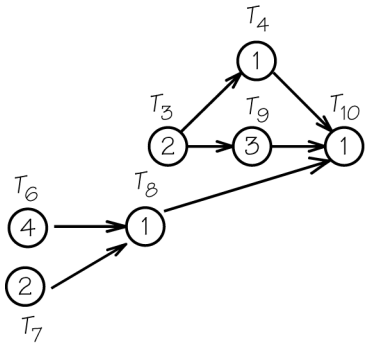


Removing T_5 and its arrows makes T_2 the head of T_2, T_3, T_9, T_{10}.

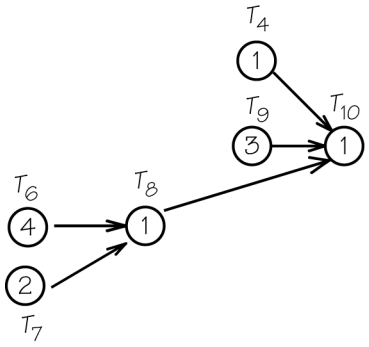


Continued on next page

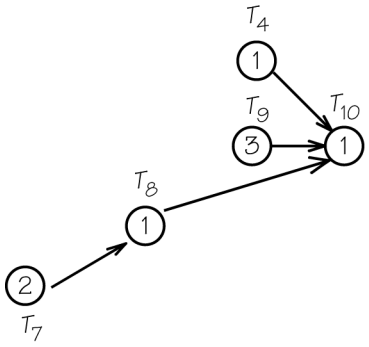
Removing T_2 and its arrows makes T_3 the head of T_3, T_9, T_{10}. (Although T_6, T_8, T_{10} has the same length, we next place on the priority list the task with the lowest number.)



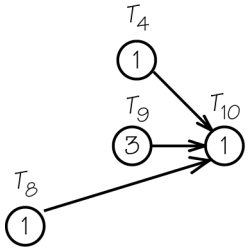
Removing T_3 and its arrows makes T_6 the head of T_6, T_8, T_{10}.



Removing T_6 and its arrows makes T_7 the head of T_7, T_8, T_{10}. (Although T_9, T_{10} has the same length, we next place on the priority list the task with the lowest number.)

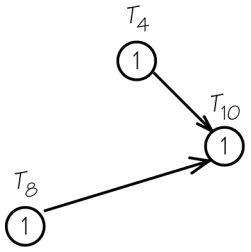


Removing T_7 and its arrows makes T_9 the head of T_9, T_{10}.

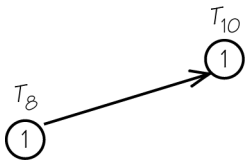


Continued on next page

Removing T_9 and its arrows makes T_4 the head of T_4, T_{10}. (Although T_8, T_{10} has the same length, we next place on the priority list the task with the lowest number.)



Removing T_4, T_8 is the head of T_8, T_{10}.

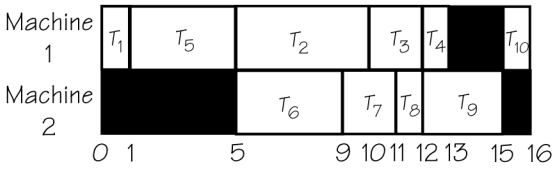


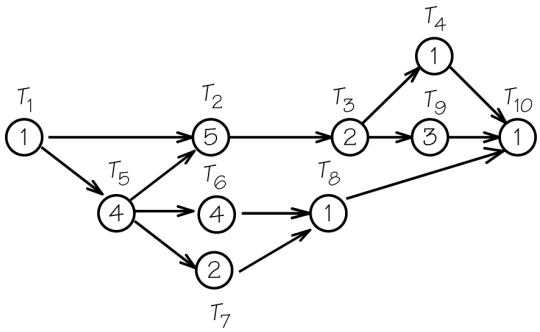
Thus the priority list is $T_1, T_5, T_2, T_3, T_6, T_7, T_9, T_4, T_8, T_{10}$.

Although we can make a schedule based on this priority list, a completion time of 16 (which is optimal) was already determined with the initial example.

Teaching Tip

Without actually constructing another schedule, determine if there are any differences compared to the schedule that we arrived to with the priority list $T_1, T_2, T_3, T_4, T_5, T_6, T_7, T_8, T_9, T_{10}$ which is as follows. (T_4, T_8, and T_9, are shuffled around with the same completion time)

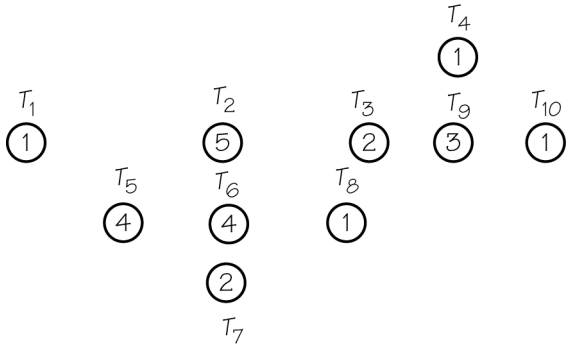




When a set of tasks are **independent** (can be done in any order), we have a variety of available algorithms to choose a priority list leading to close-to-optimal scheduling. Some algorithms perform well in the **average-case** (examining the mean of the completion times of all priority list, compared to optimal completion time), but poorly in the **worst-case** (examining how far from optimal any given completion time is). The **decreasing-time-list algorithm** schedules the longest tasks earliest. By erasing the arrows in the digraph, we obtain a set of independent tasks.

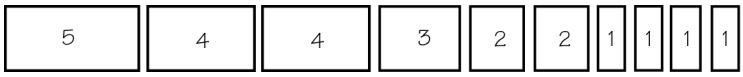
Example

Construct a decreasing-time priority list. Use this list to schedule the tasks on two processors and determine the completion time.



Solution

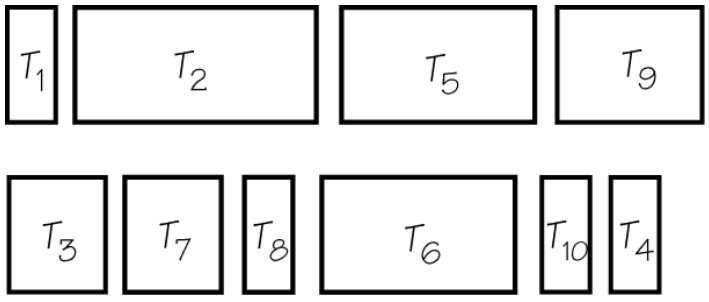
Since we treat these as independent tasks, the task reference can be removed (keeping only the time).



The schedule leads to a completion time of 12.

☝Teaching Tip

Cutting out blocks scaled to size can be used on an overhead and can assist in making the schedule. The following can be cut out (or duplicated). If you choose to use the *Guide* examples in class, you may find students enjoy manipulating these pieces. The tasks can be marked through and replaced with the time as the above example demonstrates.



Bin Packing

With the **bin-packing problem**, we consider scheduling tasks within a fixed time limit, using as few processors as possible. This is like fitting boxes into bins of a certain size – but it is used in a variety of real-world applications. We have a variety of heuristic algorithms available to do the packing well if not optimally. Three important algorithms are as follows.

- **Next fit (NF)**: Use the remaining space in the bin. If there is not enough space, open a new bin. You cannot go back to previous bins.
- **First fit (FF)**: Use the first bin that has room for your object. If no bin has room, open a new bin.
- **Worst fit (WF)**: Examine all the open bins and look for the one that has the most space in it and use that bin. If no bin has enough room, open a new bin.

Example

Pack boxes of sizes 5, 7, 2, 6, 5, 1, 3, 4, 2, 3, 6, 3 into bins of capacity 10. How many bins are required

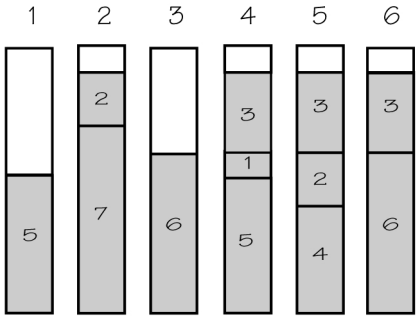
a) using the next-fit algorithm?

b) using the first-fit algorithm?

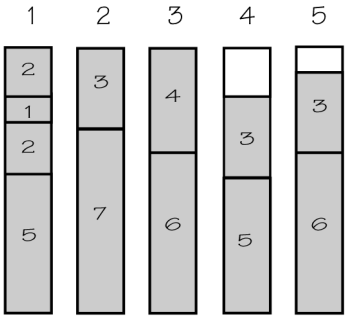
c) using the worst-fit algorithm?

Solution

a) There are six bins required.

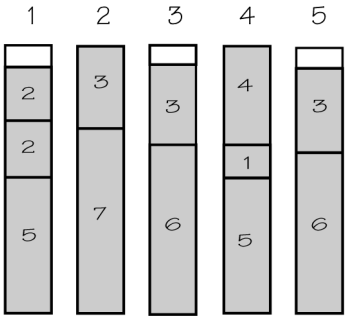


b) There are five bins required.



Continued on next page

c) There are five bins required.



☝Teaching Tip

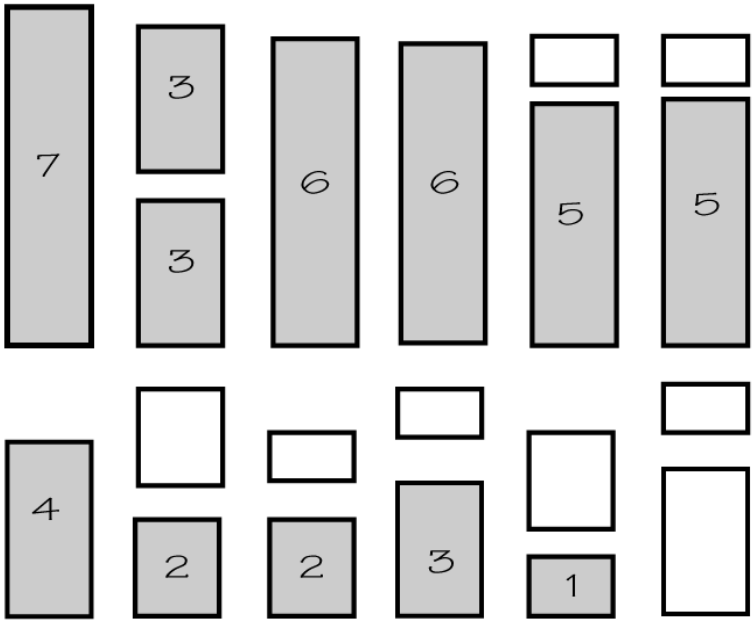
Convey to students that in the worst-fit (WF) algorithm we pack an item into a bin with the most room available. Although this algorithm can lead to the same number of bins as other algorithms, the items may be packed in a different order.

☝Teaching Tip

WF is like FF in that it permits returning to earlier bins. However, in FF you always start back at the first bin and sequentially search for a bin that will accommodate this weight, while in WF you calculate the unused space in each available bin and select the bin with the maximum room.

☝Teaching Tip

Cutting out blocks scaled to size can be used on an overhead to pack bins. The following can be cut out (or duplicated). The white blocks are for the empty space in a bin.



☝Teaching Tip

Determining the total amount of space available (as a rectangle) and the amount of space needing to be packed can be used to check if the packing makes sense. One can examine the sum of the empty space and the packed space to see if it matches the total space available. This would help students to determine if they missed a box or included extra boxes.

Each of these algorithms can be combined with **decreasing-time** heuristics, leading to the three algorithms **next-fit decreasing (NFD), first-fit decreasing (FFD),** and **worst-fit decreasing (WFD**).

Example

Pack boxes of sizes 5, 7, 2, 6, 5, 1, 3, 4, 2, 3, 6, 3 into bins of capacity 10. How many bins are required

a) using the next-fit decreasing algorithm?

b) using the first-fit decreasing algorithm?

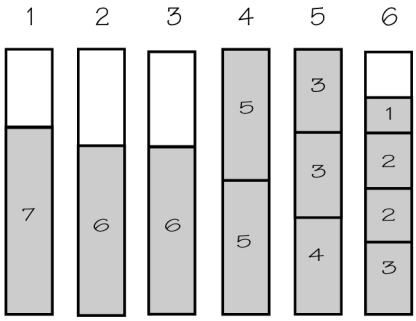
c) using the worst-fit decreasing algorithm?

Solution

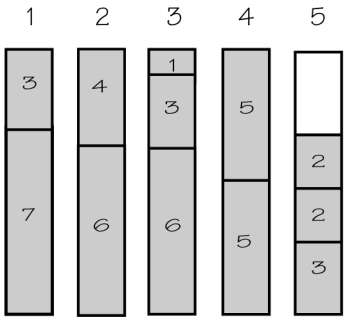
First, rearrange the boxes in decreasing order of size.

7, 6, 6, 5, 5, 4, 4, 3, 3, 3, 2, 2, 1

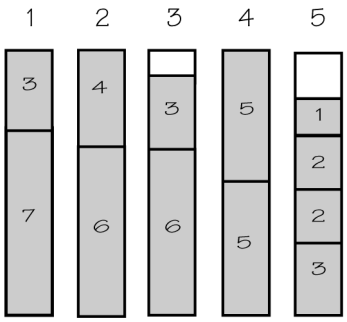
a) There are six bins required.



b) There are five bins required.



c) There are five bins required.

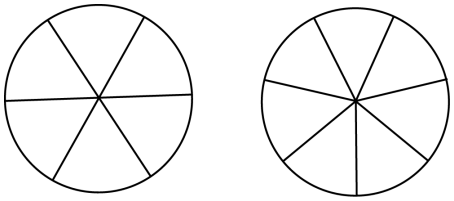


Coloring

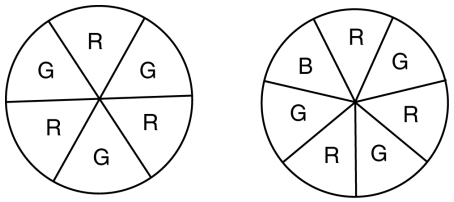
If we represent items to be scheduled (classes, interviews, etc) as vertices in a graph, then a **vertex coloring** of the graph can be used to assign resources, such as times or rooms, to the items in a conflict-free manner. The **chromatic number** of the graph determines the minimum amount of the resource that must be made available for a conflict-free schedule.

Example

Discuss the coloring of the following graphs. Can you make any generalizations regarding the chromatic number of such figures?



Solution

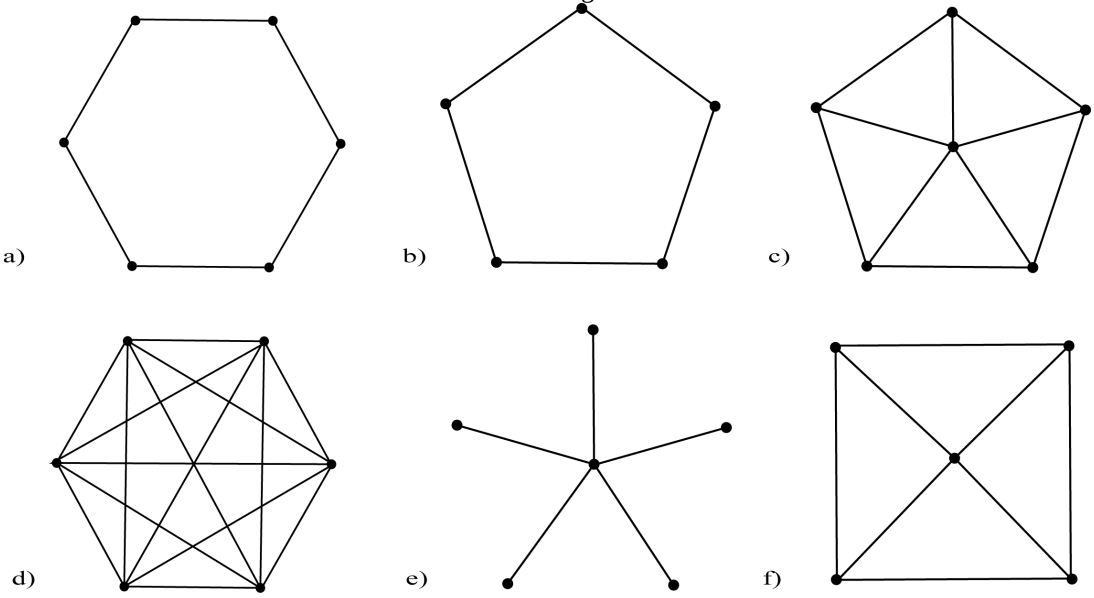


2 colors 3 colors

In general, any pie chart with an even number of sectors can be colored with 2 colors and with an odd number of sectors (greater than one) requires 3.

Example

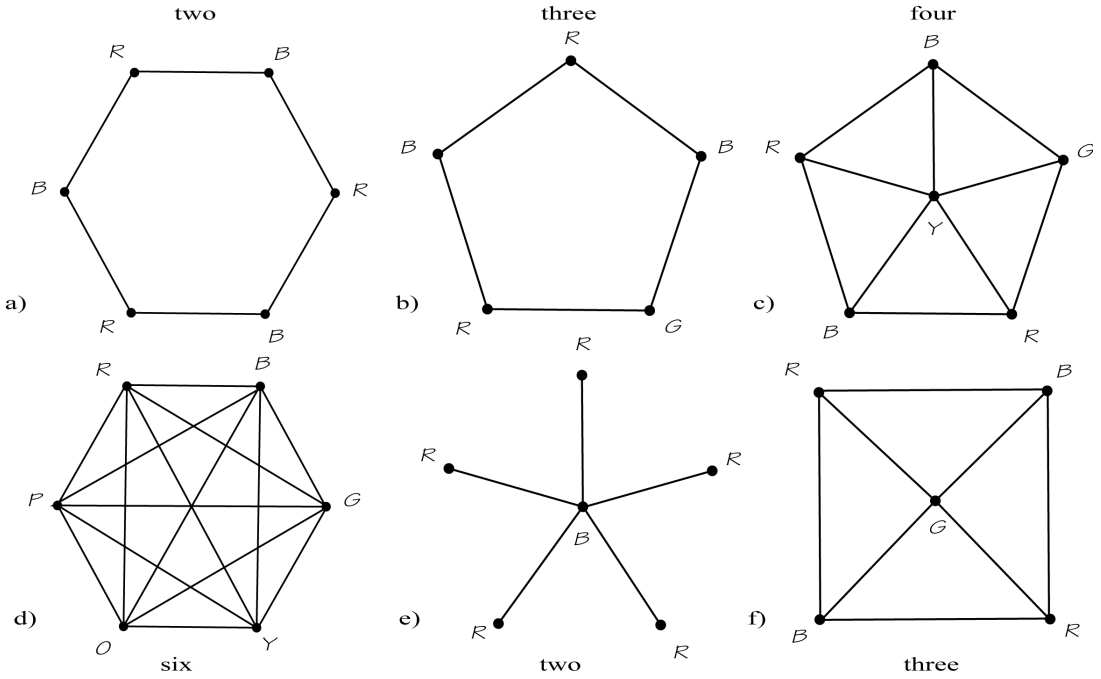
What is the chromatic number for each of the following?



Continued on next page

Solution

The following are suggested colorings. The colors are *R* (red), *B* (blue), *G* (green) *Y* (yellow), *O* (orange), and *P* (purple).



Example

Mr. Bill Its wanted to take a group of students on a field trip. He knows from experience that certain students should not be placed in the same car as others. An × indicates the children that should not travel together.

	Adam	Edwin	Scott	Sami	Dean	Doug	Dan
Adam		×			×		
Edwin	×		×	×			×
Scott		×		×			×
Sami		×	×		×		
Dean	×			×			
Doug							
Dan		×	×				

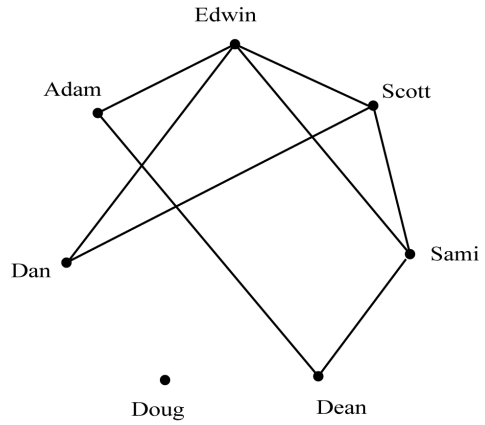
a) If a car can hold no more than 3 children, what is the minimum number of cars needed for the field trip?

b) If a car can hold no more than 2 children, what is the minimum number of cars needed for the field trip?

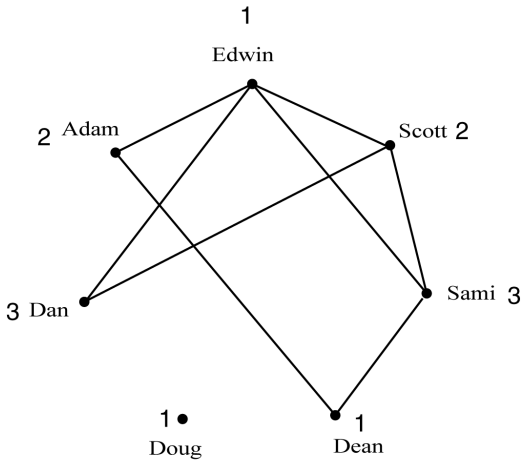
Continued on next page

Solution

A graph can be used to draw the conflicts between students. When two students should not be placed in the same car, an edge is drawn between the vertices that connect the two.



Now we could use 7 cars, but that is not optimal. This graph can be colored with 3 colors.



a) If a car can hold up to 3 students, then 3 cars is the minimum number of cars

b) If a car only has room for 2 children, then four is the minimum.

Notice in both cases, the minimum number of cars is consistent with the physical constraint of seating 7 children. However the arrangement of the children needs to be considered due to conflicts.

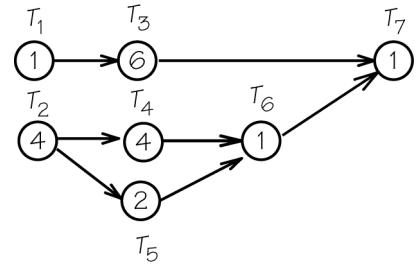
☝Teaching Tip

In this last example, you can promote class discussion by asking about the design in the conflict table. Why must there be symmetry about the diagonal? Why are there no $\times$ marks on the diagonal?

Solutions to Student Study Guide ✎ Questions

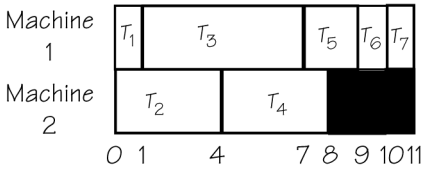
Question 1

Schedule the tasks in the digraph on two processors with priority list $T_1, T_2, T_3, T_4, T_5, T_6, T_7$. What is the completion time?



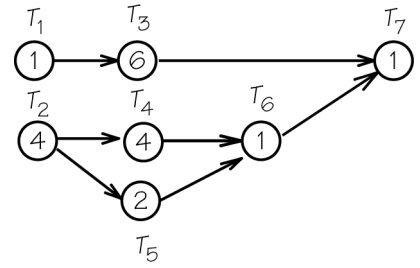
Solution

T_1 and T_2 are first, then T_3. T_4 and T_5 must wait for completion of T_2. T_6 must wait for T_4 and T_5. Also, T_7 must wait for T_3 and T_6. The completion time is 11. The following is the schedule.



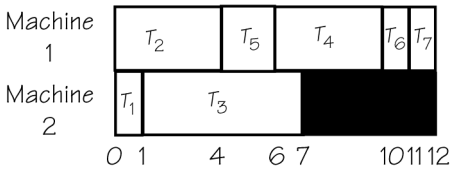
Question 2

Schedule the tasks in the digraph on two processors with priority list $T_7, T_6, T_5, T_4, T_3, T_2, T_1$. What is the completion time?



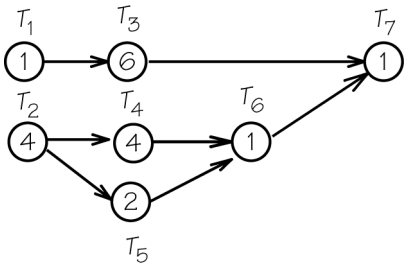
Solution

T_2 is the highest priority ready task and gets scheduled first, with T_1 next. After T_1 is done, no task is ready except for T_3, so T_3 is scheduled. When T_2 is done, T_5 is scheduled followed by T_4, T_6 and T_7. The completion time is 12. The following is the schedule.



Question 3

What is the length of the critical path?

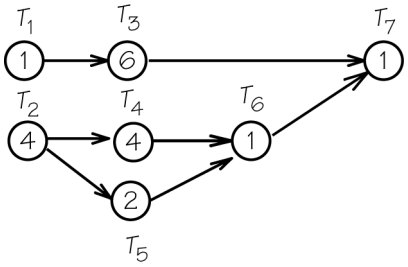


Solution

The critical path is T_2, T_4, T_6, T_7. The length is 10.

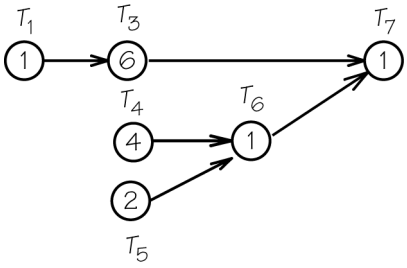
Question 4

Use critical-path scheduling to construct a priority list for the tasks in this digraph. How much total time are the machines not processing tasks?

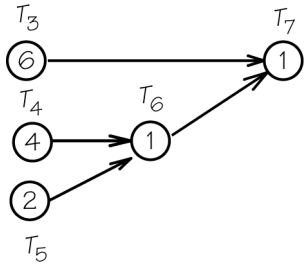


Solution

The critical path is T_2, T_4, T_6. T_7. T_2 is at the head of this critical path. When you remove T_2 and its arrows, T_1 is the head of the remaining critical path T_1, T_3, T_7.



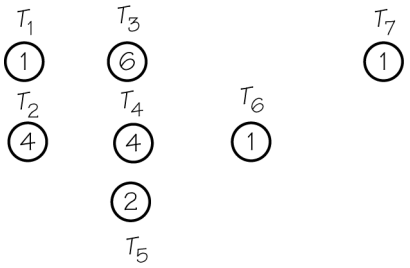
Removing T_1 and its arrow makes T_3 the head of T_3, T_7. Next, T_3 is removed as well.



T_4 is the head of the remaining critical path T_4, T_6, T_7. Removing T_4 we are left with T_5, T_6, T_7. Thus the priority list is T_2, T_1, T_3, T_4, T_5, T_6, T_7. The schedule will be like Question 1 with the machines switched. The total time that the machines will not be processing tasks will be 3.

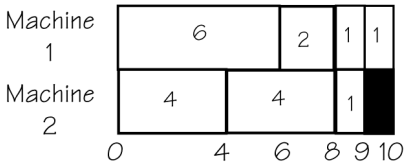
Question 5

Construct a decreasing-time priority list. What is the completion time?



Solution

We need to schedule tasks with times 6, 4, 4, 2, 1, 1, 1.



The completion time is 10.

Question 6

Consider packing boxes sized 2, 6, 2, 6, 3, 4, 1, 4, 2 into bins of capacity 7. How many bins are required if we pack using

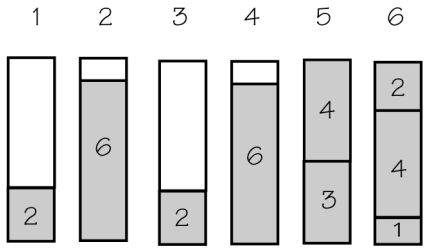
a) next-fit algorithm?

b) first-fit algorithm?

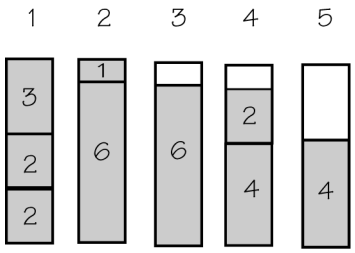
c) worst-fit algorithm?

Solution

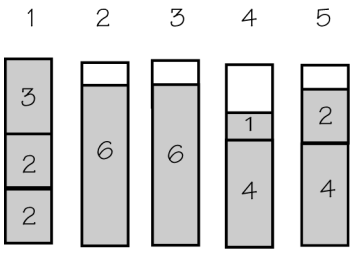
a) Six bins are required.



b) Five bins are required.



c) Five bins are required.



Question 7

Consider packing boxes sized 2, 6, 2, 6, 3, 4, 1, 4, 2 into bins of capacity 7. How many bins are required if we pack using

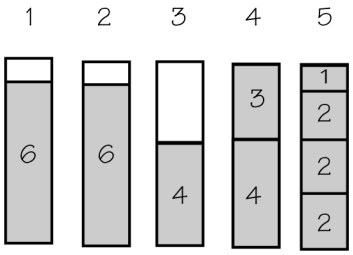
a) next-fit decreasing algorithm?

b) first-fit decreasing algorithm?

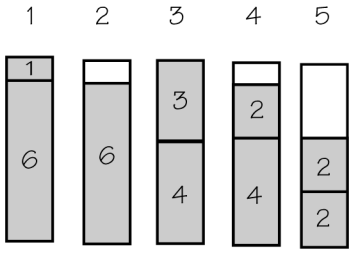
Answer

You must first put the sized in descending order: 6, 6, 4, 4, 3, 2, 2, 2, 1.

a) Five bins are required.

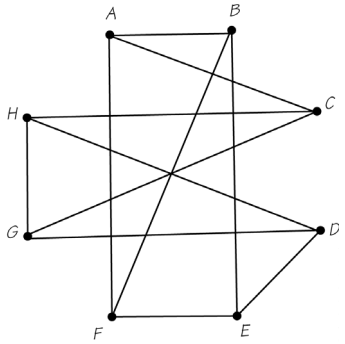


b) Five bins are required.



Question 8

Suppose the following graph shows conflicts between animals A – H. If an edge connects two animals then they cannot be put in the same cage. Determine a suitable arrangement with a minimum number of cages. What is the minimum number of cages?



Solution

There are many correct answers as far as arrangements. One such arrangement is as follows.

- Cage 1: B, C, and D
- Cage 2: A, E, and G
- Cage 3: F and H

There is a minimum of 3 cages required.