CHAPTER 3: PLANNING AND SCHEDULING

3.1 Scheduling Tasks

A ***processor*** is a person or machine or robot that works on a task.

To solve a machine scheduling problem (MSP) typically the tasks are scheduled to minimize completion time.

Take into account things such as the importance of a task and if one task needs to be completed before another task.

Assumptions and Rules

- If a processor starts working on a task, the work will continue until that task is complete.
- No processor stays idle if there is a task to be done.
- The MSP has an associated order-requirement weighted digraph.
- The tasks are arranged in a ***priority list*** that is independent of the digraph.

Possible Goals

- Minimize the completion time for a given number of processors.
- Minimizing processor idle time.
- Minimize the number of processors needed to complete the job in a specified time.

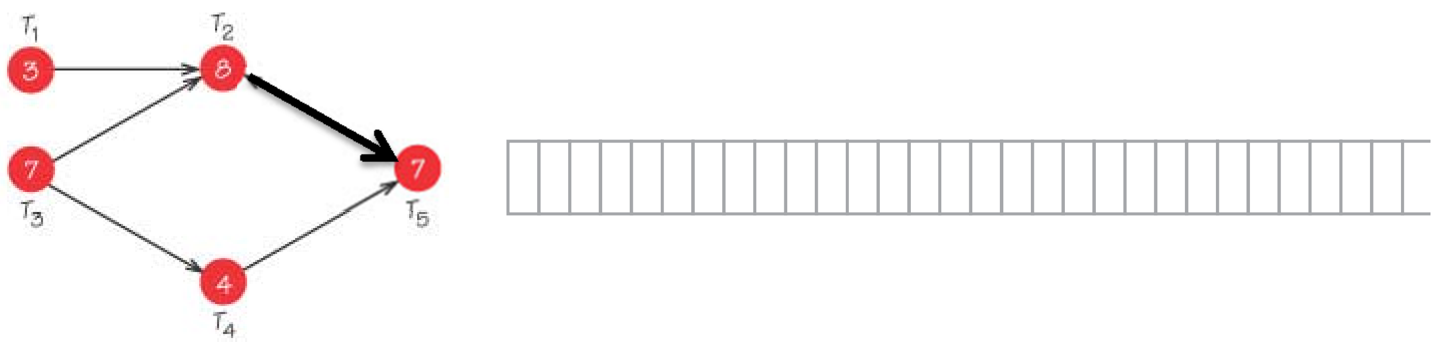
A task is considered ***ready*** if all its predecessors in the digraph have been completed.

List Processing Algorithm

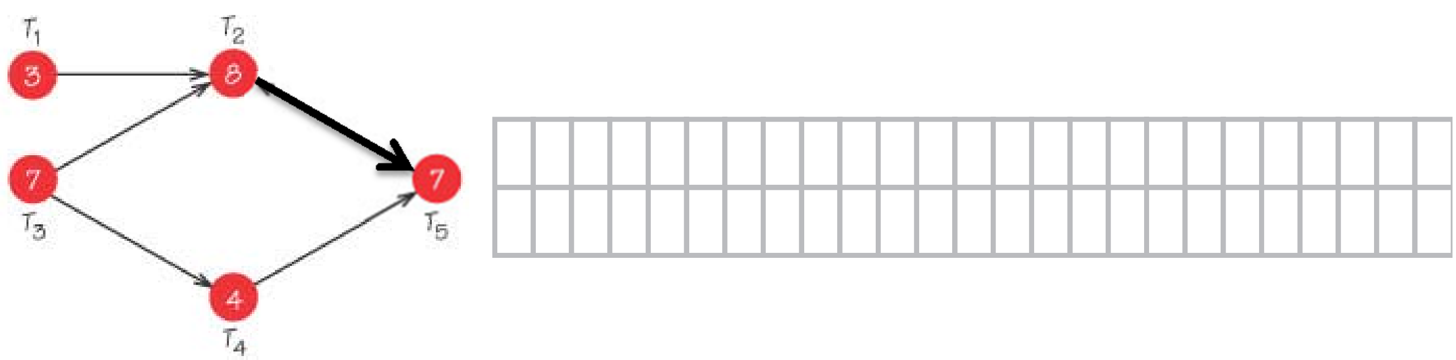
1. **Assignment of Processors:** The lowest numbered idle processor is assigned to the highest priority ready task until either all processors are assigned or all ready tasks are being worked on.
2. **Status Check:** When a processor completes a task, that processor becomes idle. Check for ready tasks and tasks not yet completed and determine which of the following applies
   a. If there are ready tasks, repeat step 1.
   b. If there are no ready tasks but not every task has been completed, the idle processor remains idle until more tasks are completed.
   c. If all tasks are completed, the job is done.

EXAMPLE

(a) What is the completion time for the job shown in the digraph below using the priority list T_3, T_2, T_1, T_4, T_5 and 1 processor?



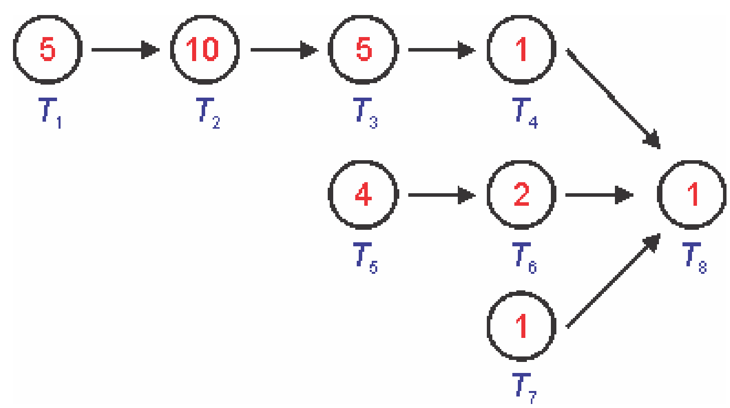
(b) What is the completion time for the job shown in the digraph below using the priority list T_3, T_2, T_1, T_4, T_5 and 2 processors?



<u>Making Fajitas</u>
Tortillas
Chicken
1 onion
Seasoning
Tomatoes
Cilantro

Slice the onion and chicken into strips. Cook chicken 10 minutes then add onions and cook for 5 more minutes. Add seasoning. Chop tomatoes and mix with cilantro. Warm tortillas. Serve

T_1: slice chicken & onions – 5 min
T_2: cook chicken – 10 min
T_3: cook onions – 5 min
T_4: add seasoning – 1 min
T_5: chop tomato – 4 min
T_6: mix tomato and cilantro – 2 min
T_7: warm tortillas – 1 min
T_8: serve – 1 min



Priority list: T_6, T_5, T_1, T_2, T_3, T_4, T_8, T_7

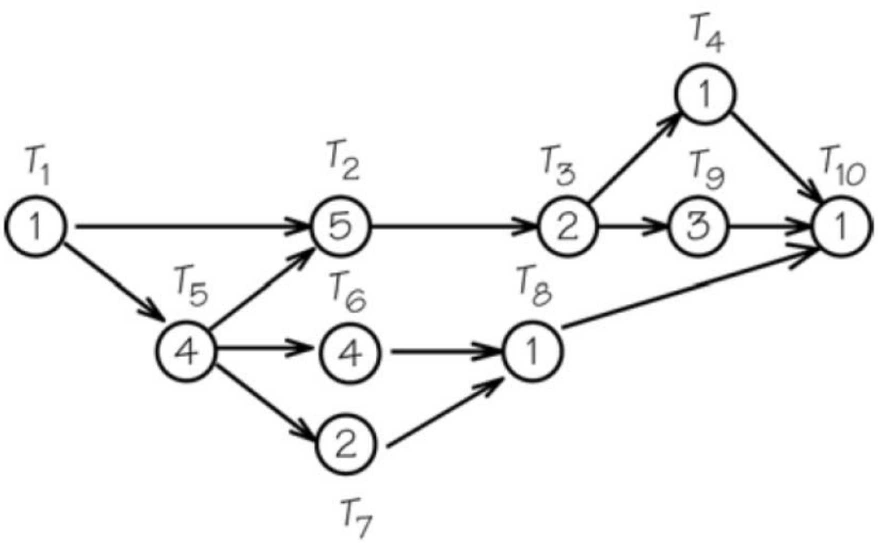
Apply the list processing algorithm using 1 and 2 processors to make fajitas. Would 3 processors make the meal any faster?

An ***optimal*** schedule is a schedule assigning processors to tasks in such a way that it results in the shortest possible finishing time for that project with that number of processors.

__EXAMPLE__

Apply the list processing algorithm to the digraph below using 2 processors and the priority list

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



Using the same digraph with 3 processors and priority list

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

Making Lunches

T_1: find lunch boxes – 5 minutes
T_2: clean lunch boxes – 3 minutes
T_3: make sandwiches – 7 minutes
T_4: wrap sandwiches – 2 minutes
T_5: wash apples – 2 minutes
T_6: wrap chips – 3 minutes
T_7: get drink – 1 minute
T_8: pack lunch boxes – 4 minute

Priority list: T_7, T_6, T_3, T_2, T_1, T_4, T_5, T_8

Use this list to schedule two processors to complete the job. Is it optimal? If not, use 3 processors.

A ***decreasing time priority list*** is created by listing all the tasks from the longest to the shortest completion times. In the case of a tie, the lowest numbered task is done first.

EXAMPLE

Create a decreasing time priority list for making lunches.

3.2 Critical Path Schedules

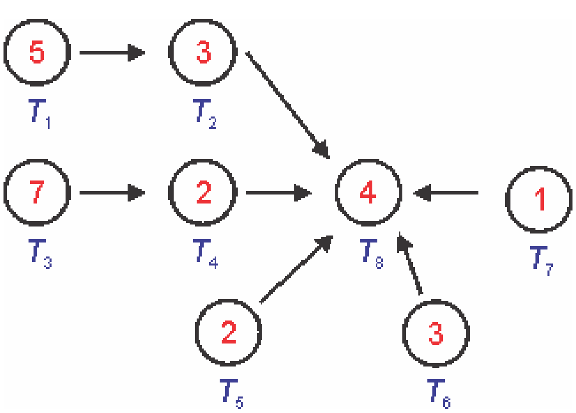
How can we create a priority list that has a reasonable chance of being optimal or near optimal?

Creating a Priority List for Critical Path Scheduling

1. Find a task that heads a critical (longest) path in the order-requirement digraph. If there is a tie, chose the lowest task number.
2. Place the task found in step 1 next in the priority list.
3. Remove the task found in step 1 from the digraph. Remove all edges attached to the removed task to form a new diagraph.
4. If all tasks have been removed, the list is completed. If tasks remain, return to step 1.

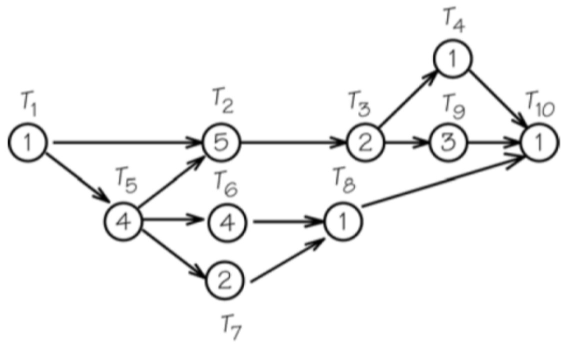
EXAMPLE

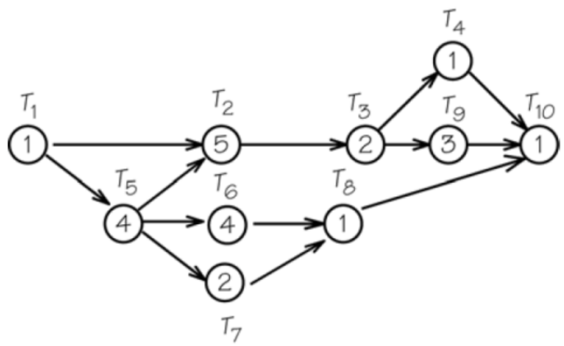
a) Create a priority list for critical path scheduling for making lunches.
b) Use this list to schedule two processors to complete the job. Is it optimal? If not, use 3 or more processors.

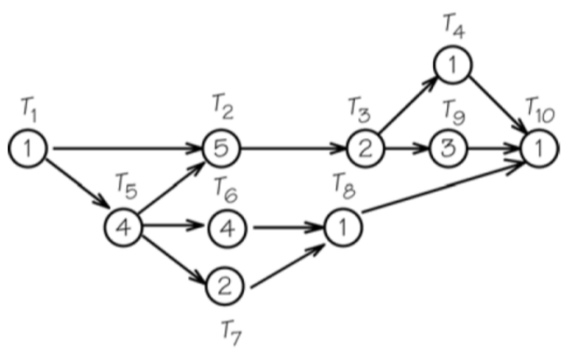


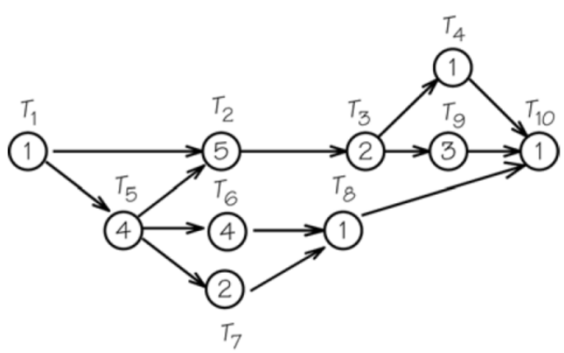
<u>EXAMPLE</u>

Create a priority list for critical path scheduling for the digraph below









3.3 Independent Tasks

When the tasks are independent, they can be done in any order. There are different algorithms that can be used to schedule independent.

EXAMPLE

Schedule 16 different skits to be into three sessions. The times for the skits are given in a decreasing time list below.

10, 9, 9, 9, 9, 9, 8, 7, 5, 4, 3, 3, 2, 1, 1, 1

Is this optimal? Why or why not?

EXAMPLE

What is the minimum time to complete 12 independent tasks on four processors when the sum of all the times of the 12 tasks is 60 minutes?

EXAMPLE

What are some applications for scheduling independent events?

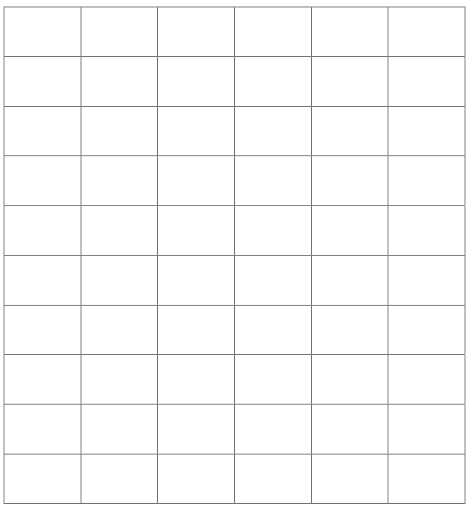
3.4 Bin Packing

EXAMPLE

You have boxes available that hold at most 10 pounds and you have items that weigh 5, 7, 2, 6, 5, 1, 3, 4, 2, 3, 6, 3 pounds. Pack these items using the following heuristic methods.

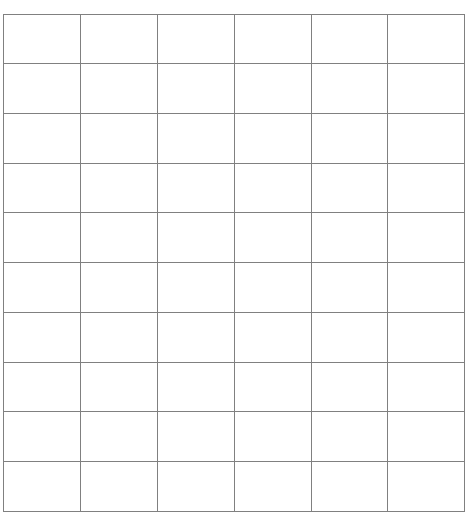
Next*-fit Algorithm (NF):** Put items into the open bin until the ***next item will not fit. Close the bin and open a new bin for the next item.

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



First*-fit Algorithm (FF):** Put items into the ***first already open bin that has space for it. If no open bin has space, open a new bin.

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



<u>Worst</u>*-fit Algorithm (WF):** Put items into an already open bin that has the ***<u>most</u> space for it. If no open bin has space, open a new bin.

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

<u>Best</u>*-fit Algorithm (BF):** Put items into an already open bin that has the ***<u>least</u> space for it. If no open bin has space, open a new bin.

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

Next-fit Decreasing Algorithm (NFD): Arrange the items from largest to smallest. Then put items into the open bin until the next item will not fit. Close the bin and open a new bin for the next item.

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

First-fit Decreasing Algorithm (FFD): Arrange the items from largest to smallest. Then put items into the first already open bin that has space for it. If no open bin has space, open a new bin.

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

Worst-fit <u>Decreasing</u> Algorithm (WFD): Arrange the items from largest to smallest. Then put items into an already open bin that has the most space for it. If no open bin has space, open a new bin.

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

Best-fit <u>Decreasing</u> Algorithm (BFD): Arrange the items from largest to smallest. Then put items into an already open bin that has the least space for it. If no open bin has space, open a new bin.

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

EXAMPLE

You have 20 boxes to put on shelves that have a fixed width of 12”. The widths of the boxes, in inches, are

3, 7, 2, 4, 4, 4, 5, 3, 9, 5, 8, 2, 7, 8, 4, 6, 4, 1, 10, 5

What is the optimal number of shelves?

(a) FF: 3, 7, 2, 4, 4, 4, 5, 3, 9, 5, 8, 2, 7, 8, 4, 6, 4, 1, 10, 5

(b) NF: 3, 7, 2, 4, 4, 4, 5, 3, 9, 5, 8, 2, 7, 8, 4, 6, 4, 1, 10, 5

(c) BF: 3, 7, 2, 4, 4, 4, 5, 3, 9, 5, 8, 2, 7, 8, 4, 6, 4, 1, 10, 5

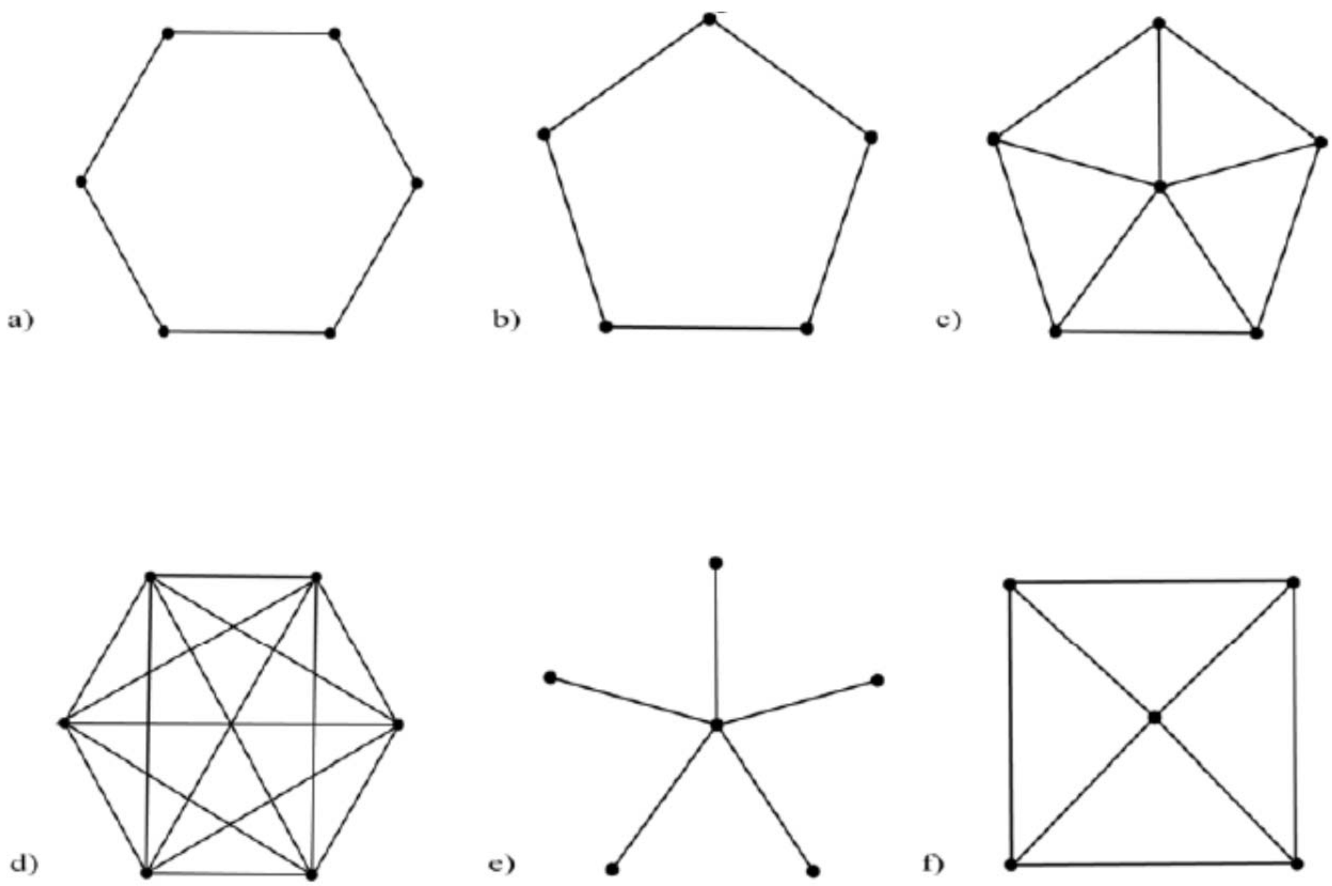
3.5 Resolving Conflict via Coloring

The ***vertex coloring*** problem for a graph requires assigning each vertex of the graph a color (or label) such that two vertices joined by an edge are assigned different colors.

The ***chromatic number*** of a graph is the minimum number of colors needed to label the vertices of the graph so that no two vertices joined by an edge have the same color.

EXAMPLE

Find the chromatic number for each of the graphs below:



EXAMPLE

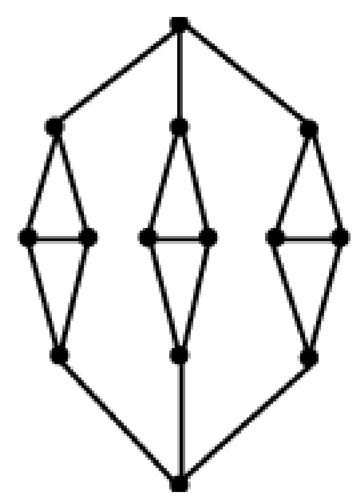
Animals will be put in enclosures in a wildlife park. The table below shows an X to indicate which animals cannot be placed with certain other animals. What is the fewest number of enclosures needed? How many animals will be in each enclosure?

	A	L	Z	G	H	E	R
Antelope		X			X		
Lion	X		X	X			X
Zebra		X		X			X
Gazelle		X	X		X		
Hyena	X			X			
Elephant							
Rhino		X	X				

The ***edge-coloring number*** of a graph is the minimum number of colors needed to color the edges of the graph so that edges that share a common vertex get different colors.

EXAMPLE

There are 14 teams in a league and they wish to schedule games to be played several weeks into the season. The vertices below represent the teams and two vertices are connected if the two teams have not played yet. What is the fewest number of days that are needed to play the remaining games?



When a graph has been drawn on a piece of paper so that the edges meet only at vertices, the graph divides the paper up into regions called ***faces***. The faces include the one called the ***infinite face***, which surrounds the whole graph. The ***face-coloring number*** of the graph is the minimum number of colors needed to color the faces of the graph so that two faces that share an edge receive different colors. Note that if two faces meet only at a vertex, then they can be colored the same color.

EXAMPLE

Find the fewest number of colors need to color the graphs below such that no two edges have the same color (map coloring).