## Chapter 8: The Mathematics of Scheduling

<table>
<thead>
<tr>
<th>Pd 2 &amp; 6</th>
<th>Pd 4 &amp; 7</th>
<th>Lesson</th>
<th>Homework</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tues 11/10 B</td>
<td>Wed 11/11 C</td>
<td>8.1, 8.2 The Basic Elements of Scheduling, Directed Graphs (Digraphs)</td>
<td>HW 8.1/8.2 TBA</td>
</tr>
<tr>
<td>Thurs 11/12 A</td>
<td>Fri 11/13 B</td>
<td>8.3 Priority List Scheduling</td>
<td>Worksheet 8.3 – Priority Lists &amp; Scheduling</td>
</tr>
<tr>
<td>Fri 11/13 B</td>
<td>Mon 11/16 C</td>
<td>8.5/8.6 Critical Path/Critical Path Algorithm</td>
<td>Worksheet 8.5/8.6 – Critical Path Algorithm</td>
</tr>
<tr>
<td>Tues 11/17 A</td>
<td>Wed 11/18 B</td>
<td>8.7 Independent Tasks, Relative Error</td>
<td>HW 8.7 Pg 310-313 #23-26, 52, 54, 56 ** Use Scheduling grids or graph paper for #52-56! **</td>
</tr>
<tr>
<td>Wed 11/18 B</td>
<td>Thurs 11/19 C</td>
<td>HW 8.7 DUE Review Day Chapter 8</td>
<td>Review Sheet – Not for a grade</td>
</tr>
<tr>
<td>Mon 11/20 A</td>
<td>Test Ch 8</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
8.2 Directed Graphs (Digraphs)

Finite Math Graph Memories:

Chapter 5: Graph Model

Chapter 6: Complete Graph $K_n$
Each of $N$ vertices is connected to every other vertex in the graph.

Chapter 7: Trees
A connected graph with no circuits.

Chapter 8: DIGRAPHS
A graph with direction associated with the edges.

Euler Path/Circuit: Covers each edge exactly once. Vertices may be visited multiple times.

Hamilton Path/Circuit: Covers each vertex exactly once. Many edges are left untraveled.

Minimum Spanning Tree: The spanning subgraph of a network with the least weight.
Steiner Tree: The shortest possible network, if it exists. (Steiner tree forms three 120 degree angles at an interior junction point.)

SIMILARITIES?

DIFFERENCES?
DIGRAPHS
A graph in which the edges have a DIRECTION associated with them is called a directed graph or “digraph” for short. Digraphs are used to represent precedence relationships in scheduling problems.

ARC
An edge with direction associated with it.

Order matters!

Arc XY

Arc YX

PATH
A sequence of arcs that can only be traveled in the direction indicated and no arc appears more than once.

Example:

CYCLE
A path that starts and ends at the same vertex is a cycle in the digraph.

Example:

INCIDENT TO/FROM
If arc AB is an arc in a digraph, then we say:

vertex A is __________________ vertex B (think: __________________)

vertex B is __________________ vertex A (think: __________________)

ADJACENT ARCS
Arc #1 is adjacent to Arc #2 if one Arc #1 BEGINS where Arc #2 ENDS. *** IN OTHER WORDS... An arc is adjacent to your arc if it’s LAST letter is your arcs FIRST letter. What would come right before it?****

Example: CD is adjacent to _______.

CD is NOT adjacent to _______.

Example. Draw the digraph with vertex set \( V = \{ G, H, O, S, T \} \) and arc set \( A = \{ GH, GO, OH, ST, TG, TH \} \).
Example: Consider the digraph:

(a) find all vertices incident to B
(b) find all vertices incident from B
(c) find an arc adjacent to AB
(d) find an arc adjacent to ED
(e) find any path from A to D
(f) find any cycle in the digraph

OUTDEGREE/INDEGREE

out-degree of vertex A is the number of arcs that have A as a _____________
in-degree of vertex A is the number of arcs that have A as an _____________

Example: outdegree(C) = _____
outdegree (D) = _____
indegree (C) = _____
indegree (D) = _____

** VOCAB ACTIVITY**
How well do you know these new words?

Applications of Digraphs

Traffic Flow: Graph Models of cities where vertices represent intersections and arcs represent one-way streets. A two way street would have two arcs, one for each direction.

Telephone Traffic: Telephone companies use “call digraphs” to analyze calls to and from numbers in their network.

Tournaments: Teams are vertices. Outcomes of games are arcs. AB “A defeated B”

Organization Charts: “Who’s the Big Cheese?” Vertices are people, arcs represent who is the immediate boss – a “chain of command” model.

Scheduling: Organizes the precedence relationships in a scheduling problem so we have a visual reminder of what tasks must be completed first. We call these “project digraphs”.
Example: Suppose the vertices in the following digraph represent basketball teams and that an arc from X to Y means “X defeated Y” in a game.

Which team(s) won the most games?

Which team(s) won the fewest games?

Did any teams play each other more than once?

Example: The course catalog at your college gives the following information regarding math classes.

A prerequisite for Calculus III is Calculus II. A prerequisite for Calculus II is Calculus I. Linear Algebra can only be taken after both Calculus II and Fundamentals of Mathematical Proofs have been taken. A prerequisite for Fundamentals of Mathematical Proofs is Calculus I. In order to take Real Analysis I, a student must have completed Calculus III and Linear Algebra. A prerequisite for Real Analysis II is Real Analysis I.

Draw a digraph which models the relationships between the math classes.

House Building Example: See pages 280 & 281
8.1 Basic Elements of Scheduling:

**PROCESSOR:**
Whomever or whatever is working on a task
- Use $P_1, P_2, \ldots, P_N$, etc.

**TASK:**
A task is an ______________ unit of work that cannot be broken up into smaller units (always carried out by a single processor.)
- Use Capital Letters: $A, B, C,$ etc. or maybe PL (Plumbing) or EP (Exterior Painting)

A task that cannot be started yet: A task that can be started now:

A task that is presently being carried out: A task that is done:

**PROCESSING TIME:** The amount of time it takes a task to be completed by one processor
Assume “Robotic Behavior” or humans trained to be very standardized

Notation: $X(y) =$ It takes $y$ amount of time for one processor to complete task $X$

Example: $W(15) =$ It takes 15 minutes to Wash the windows

**PRECEDENCE RELATIONS:**
A situation where one or more tasks must be ______________ before another task ______________.

In our house example: what is an example of a precedence relationship?

**INDEPENDENT TASKS**
Tasks with no precedence relations between them.

Example:

Precedence Relation:
In this chapter, we will use PROJECT DIGRAPHS to help us OPTIMIZE SCHEDULING PROBLEMS.

**Project Digraph:**

Example: Look at the following PROJECT DIGRAPH

What are some precedence relationships?

Name a pair of independent tasks

Note: A digraph has no start and end. A project digraph has a start(0) and end (0). Start and End are “imaginary tasks” that just help you “see” the path through the tasks. -- Think of Start as “cutting the ribbon” and End as declaring the project complete.

**Scheduling Problem**

Example:

You wreck your car. There are 4 repairs. There are two mechanics. Exterior body work must be completed before painting and exterior work can begin (A → C)

(A) exterior body work: 4 hrs; (B) engine repairs: 5 hrs; (C) painting and exterior work: 7 hrs; (D) transmission repair: 3 hrs

Possible Schedules:

Seems like we could do better.

Due to the precedence relation A → C WE CANNOT DO ANY BETTER THAN Fin = 11 hrs. This mean 11hrs is our CRITICAL TIME FOR THIS PROJECT. Every project has an absolute minimum time.
Example: **Building a Dream Home on Mars**
You have earned a trip to Mars and will be living in a MHU (Martian Habitation Unit). The MHU must be assembled by special construction robots that can be rented by the hour. The precedence relationships in the chart must be followed. You have to tell the robots when to do what. How many should you rent? What is the quickest way to get the job done?

<table>
<thead>
<tr>
<th>Task</th>
<th>Label (P-time)</th>
<th>Precedent tasks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assemble pad</td>
<td>AP(7)</td>
<td></td>
</tr>
<tr>
<td>Assemble flooring</td>
<td>AF(5)</td>
<td></td>
</tr>
<tr>
<td>Assemble wall units</td>
<td>AW(6)</td>
<td></td>
</tr>
<tr>
<td>Assemble dome frame</td>
<td>AD(8)</td>
<td></td>
</tr>
<tr>
<td>Install floors</td>
<td>IF(5)</td>
<td>AP, AF</td>
</tr>
<tr>
<td>Install interior walls</td>
<td>IW(7)</td>
<td>IF, AW</td>
</tr>
<tr>
<td>Install dome frame</td>
<td>ID(5)</td>
<td>AD, IW</td>
</tr>
</tbody>
</table>

If we make a project digraph it is much easier to see which tasks are “ready” from the start and which tasks we will have to wait to start.

Which tasks could be started right away?

Example:
Draw a project digraph for a project consisting of the six tasks described by the following table:

<table>
<thead>
<tr>
<th>Task</th>
<th>Processing Time</th>
<th>Precedence Relation</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>3</td>
<td>A, B</td>
</tr>
<tr>
<td>C</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>1</td>
<td>C</td>
</tr>
<tr>
<td>E</td>
<td>4</td>
<td>D</td>
</tr>
<tr>
<td>F</td>
<td>8</td>
<td>B</td>
</tr>
</tbody>
</table>
8.3 Scheduling with Priority Lists

Nothing in the project digraph actually TELLS us how to schedule. We could RANDOMLY approach the schedule, just following the precedence relations, but it would be better to have a plan.

**PRIORITY LIST**

A list of all tasks PRIORITIZED in the __________ we __________ to execute them.

For example: Priority List = W, X, Y, Z

- If we have to make a choice between W and X, we choose ______
- If W is not READY, we would _______ over it and move to the next READY task.
- If there are no READY tasks, the processors would sit _________ until something changes.

**Example:**

**TRIAL 1:**
We are going to schedule a project with the project digraph shown at the right. We have two processors: P1 and P2. We have a priority list:

Priority List: A, B, C, D, E

NOTATION:

ready: A
in execution: A
complete: A
ineligible: A
TRIAL 2:
Two processors.
Priority List: E, D, C, B, A

TRIAL 3:
Three processors.
Priority List: E, D, C, B, A

So, how do you decide WHAT priority list to use?

In a project with N tasks, there are N! possible priority lists!

In our project above with 5 tasks, we would have to check ______________ lists to find an optimal answer! Just like Chapter 6, we are going to need to find some efficient algorithm that will give us an approximate answer that we can consider "good enough."
Example:
Schedule the following on two processors using the given priority list.

Priority List:

**C, B, E, D, A, F**

Example:
Schedule the following project on two processors using the given priority list.

Priority List:

**G, F, A, C, H, B, D, E**
Example:

Schedule the following project on three processors using the given priority list.

Priority List:

G, E, F, A, C, H, D, B

8.5 Critical Paths/ 8.6 The Critical Path Algorithm

Recall our "Car Wreck" example from the first day of class.

Example 8.1

You wreck your car. There are 4 repairs. There are two mechanics.

(A) exterior body work: 4 hrs;
(B) engine repairs: 5 hrs;
(C) painting and exterior work: 7 hrs;
(D) transmission repair: 3 hrs

Recall that we determined there was no way to schedule the project in less than 11 hrs.

The longest path from start to end in this graph is 11 hrs. This is called our critical path.
Notation:

\( X(6) = \text{processing time of 6} \)

-- “it takes 6 hours to complete task A.”

\( X[8] = \text{critical time of 8,} \)

-- “it takes 8 hours to reach the “END” once you start \( X. \)”

Example:

Find the critical time for each vertex.

A: \hspace{1cm} C: \hspace{1cm} E:

B: \hspace{1cm} D: \hspace{1cm} Start:

What is the critical path for the project?

The more precedence relationships you have, the more difficult it may be to determine the critical time of a vertex or of a project. You need to use the “Backflow” Algorithm.

**CRITICAL PATH**

For a given vertex \( X \) of a project digraph, the **critical path for \( X \)** is the path from \( X \) to END with longest processing time.

When we add the processing times of all the tasks along the critical path for a vertex \( X \), we get the **critical time for \( X \)**.

---

**THE BACKFLOW ALGORITHM**

Start from the END of the project.

Determine the critical time for a vertex.

The critical time for a vertex =

\[ \text{the processing time for the vertex} + \text{the LARGEST critical time of any vertex incident from that vertex.} \]

(see picture to left)

Example: The CRITICAL TIME for vertex C is:
Example:
Find the critical time for each vertex. Find the critical path for the entire project.

Critical Path:

Example:  Find the critical time for each vertex.  Find the critical path for the entire project.

Critical Path:

Critical Path Algorithm:  Another way of creating a priority list.
Step 1: Find the critical time of each vertex.
Step 2: List the vertices in order of decreasing CRITICAL TIME
Step 3: Schedule the project!
Example: Schedule the project on two processors using the critical path algorithm.

Critical Times:
A = _____ B = ______
C = _____ D = ______
E = ______ F = ______
G = ______ H = ______

What is the critical path?

Priority List:

Example: Schedule the project on two processors using the critical path algorithm.

Critical Times:
A = _____ B = ______
C = _____ D = ______
E = ______ F = ______
G = ______ H = ______

What is the critical path?

Priority List:
8.7 Scheduling Independent Tasks

In the special case where there are no precedence relationships, we have independent tasks.

This means that each task is READY from the start of the project.
Also, CRITICAL TIME = PROCESS TIME!

Example:
You and two friends are cooking a meal and there are nine dishes to prepare. The nine dishes can be prepared in any order, so we can consider them to be independent tasks. The tasks and the time to complete each (in minutes) are listed below.

A(70), B(90), C(100), D(70), E(80), F(20), G(20), H(80), I (10)

Trial 1: Schedule the tasks using the critical path algorithm (For independent tasks, critical time equals processing time)

Priority List:

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Trial 2: Find an OPTIMAL SCHEDULE using 3 processors.

What is the relative error of Trial 1?

Error = ( Fin – Opt) / Opt
Example:
Consider the seven independent tasks below and their times (in minutes):
A(60), B(30), C(40), D(30), E(50), F(30), G(40)

Trial 1:
Schedule on three processors using the critical path algorithm.
Priority List:

<table>
<thead>
<tr>
<th></th>
<th>50</th>
<th>100</th>
<th>150</th>
<th>200</th>
<th>250</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Trial 2:
Find an optimal schedule using 3 processors. What is the relative error of Trial 1?

<table>
<thead>
<tr>
<th></th>
<th>50</th>
<th>100</th>
<th>150</th>
<th>200</th>
<th>250</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Example:
Find an optimal schedule for six processors for the thirteen independent tasks with times:
3, 3, 4, 4, 5, 5, 5, 5, 6, 6, 7, 7, 12

<table>
<thead>
<tr>
<th></th>
<th>5</th>
<th>10</th>
<th>15</th>
<th>20</th>
<th>25</th>
<th>30</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Why doesn’t it make sense to schedule more than 6 copiers to this job?
Example:
A project consists of 11 tasks (A through K) with the following processing times, in hours:
A(10), B(7), C(11), D(8), E(9), F(5), G(3), H(6), I(4), J(7), K(5)

If a schedule with 3 processors has a completion time of 31 hours, what is the total idle time in the schedule?

Why can the completion time for 3 processors never be less than 25 hours?

Graham’s Bound

In 1969, American mathematician Ron Graham (who has a really interesting bio on page 306) proved that when scheduling independent tasks using the critical path algorithm, your answer always has a relative error of at most 33 1/3 %

The Graham bound for the relative error \( \varepsilon \) when scheduling a set of independent tasks with \( N \) processors is

\[
\varepsilon = \frac{N - 1}{3N} .
\]

<table>
<thead>
<tr>
<th>( N )</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>( \cdots )</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>( (N - 1)/3N )</td>
<td>1/6 ( \approx 16.66% )</td>
<td>2/9 ( \approx 22.22% )</td>
<td>3/12 ( = 25% )</td>
<td>4/15 ( \approx 26.66% )</td>
<td>5/18 ( \approx 27.78% )</td>
<td>( \cdots )</td>
<td>99/300 ( = 33% )</td>
</tr>
</tbody>
</table>