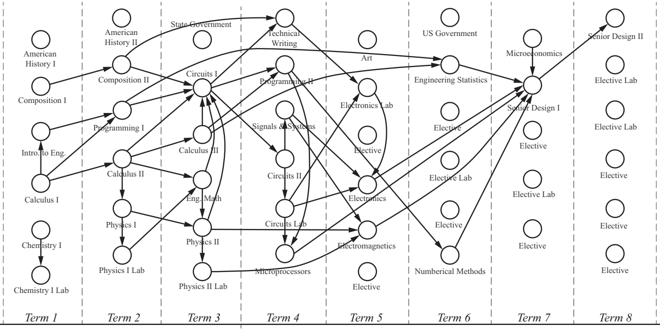
Characterizing the Complexity of Curricular Patterns in Engineering Programs

Gregory L. Heileman Ahmad Slim
Michael Hickman Chaouki T. Abdallah
{heileman, ahslim, mhickman, chaouki}@unm.edu

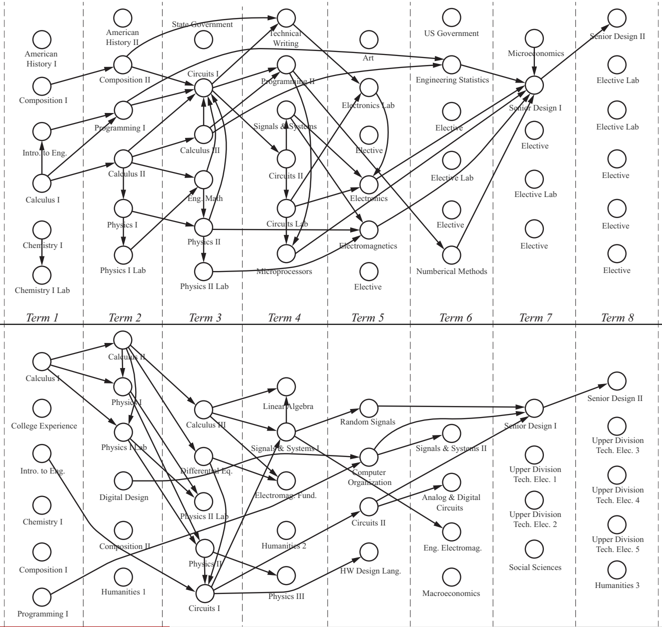
Department of Electrical & Computer Engineering
University of New Mexico

June 26, 2017

Two EE Programs

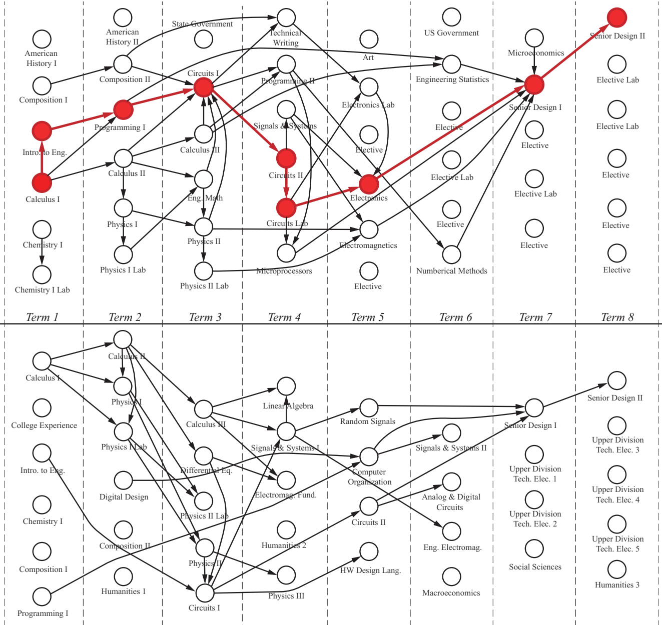


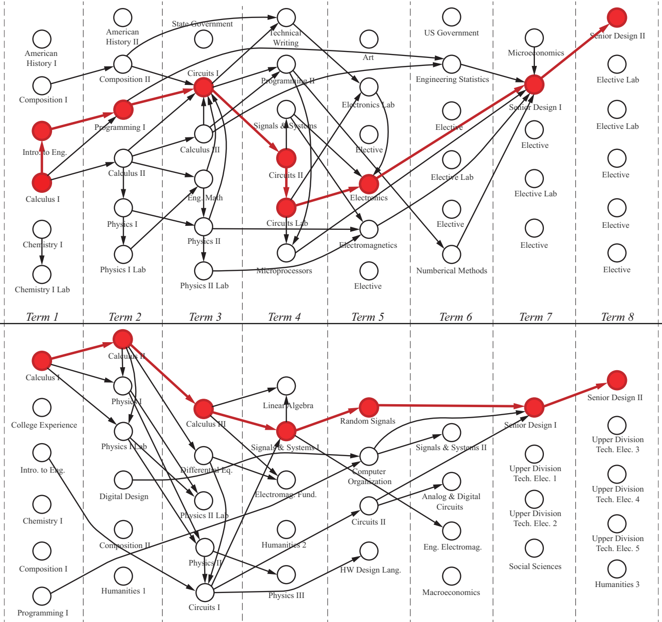
Two EE Programs



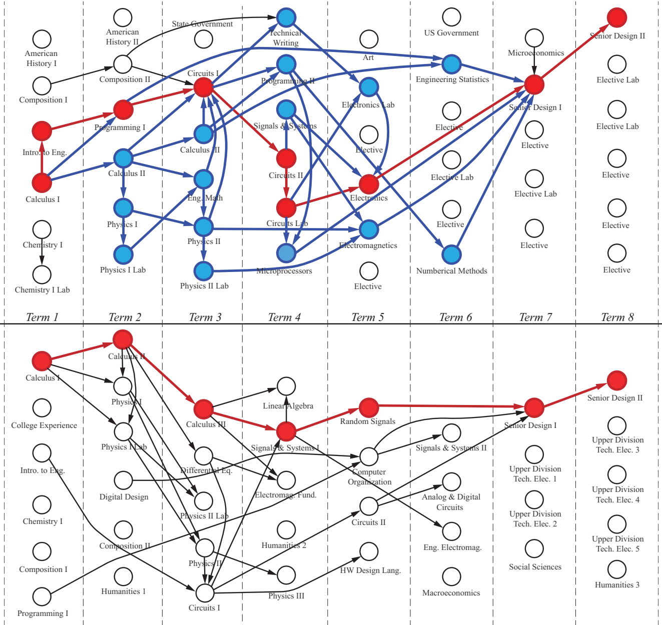
Two EE Programs

THE UNIVERSITY of NEW MEXICO

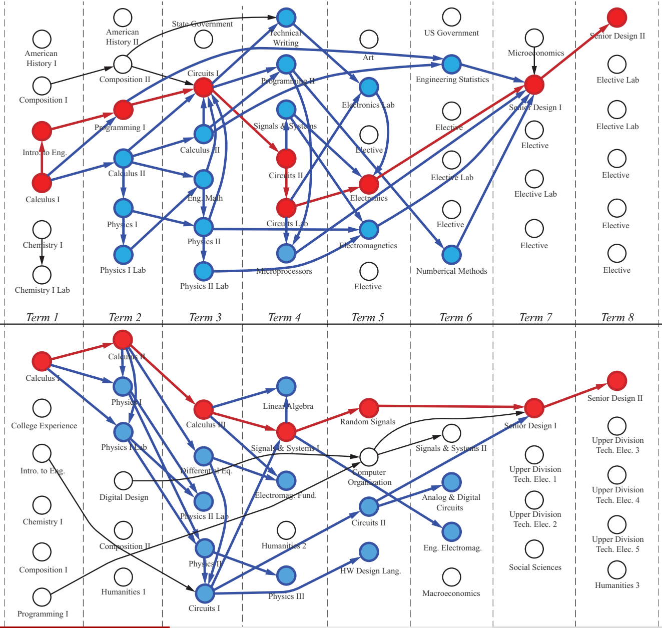



American History I
Composition I
Intro. to Eng.
Calculus I
Chemistry I
Chemistry I Lab
American History II
Composition II
Programming I
Calculus II
Physics I
Physics I Lab
State Government
Circuits I
Calculus III
Eng. Math
Physics II
Physics II Lab
Technical Writing
Programming II
Signals & Systems
Circuits II
Circuits Lab
Microprocessors
Art
Electronics Lab
Elective
Electronics
Electromagnetics
Elective
US Government
Engineering Statistics
Elective
Elective Lab
Elective
Numberical Methods
Microeconomics
Senior Design I
Elective
Elective Lab
Elective
Senior Design II
Elective Lab
Elective Lab
Elective
Elective
Elective
Term 1
Term 2
Term 3
Term 4
Term 5
Term 6
Term 7
Term 8
Calculus I
College Experience
Intro. to Eng.
Chemistry I
Composition I
Programming I
Calculus II
Physics I
Physics I Lab
Digital Design
Composition II
Humanities 1
Calculus III
Differential Eq.
Physics II Lab
Physics II
Circuits I
Linear Algebra
Signals & Systems I
Electromag. Fund.
Humanities 2
Physics III
Random Signals
Computer Organization
Circuits II
HW Design Lang.
Signals & Systems II
Analog & Digital Circuits
Eng. Electromag.
Macroeconomics
Senior Design I
Upper Division Tech. Elec. 1
Upper Division Tech. Elec. 2
Social Sciences
Senior Design II
Upper Division Tech. Elec. 3
Upper Division Tech. Elec. 4
Upper Division Tech. Elec. 5
Humanities 3

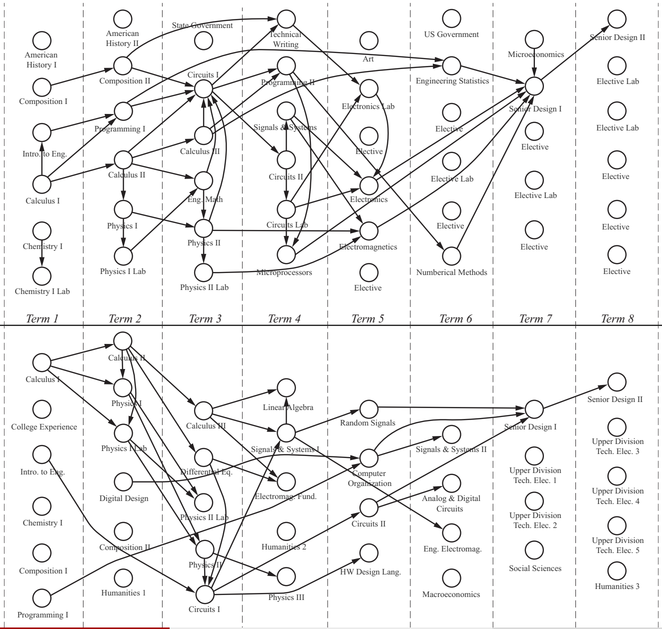
Two EE Programs



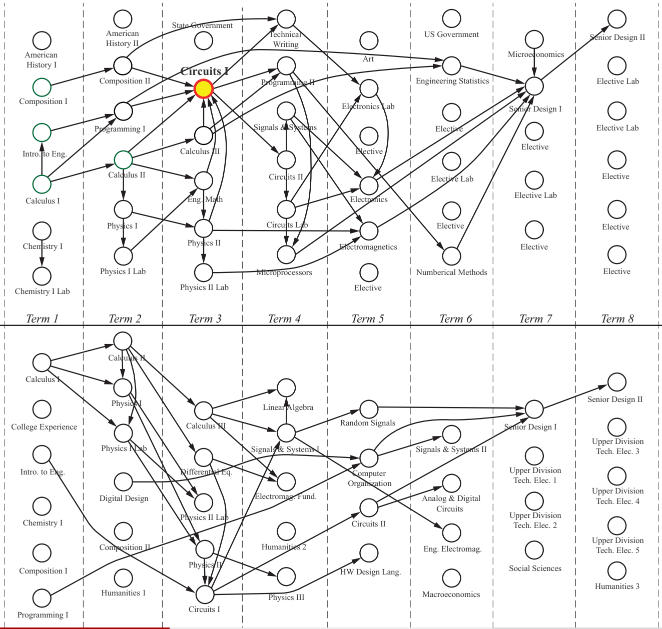
Two EE Programs



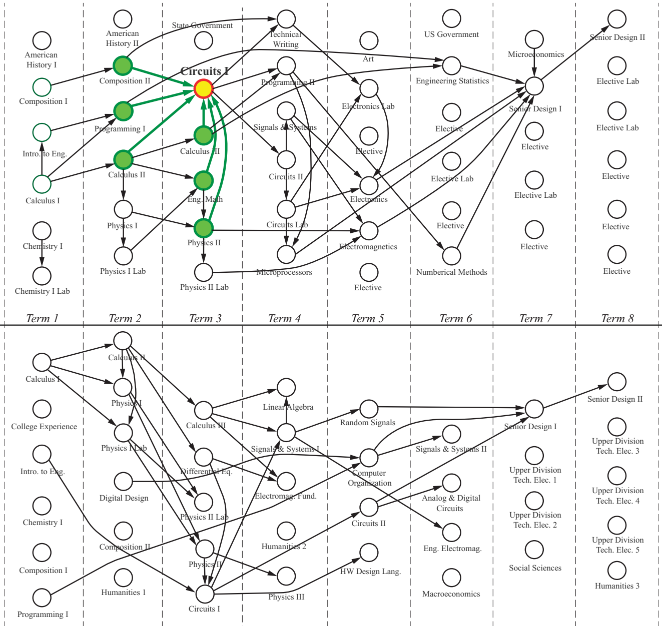
Two EE Programs



Two EE Programs

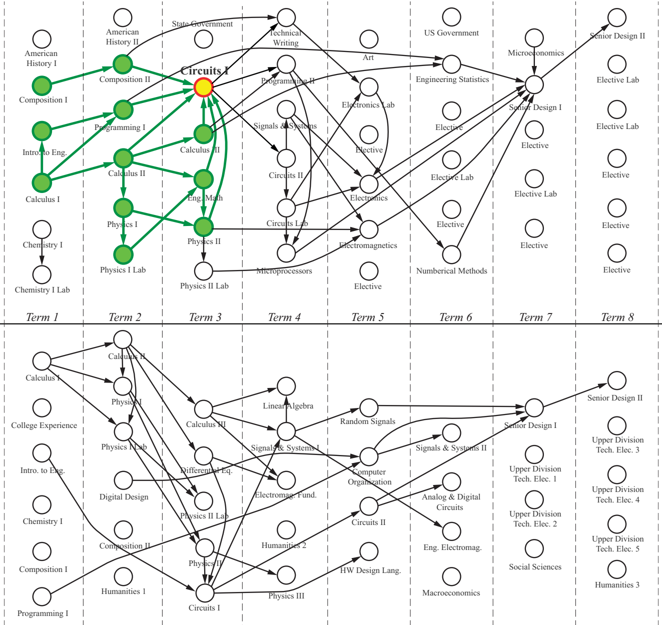


Two EE Programs



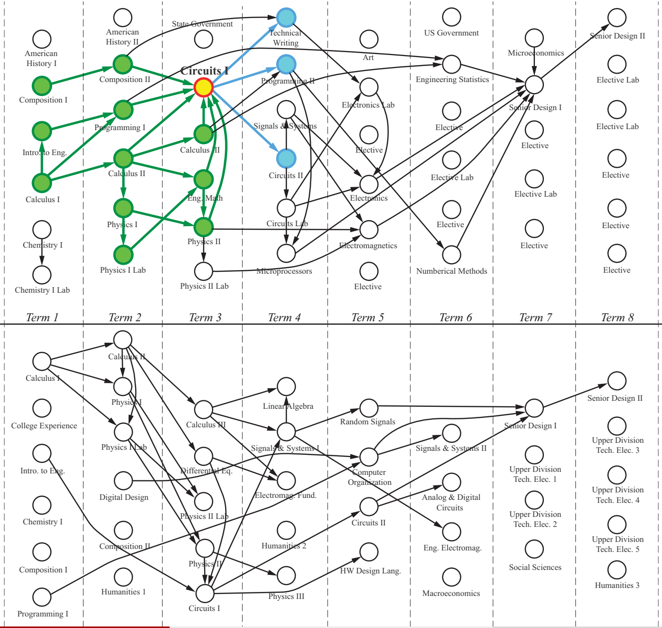
Two EE Programs

THE UNIVERSITY of NEW MEXICO

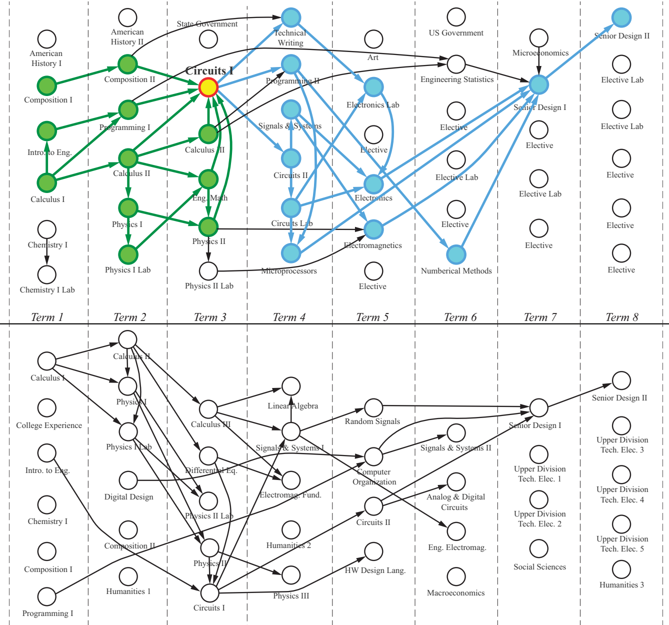


Two EE Programs

THE UNIVERSITY of NEW MEXICO

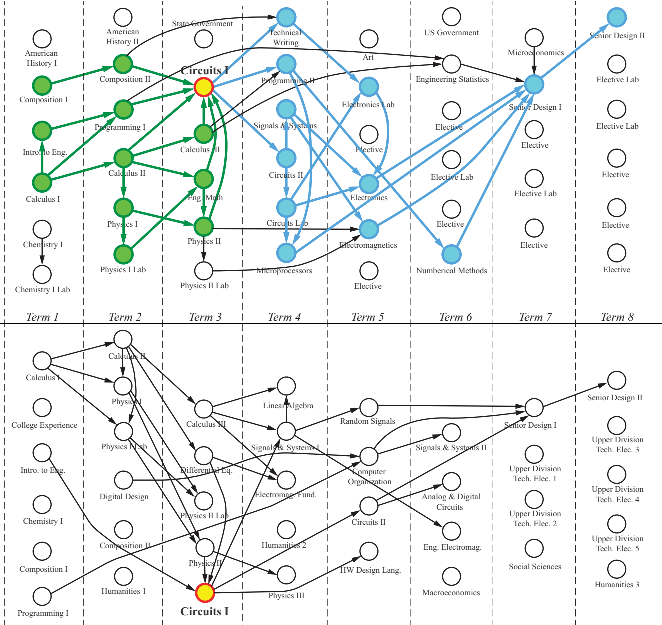


Two EE Programs

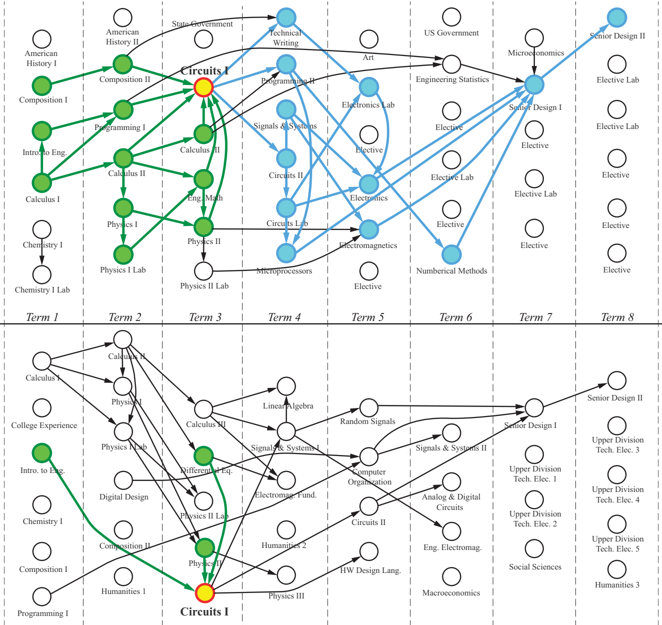


Two EE Programs

THE UNIVERSITY of NEW MEXICO

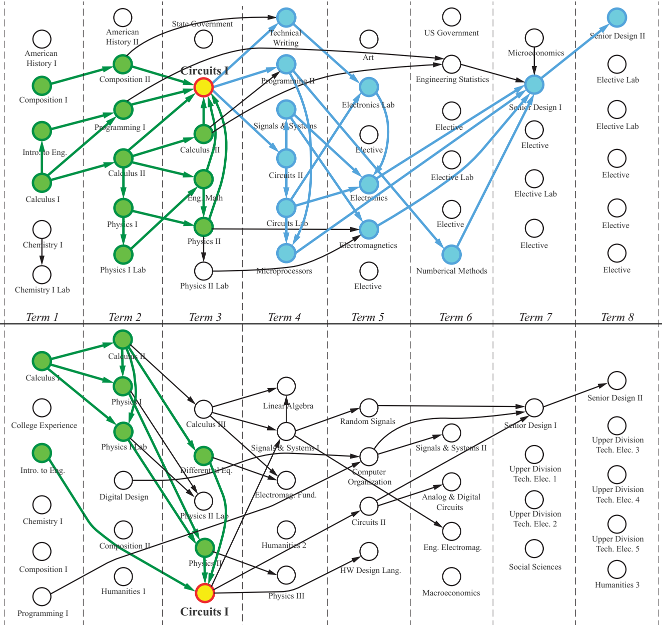


Two EE Programs



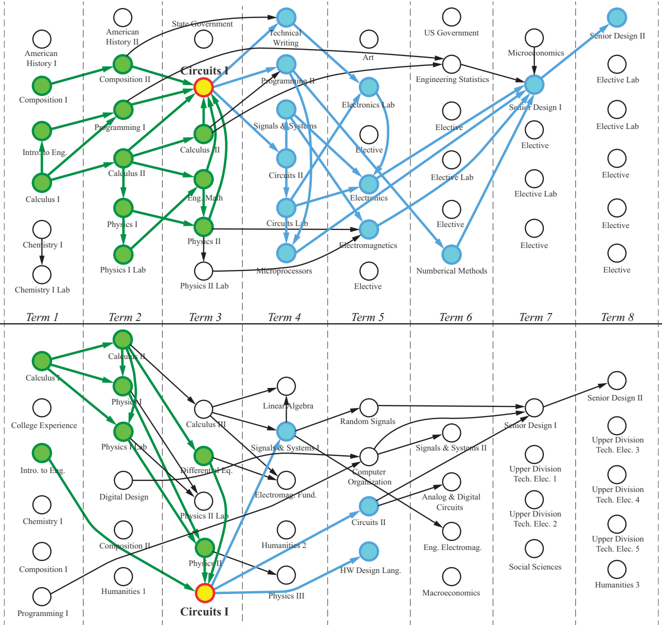
Two EE Programs

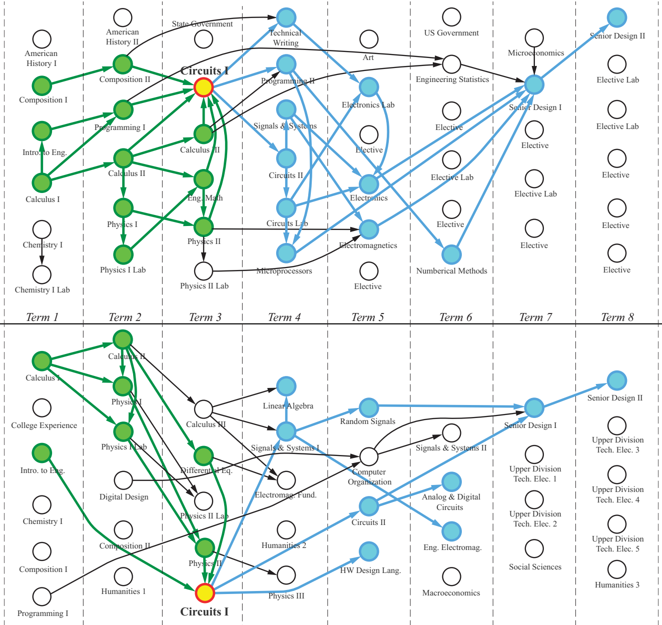
THE UNIVERSITY of NEW MEXICO



Two EE Programs

THE UNIVERSITY of NEW MEXICO




American History I
Composition I
Intro. to Eng.
Calculus I
Chemistry I
Chemistry I Lab
American History II
Composition II
Programming I
Calculus II
Physics I
Physics I Lab
State Government
Circuits I
Calculus III
Eng. Math
Physics II
Physics II Lab
Technical Writing
Programming II
Signals & Systems
Circuits II
Circuits Lab
Microprocessors
Art
Electronics Lab
Elective
Electronics
Electromagnetics
Elective
US Government
Engineering Statistics
Elective
Elective Lab
Elective
Numberical Methods
Microeconomics
Senior Design I
Elective
Elective Lab
Elective
Senior Design II
Elective Lab
Elective Lab
Elective
Elective
Elective
Term 1
Term 2
Term 3
Term 4
Term 5
Term 6
Term 7
Term 8
Calculus I
College Experience
Intro. to Eng.
Chemistry I
Composition I
Programming I
Calculus II
Physics I
Physics I Lab
Digital Design
Composition II
Humanities 1
Calculus III
Differential Eq.
Physics II Lab
Physics II
Circuits I
Linear Algebra
Signals & Systems I
Electromag. Fund.
Humanities 2
Physics III
Random Signals
Computer Organization
Circuits II
HW Design Lang.
Signals & Systems II
Analog & Digital Circuits
Eng. Electromag.
Macroeconomics
Senior Design I
Upper Division Tech. Elec. 1
Upper Division Tech. Elec. 2
Social Sciences
Senior Design II
Upper Division Tech. Elec. 3
Upper Division Tech. Elec. 4
Upper Division Tech. Elec. 5
Humanities 3

Curriculum Graph

- We represent a curriculum C consisting of n courses as a directed graph $G_C = (V, E)$,

Curriculum Graph

- We represent a curriculum C consisting of n courses as a directed graph $G_C = (V, E)$, where:
  - **Courses**: each vertex $v_1, \ldots, v_n \in V$ represents a requirement in C,

Curriculum Graph

- We represent a curriculum C consisting of n courses as a directed graph $G_C = (V, E)$, where:
  - **Courses**: each vertex $v_1, \ldots, v_n \in V$ represents a requirement in C,
  - **Prerequisites**: there is a directed edge $(v_i, v_j) \in E$ from requirement v_i to v_j if v_i must be satisfied prior to the satisfaction of v_j.

- We represent a curriculum C consisting of n courses as a directed graph $G_C = (V, E)$, where:
  - **Courses**: each vertex $v_1, \ldots, v_n \in V$ represents a requirement in C,
  - **Prerequisites**: there is a directed edge $(v_i, v_j) \in E$ from requirement v_i to v_j if v_i must be satisfied prior to the satisfaction of v_j.
- We refer to G_C as a **curriculum graph**.

Curriculum Graph

- We represent a curriculum C consisting of n courses as a directed graph $G_C = (V, E)$, where:
  - **Courses**: each vertex $v_1, \ldots, v_n \in V$ represents a requirement in C,
  - **Prerequisites**: there is a directed edge $(v_i, v_j) \in E$ from requirement v_i to v_j if v_i must be satisfied prior to the satisfaction of v_j.
- We refer to G_C as a **curriculum graph**. The structure of G_C influences how difficult it is to complete a curriculum

- We represent a curriculum C consisting of n courses as a directed graph $G_C = (V, E)$, where:
  - **Courses**: each vertex $v_1, \ldots, v_n \in V$ represents a requirement in C,
  - **Prerequisites**: there is a directed edge $(v_i, v_j) \in E$ from requirement v_i to v_j if v_i must be satisfied prior to the satisfaction of v_j.
- We refer to G_C as a **curriculum graph**. The structure of G_C influences how difficult it is to complete a curriculum
- The **structural complexity** of C, denoted α_C, is a function of relevant properties of G_C:

$$\alpha_C = g(G_C).$$

Structural Complexity

- Structural complexity is completely determined by G_C.

Structural Complexity

- Structural complexity is completely determined by G_C.
- The following graph properties impact student progression:
  - **Delay Factor:** characterized by long paths in the curriculum.

Structural Complexity

- Structural complexity is completely determined by G_C.
- The following graph properties impact student progression:
  - **Delay Factor:** characterized by long paths in the curriculum.
  - **Blocking Factor:** the number of courses a student is precluded from taking until they pass a given class.

Structural Complexity

- Structural complexity is completely determined by G_C.
- The following graph properties impact student progression:
  - **Delay Factor:** characterized by long paths in the curriculum.
  - **Blocking Factor:** the number of courses a student is precluded from taking until they pass a given class.
  - **Central Courses:** key courses in a curriculum — many prerequisites must be satisfied to reach them, and they "unblock" many courses in the curriculum that follow them.

Structural Complexity

- Structural complexity is completely determined by G_C.
- The following graph properties impact student progression:
  - **Delay Factor:** characterized by long paths in the curriculum.
  - **Blocking Factor:** the number of courses a student is precluded from taking until they pass a given class.
  - **Central Courses:** key courses in a curriculum — many prerequisites must be satisfied to reach them, and they “unblock” many courses in the curriculum that follow them.
  - **Degrees of Freedom:** the extent to which a curriculum can be rearranged if certain courses are not passed.

The **delay factor** associated with course v_k in curriculum $G_C = (V, E)$, denoted $d(v_k)$, is the number of nodes in the longest path in G_C that passes through v_k.

The **delay factor** associated with course v_k in curriculum $G_C = (V, E)$, denoted $d(v_k)$, is the number of nodes in the longest path in G_C that passes through v_k.

- $t = \max_{v_k \in V} d(v_k)$ determines the minimum number of terms required to complete a curriculum.

Structural Complexity – Delay Factor

The **delay factor** associated with course v_k in curriculum $G_C = (V, E)$, denoted $d(v_k)$, is the number of nodes in the longest path in G_C that passes through v_k.

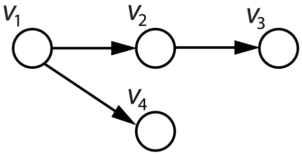
- $t = \max_{v_k \in V} d(v_k)$ determines the minimum number of terms required to complete a curriculum.
- Failure to complete a course on the longest path means that at least $t + 1$ terms will be required to complete the curriculum.

The **delay factor** associated with course v_k in curriculum $G_C = (V, E)$, denoted $d(v_k)$, is the number of nodes in the longest path in G_C that passes through v_k.

- $t = \max_{v_k \in V} d(v_k)$ determines the minimum number of terms required to complete a curriculum.
- Failure to complete a course on the longest path means that at least $t + 1$ terms will be required to complete the curriculum.

Ex:

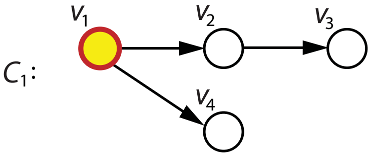
C_1:



The **delay factor** associated with course v_k in curriculum $G_C = (V, E)$, denoted $d(v_k)$, is the number of nodes in the longest path in G_C that passes through v_k.

- $t = \max_{v_k \in V} d(v_k)$ determines the minimum number of terms required to complete a curriculum.
- Failure to complete a course on the longest path means that at least $t + 1$ terms will be required to complete the curriculum.

Ex:

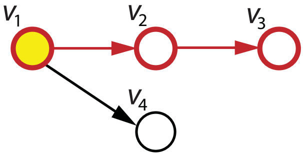


The **delay factor** associated with course v_k in curriculum $G_C = (V, E)$, denoted $d(v_k)$, is the number of nodes in the longest path in G_C that passes through v_k.

- $t = \max_{v_k \in V} d(v_k)$ determines the minimum number of terms required to complete a curriculum.
- Failure to complete a course on the longest path means that at least $t + 1$ terms will be required to complete the curriculum.

Ex:

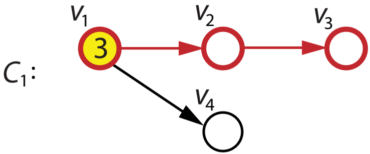
C_1:



The **delay factor** associated with course v_k in curriculum $G_C = (V, E)$, denoted $d(v_k)$, is the number of nodes in the longest path in G_C that passes through v_k.

- $t = \max_{v_k \in V} d(v_k)$ determines the minimum number of terms required to complete a curriculum.
- Failure to complete a course on the longest path means that at least $t + 1$ terms will be required to complete the curriculum.

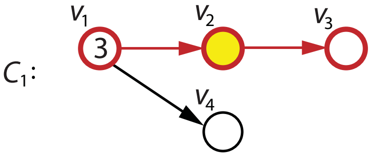
Ex:



The **delay factor** associated with course v_k in curriculum $G_C = (V, E)$, denoted $d(v_k)$, is the number of nodes in the longest path in G_C that passes through v_k.

- $t = \max_{v_k \in V} d(v_k)$ determines the minimum number of terms required to complete a curriculum.
- Failure to complete a course on the longest path means that at least $t + 1$ terms will be required to complete the curriculum.

Ex:

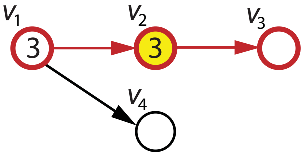


The **delay factor** associated with course v_k in curriculum $G_C = (V, E)$, denoted $d(v_k)$, is the number of nodes in the longest path in G_C that passes through v_k.

- $t = \max_{v_k \in V} d(v_k)$ determines the minimum number of terms required to complete a curriculum.
- Failure to complete a course on the longest path means that at least $t + 1$ terms will be required to complete the curriculum.

Ex:

C_1:

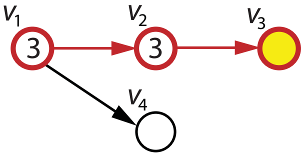


The **delay factor** associated with course v_k in curriculum $G_C = (V, E)$, denoted $d(v_k)$, is the number of nodes in the longest path in G_C that passes through v_k.

- $t = \max_{v_k \in V} d(v_k)$ determines the minimum number of terms required to complete a curriculum.
- Failure to complete a course on the longest path means that at least $t + 1$ terms will be required to complete the curriculum.

Ex:

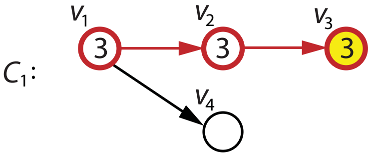
C_1:



The **delay factor** associated with course v_k in curriculum $G_C = (V, E)$, denoted $d(v_k)$, is the number of nodes in the longest path in G_C that passes through v_k.

- $t = \max_{v_k \in V} d(v_k)$ determines the minimum number of terms required to complete a curriculum.
- Failure to complete a course on the longest path means that at least $t + 1$ terms will be required to complete the curriculum.

Ex:

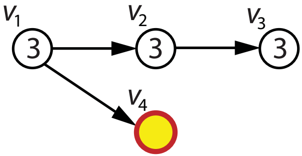


The **delay factor** associated with course v_k in curriculum $G_C = (V, E)$, denoted $d(v_k)$, is the number of nodes in the longest path in G_C that passes through v_k.

- $t = \max_{v_k \in V} d(v_k)$ determines the minimum number of terms required to complete a curriculum.
- Failure to complete a course on the longest path means that at least $t + 1$ terms will be required to complete the curriculum.

Ex:

C_1:

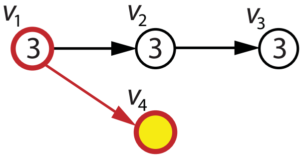


The **delay factor** associated with course v_k in curriculum $G_C = (V, E)$, denoted $d(v_k)$, is the number of nodes in the longest path in G_C that passes through v_k.

- $t = \max_{v_k \in V} d(v_k)$ determines the minimum number of terms required to complete a curriculum.
- Failure to complete a course on the longest path means that at least $t + 1$ terms will be required to complete the curriculum.

Ex:

C_1:



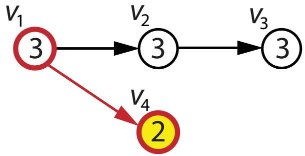
Structural Complexity – Delay Factor

The **delay factor** associated with course v_k in curriculum $G_C = (V, E)$, denoted $d(v_k)$, is the number of nodes in the longest path in G_C that passes through v_k.

- $t = \max_{v_k \in V} d(v_k)$ determines the minimum number of terms required to complete a curriculum.
- Failure to complete a course on the longest path means that at least $t + 1$ terms will be required to complete the curriculum.

Ex:

C_1:



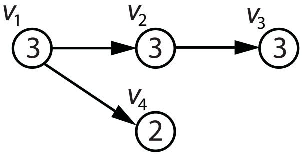
Structural Complexity – Delay Factor

The **delay factor** associated with course v_k in curriculum $G_C = (V, E)$, denoted $d(v_k)$, is the number of nodes in the longest path in G_C that passes through v_k.

- $t = \max_{v_k \in V} d(v_k)$ determines the minimum number of terms required to complete a curriculum.
- Failure to complete a course on the longest path means that at least $t + 1$ terms will be required to complete the curriculum.

Ex:

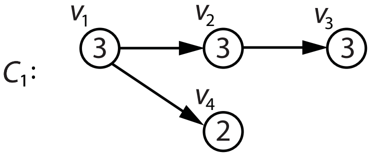
C_1:

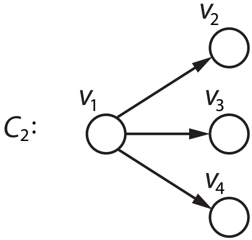


The **delay factor** associated with course v_k in curriculum $G_C = (V, E)$, denoted $d(v_k)$, is the number of nodes in the longest path in G_C that passes through v_k.

- $t = \max_{v_k \in V} d(v_k)$ determines the minimum number of terms required to complete a curriculum.
- Failure to complete a course on the longest path means that at least $t + 1$ terms will be required to complete the curriculum.

Ex:



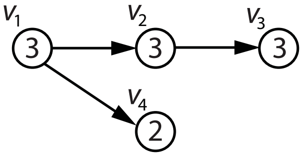


The **delay factor** associated with course v_k in curriculum $G_C = (V, E)$, denoted $d(v_k)$, is the number of nodes in the longest path in G_C that passes through v_k.

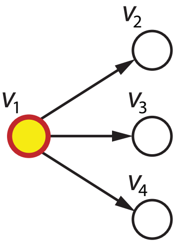
- $t = \max_{v_k \in V} d(v_k)$ determines the minimum number of terms required to complete a curriculum.
- Failure to complete a course on the longest path means that at least $t + 1$ terms will be required to complete the curriculum.

Ex:

C_1:



C_2:

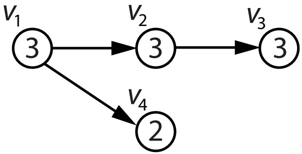


The **delay factor** associated with course v_k in curriculum $G_C = (V, E)$, denoted $d(v_k)$, is the number of nodes in the longest path in G_C that passes through v_k.

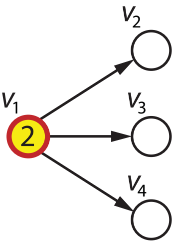
- $t = \max_{v_k \in V} d(v_k)$ determines the minimum number of terms required to complete a curriculum.
- Failure to complete a course on the longest path means that at least $t + 1$ terms will be required to complete the curriculum.

Ex:

C_1:



C_2:

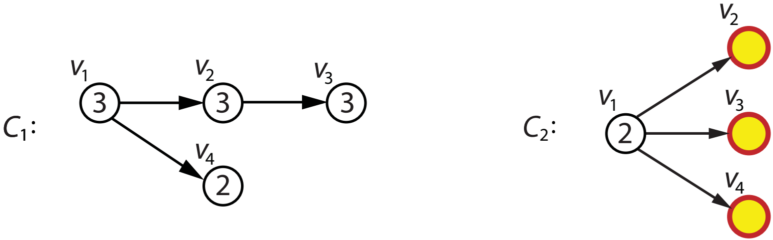


Structural Complexity – Delay Factor

The **delay factor** associated with course v_k in curriculum $G_C = (V, E)$, denoted $d(v_k)$, is the number of nodes in the longest path in G_C that passes through v_k.

- $t = \max_{v_k \in V} d(v_k)$ determines the minimum number of terms required to complete a curriculum.
- Failure to complete a course on the longest path means that at least $t + 1$ terms will be required to complete the curriculum.

Ex:

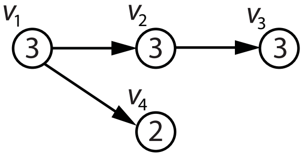


The **delay factor** associated with course v_k in curriculum $G_C = (V, E)$, denoted $d(v_k)$, is the number of nodes in the longest path in G_C that passes through v_k.

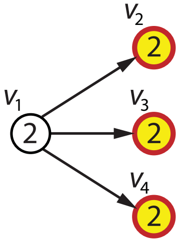
- $t = \max_{v_k \in V} d(v_k)$ determines the minimum number of terms required to complete a curriculum.
- Failure to complete a course on the longest path means that at least $t+1$ terms will be required to complete the curriculum.

Ex:

C_1:



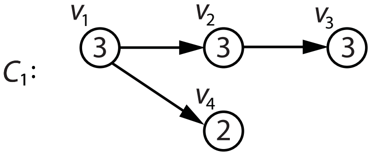
C_2:

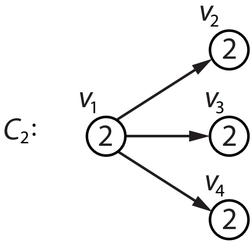


The **delay factor** associated with course v_k in curriculum $G_C = (V, E)$, denoted $d(v_k)$, is the number of nodes in the longest path in G_C that passes through v_k.

- $t = \max_{v_k \in V} d(v_k)$ determines the minimum number of terms required to complete a curriculum.
- Failure to complete a course on the longest path means that at least $t + 1$ terms will be required to complete the curriculum.

Ex:





Define the **blocking factor** associated with course v_i, denoted $b(v_k)$ in curriculum $G_C = (V, E)$ as:

$$b(v_i) = \sum_{v_j \in V} I(v_i, v_j),$$

where I is the indicator function:

$$I(v_i, v_j) = \begin{cases} 1, & \text{if } v_i \rightsquigarrow v_j; \\ 0, & \text{if } v_i \not\rightsquigarrow v_j. \end{cases}$$

Structural Complexity – Blocking Factor

Define the **blocking factor** associated with course v_i, denoted $b(v_k)$ in curriculum $G_C = (V, E)$ as:

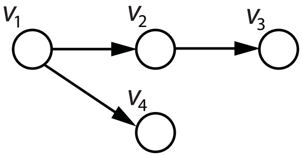
$$b(v_i) = \sum_{v_j \in V} I(v_i, v_j),$$

where I is the indicator function:

$$I(v_i, v_j) = \begin{cases} 1, & \text{if } v_i \rightsquigarrow v_j; \\ 0, & \text{if } v_i \not\rightsquigarrow v_j. \end{cases}$$

Ex:

C_1:



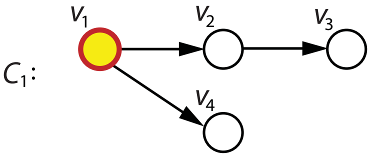
Define the **blocking factor** associated with course v_i, denoted $b(v_k)$ in curriculum $G_C = (V, E)$ as:

$$b(v_i) = \sum_{v_j \in V} I(v_i, v_j),$$

where I is the indicator function:

$$I(v_i, v_j) = \begin{cases} 1, & \text{if } v_i \rightsquigarrow v_j; \\ 0, & \text{if } v_i \not\rightsquigarrow v_j. \end{cases}$$

Ex:



Define the **blocking factor** associated with course v_i, denoted $b(v_k)$ in curriculum $G_C = (V, E)$ as:

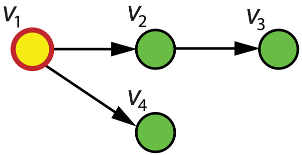
$$b(v_i) = \sum_{v_j \in V} I(v_i, v_j),$$

where I is the indicator function:

$$I(v_i, v_j) = \begin{cases} 1, & \text{if } v_i \rightsquigarrow v_j; \\ 0, & \text{if } v_i \not\rightsquigarrow v_j. \end{cases}$$

Ex:

C_1:



Structural Complexity – Blocking Factor

Define the **blocking factor** associated with course v_i, denoted $b(v_k)$ in curriculum $G_C = (V, E)$ as:

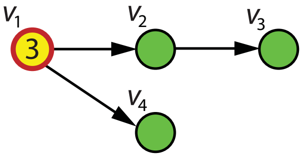
$$b(v_i) = \sum_{v_j \in V} I(v_i, v_j),$$

where I is the indicator function:

$$I(v_i, v_j) = \begin{cases} 1, & \text{if } v_i \rightsquigarrow v_j; \\ 0, & \text{if } v_i \not\rightsquigarrow v_j. \end{cases}$$

Ex:

C_1:



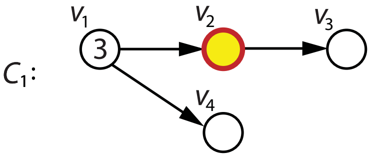
Define the **blocking factor** associated with course v_i, denoted $b(v_k)$ in curriculum $G_C = (V, E)$ as:

$$b(v_i) = \sum_{v_j \in V} I(v_i, v_j),$$

where I is the indicator function:

$$I(v_i, v_j) = \begin{cases} 1, & \text{if } v_i \rightsquigarrow v_j; \\ 0, & \text{if } v_i \not\rightsquigarrow v_j. \end{cases}$$

Ex:



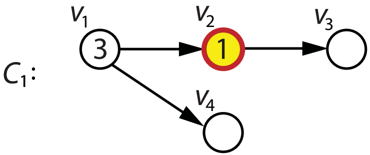
Define the **blocking factor** associated with course v_i, denoted $b(v_k)$ in curriculum $G_C = (V, E)$ as:

$$b(v_i) = \sum_{v_j \in V} I(v_i, v_j),$$

where I is the indicator function:

$$I(v_i, v_j) = \begin{cases} 1, & \text{if } v_i \rightsquigarrow v_j; \\ 0, & \text{if } v_i \not\rightsquigarrow v_j. \end{cases}$$

Ex:



Structural Complexity – Blocking Factor

Define the **blocking factor** associated with course v_i, denoted $b(v_k)$ in curriculum $G_C = (V, E)$ as:

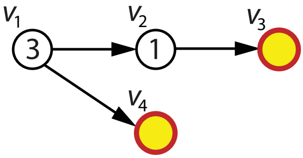
$$b(v_i) = \sum_{v_j \in V} I(v_i, v_j),$$

where I is the indicator function:

$$I(v_i, v_j) = \begin{cases} 1, & \text{if } v_i \rightsquigarrow v_j; \\ 0, & \text{if } v_i \not\rightsquigarrow v_j. \end{cases}$$

Ex:

C_1:



Define the **blocking factor** associated with course v_i, denoted $b(v_k)$ in curriculum $G_C = (V, E)$ as:

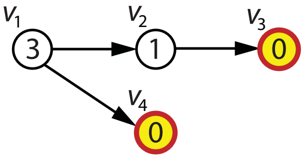
$$b(v_i) = \sum_{v_j \in V} I(v_i, v_j),$$

where I is the indicator function:

$$I(v_i, v_j) = \begin{cases} 1, & \text{if } v_i \rightsquigarrow v_j; \\ 0, & \text{if } v_i \not\rightsquigarrow v_j. \end{cases}$$

Ex:

C_1:



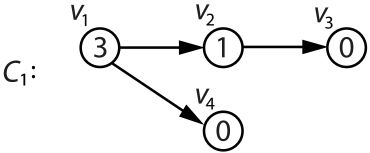
Define the **blocking factor** associated with course v_i, denoted $b(v_k)$ in curriculum $G_C = (V, E)$ as:

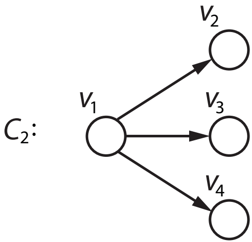
$$b(v_i) = \sum_{v_j \in V} I(v_i, v_j),$$

where I is the indicator function:

$$I(v_i, v_j) = \begin{cases} 1, & \text{if } v_i \rightsquigarrow v_j; \\ 0, & \text{if } v_i \not\rightsquigarrow v_j. \end{cases}$$

Ex:





Define the **blocking factor** associated with course v_i, denoted $b(v_k)$ in curriculum $G_C = (V, E)$ as:

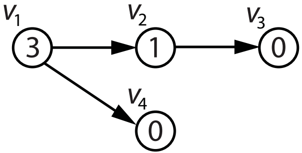
$$b(v_i) = \sum_{v_j \in V} I(v_i, v_j),$$

where I is the indicator function:

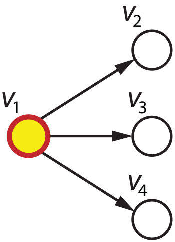
$$I(v_i, v_j) = \begin{cases} 1, & \text{if } v_i \rightsquigarrow v_j; \\ 0, & \text{if } v_i \not\rightsquigarrow v_j. \end{cases}$$

Ex:

C_1:



C_2:



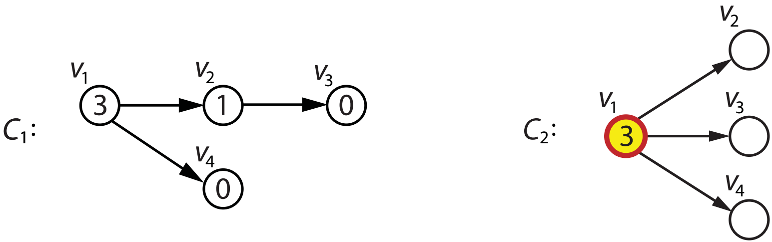
Define the **blocking factor** associated with course v_i, denoted $b(v_k)$ in curriculum $G_C = (V, E)$ as:

$$b(v_i) = \sum_{v_j \in V} I(v_i, v_j),$$

where I is the indicator function:

$$I(v_i, v_j) = \begin{cases} 1, & \text{if } v_i \rightsquigarrow v_j; \\ 0, & \text{if } v_i \not\rightsquigarrow v_j. \end{cases}$$

Ex:



Structural Complexity – Blocking Factor

Define the **blocking factor** associated with course v_i, denoted $b(v_k)$ in curriculum $G_C = (V, E)$ as:

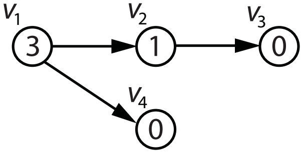
$$b(v_i) = \sum_{v_j \in V} I(v_i, v_j),$$

where I is the indicator function:

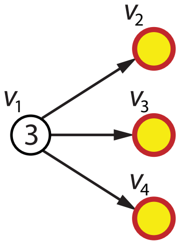
$$I(v_i, v_j) = \begin{cases} 1, & \text{if } v_i \rightsquigarrow v_j; \\ 0, & \text{if } v_i \not\rightsquigarrow v_j. \end{cases}$$

Ex:

C_1:



C_2:



Define the **blocking factor** associated with course v_i, denoted $b(v_k)$ in curriculum $G_C = (V, E)$ as:

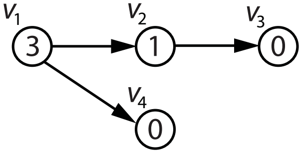
$$b(v_i) = \sum_{v_j \in V} I(v_i, v_j),$$

where I is the indicator function:

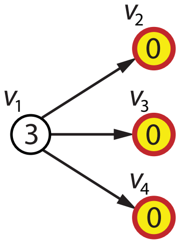
$$I(v_i, v_j) = \begin{cases} 1, & \text{if } v_i \rightsquigarrow v_j; \\ 0, & \text{if } v_i \not\rightsquigarrow v_j. \end{cases}$$

Ex:

C_1:



C_2:



Structural Complexity – Blocking Factor

Define the **blocking factor** associated with course v_i, denoted $b(v_k)$ in curriculum $G_C = (V, E)$ as:

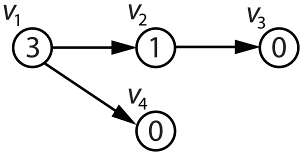
$$b(v_i) = \sum_{v_j \in V} I(v_i, v_j),$$

where I is the indicator function:

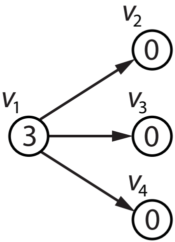
$$I(v_i, v_j) = \begin{cases} 1, & \text{if } v_i \rightsquigarrow v_j; \\ 0, & \text{if } v_i \not\rightsquigarrow v_j. \end{cases}$$

Ex:

C_1:



C_2:



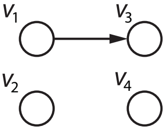
Network Information Theory —

- **Betweenness Centrality**: measures the extent to which a vertex lies on paths between other vertices.

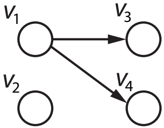
- We're working on the proper way to quantify this.

Structural Complexity – Degrees of Freedom

C_1:

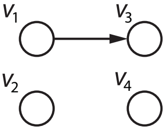


C_2:

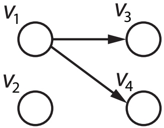


Structural Complexity – Degrees of Freedom

C_1:



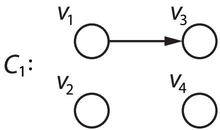
C_2:

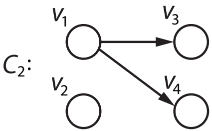


C_1 **Plan —**

term 1: v_1, v_2, v_4

term 2: v_3





C_1 **Plan —**

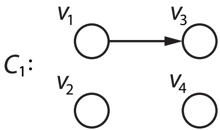
term 1: v_1, v_2, v_4

term 2: v_3

C_2 **Plan —**

term 1: v_1, v_2

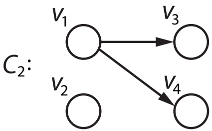
term 2: v_3, v_4



C_1 Plan —

term 1: v_1, v_2, v_4

term 2: v_3

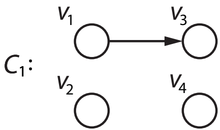


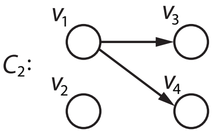
C_2 Plan —

term 1: v_1, v_2

term 2: v_3, v_4

Assume v_1 and v_4 are not passed on first attempt.



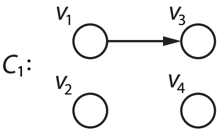


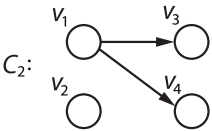
C_1 **Plan —**
term 1: v_1, v_2, v_4
term 2: v_3

C_2 **Plan —**
term 1: v_1, v_2
term 2: v_3, v_4

Assume v_1 and v_4 are not passed on first attempt.

C_1 **Revised Plan —**
term 1: v_1, v_2, v_4
term 2: v_1, v_4
term 3: v_3





C_1 **Plan —**
term 1: v_1, v_2, v_4
term 2: v_3

C_2 **Plan —**
term 1: v_1, v_2
term 2: v_3, v_4

Assume v_1 and v_4 are not passed on first attempt.

C_1 **Revised Plan —**
term 1: v_1, v_2, v_4
term 2: v_1, v_4
term 3: v_3

C_2 **Revised Plan —**
term 1: v_1, v_2
term 2: v_1
term 3: v_3, v_4
term 4: v_4

- The aforementioned graph properties all influence progression when students are unable to complete classes.

- The aforementioned graph properties all influence progression when students are unable to complete classes.
  Consider the two extremes:
  - Students always complete every class on the first attempt

- The aforementioned graph properties all influence progression when students are unable to complete classes.
  Consider the two extremes:
  - Students always complete every class on the first attempt — G_C is nearly irrelevant, all student will graduate on time.

- The aforementioned graph properties all influence progression when students are unable to complete classes.
  Consider the two extremes:
  - Students always complete every class on the first attempt — G_C is nearly irrelevant, all student will graduate on time.
  - Students are never able to complete a class

Instructional Complexity

- The aforementioned graph properties all influence progression when students are unable to complete classes.
  Consider the two extremes:
  - Students always complete every class on the first attempt — G_C is nearly irrelevant, all student will graduate on time.
  - Students are never able to complete a class — G_C is irrelevant, no student will ever graduate.
- There is another factor at play here.

- The aforementioned graph properties all influence progression when students are unable to complete classes.
  Consider the two extremes:
  - Students always complete every class on the first attempt — G_C is nearly irrelevant, all student will graduate on time.
  - Students are never able to complete a class — G_C is irrelevant, no student will ever graduate.
- There is another factor at play here.
- The **instructional complexity** of curriculum C, denoted γ_C, is a function of the difficulties of the courses in C:

$$\gamma_C = h(\text{course difficulties})$$

Instructional Complexity

- The difficulty of a course is a function of numerous factors, including:
  - instructor quality

Instructional Complexity

- The difficulty of a course is a function of numerous factors, including:
  - instructor quality
  - course content

Instructional Complexity

- The difficulty of a course is a function of numerous factors, including:
  - instructor quality
  - course content
  - support services provided

Instructional Complexity

- The difficulty of a course is a function of numerous factors, including:
  - instructor quality
  - course content
  - support services provided
  - student background preparation

Instructional Complexity

- The difficulty of a course is a function of numerous factors, including:
  - instructor quality
  - course content
  - support services provided
  - student background preparation
  - etc.

Instructional Complexity

- The difficulty of a course is a function of numerous factors, including:
  - instructor quality
  - course content
  - support services provided
  - student background preparation
  - etc.

- γ_C is extremely difficult to characterize.

Instructional Complexity

- The difficulty of a course is a function of numerous factors, including:
  - instructor quality
  - course content
  - support services provided
  - student background preparation
  - etc.

- γ_C is extremely difficult to characterize.

- The historic pass rates of the courses in a curriculum C provides a good approximation to γ_C.

- The overall complexity of a curriculum C, denoted Ψ_C, is a combination of the inherent difficulty associated with traversing a curriculum graph (structural complexity), and the manner in which the courses are taught (instructional complexity):

$$\Psi_C = f(\alpha_C, \gamma_C).$$

Curricular Complexity – A Model

- The overall complexity of a curriculum C, denoted Ψ_C, is a combination of the inherent difficulty associated with traversing a curriculum graph (structural complexity), and the manner in which the courses are taught (instructional complexity):

$$\Psi_C = f(\alpha_C, \gamma_C).$$

- $\uparrow$ complexity $\implies$ $\downarrow$ lower completion rates

Curricular Complexity – A Model

- The overall complexity of a curriculum C, denoted Ψ_C, is a combination of the inherent difficulty associated with traversing a curriculum graph (structural complexity), and the manner in which the courses are taught (instructional complexity):

$$\Psi_C = f(\alpha_C, \gamma_C).$$

- $\uparrow$ complexity $\implies$ $\downarrow$ lower completion rates

- Does higher curricular complexity lead to higher quality (improved student learning outcomes)?

- How do we make Ψ_C useful?

- How do we make Ψ_C useful? We need to better characterize $f(\cdot), g(\cdot)$ and $h(\cdot)$.

- How do we make Ψ_C useful? We need to better characterize $f(\cdot), g(\cdot)$ and $h(\cdot)$.

- We can learn $g(\cdot)$ —

$$\Psi_C = f(\alpha_C, \gamma_C)$$

- How do we make Ψ_C useful? We need to better characterize $f(\cdot), g(\cdot)$ and $h(\cdot)$.

- We can learn $g(\cdot)$ —

$$\begin{aligned}\Psi_C &= f(\alpha_C, \gamma_C) \\ &= f(g(G_C), h(\cdot, \cdot, \cdots))\end{aligned}$$

- How do we make Ψ_C useful? We need to better characterize $f(\cdot), g(\cdot)$ and $h(\cdot)$.
- We can learn $g(\cdot)$ —

$$
\begin{aligned}
\Psi_C &= f(\alpha_C, \gamma_C) \\
&= f(g(G_C), h(\cdot, \cdot, \cdots)) \\
&= f(g(G_C), \text{course pass rates})
\end{aligned}
$$

—Simulation—

- How do we make Ψ_C useful? We need to better characterize $f(\cdot), g(\cdot)$ and $h(\cdot)$.

- We can learn $g(\cdot)$ —

$$\begin{aligned} \Psi_C &= f(\alpha_C, \gamma_C) \\ &= f(g(G_C), h(\cdot, \cdot, \cdots)) \\ &= f(g(G_C), \text{course pass rates}) \end{aligned}$$

—Simulation—

$$\text{Completion rates} = f\,(g(G_C), \text{fixed pass rate})$$

- How do we make Ψ_C useful? We need to better characterize $f(\cdot), g(\cdot)$ and $h(\cdot)$.

- We can learn $g(\cdot)$ —

$$
\begin{aligned}
\Psi_C &= f(\alpha_C, \gamma_C) \\
&= f(g(G_C), h(\cdot, \cdot, \cdots)) \\
&= f(g(G_C), \text{course pass rates})
\end{aligned}
$$

—Simulation—

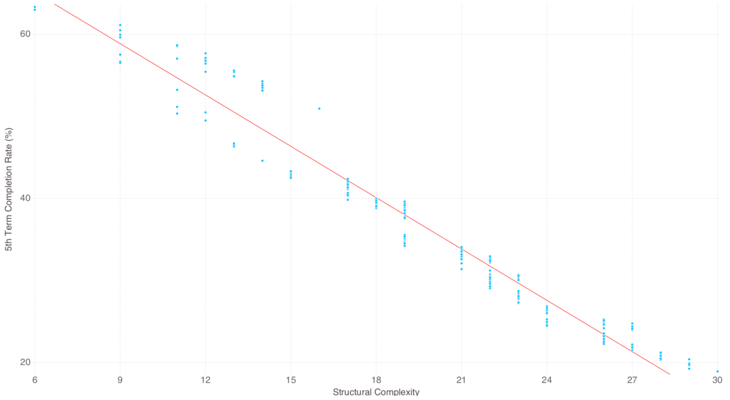
$$\text{Completion rates} = f\left(g(G_C), \text{fixed pass rate}\right)$$

Vary $g(\cdot)$, and measure correlation between completion rates and the various $g(\cdot)$'s.

Learning Structural Complexity

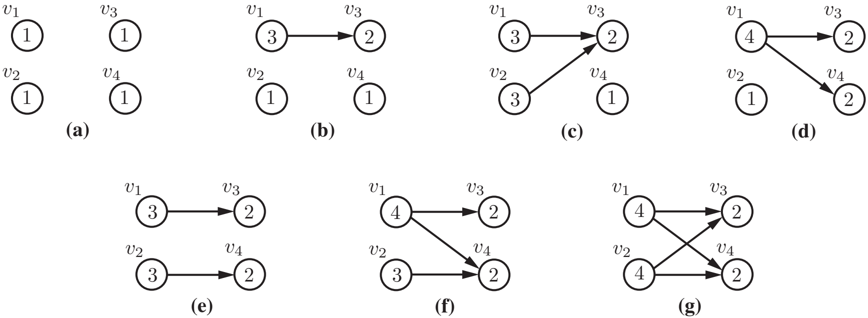
$$g(G_C) = \sum_{v_k \in V} (d(v_k) + b(v_k))$$

$$g(G_C) = \sum_{v_k \in V} \left(d(v_k) + b(v_k)\right)$$



$g(G_C)$ vs. 5-term completion rates for six-course curricula balanced over 3 terms

All four-course curricula balanced over two terms:



Structural Complexity Validation

	t_1	t_2	t_3	t_4
v_1	50	75	87.5	93.75
v_2	50	75	87.5	93.75
v_3	50	75	87.5	93.75
v_4	0	50	75	87.75
grad. rate	0	28.13	50.24	**72.30**

(a)

	t_1	t_2	t_3	t_4
v_1	50	75	87.75	93.75
v_2	50	75	87.75	93.75
v_3	0	25	50	68.75
v_4	50	75	87.75	93.75
grad. rate	0	10.55	33.78	**56.65**

(b)

	t_1	t_2	t_3	t_4
v_1	50	75	87.75	93.75
v_2	50	75	87.75	93.75
v_3	0	12.5	34.38	55.47
v_4	50	75	87.75	93.75
grad. rate	0	5.27	23.23	**45.71**

(c)

	t_1	t_2	t_3	t_4
v_1	50	75	87.75	93.75
v_2	50	75	87.75	93.75
v_3	0	25	50	68.75
v_4	0	25	50	68.75
grad. rate	0	3.52	19.25	**41.54**

(d)

	t_1	t_2	t_3	t_4
v_1	50	75	87.75	93.75
v_2	50	75	87.75	93.75
v_3	0	25	50	68.75
v_4	0	25	50	68.75
grad. rate	0	3.52	19.25	**41.54**

(e)

	t_1	t_2	t_3	t_4
v_1	50	75	87.75	93.75
v_2	50	75	87.75	93.75
v_3	0	25	50	68.75
v_4	0	12.5	34.38	55.47
grad. rate	0	1.76	13.24	**33.52**

(f)

	t_1	t_2	t_3	t_4
v_1	50	75	87.75	93.75
v_2	50	75	87.75	93.75
v_3	0	12.5	34.38	55.47
v_4	0	12.5	34.38	55.47
grad. rate	0	1.58	9.10	**27.04**

(g)

Application: Curricular Design Patterns

The notion of *design patterns* originated as a concept in architecture intended to capture the essence of an architectural design.

Architectural Design Patterns

The notion of *design patterns* originated as a concept in architecture intended to capture the essence of an architectural design.

- Allows architects to capture, in general terms, design decisions and ideas that have proven successful in solving particular design challenges.

Architectural Design Patterns

The notion of *design patterns* originated as a concept in architecture intended to capture the essence of an architectural design.

- Allows architects to capture, in general terms, design decisions and ideas that have proven successful in solving particular design challenges.
- Provides a useful collection of knowledge that other architects may consult in the future when confronting similar design challenges.

Architectural Design Patterns

The notion of *design patterns* originated as a concept in architecture intended to capture the essence of an architectural design.

- Allows architects to capture, in general terms, design decisions and ideas that have proven successful in solving particular design challenges.
- Provides a useful collection of knowledge that other architects may consult in the future when confronting similar design challenges.
- These design patterns constitute a language that architects may use to more efficiently communicate with one another.

Summary:

“[e]ach pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice.”

- In 1987, software engineers began experimenting with the notion of applying design patterns to the challenges confronted in the design of large-scale software systems.

Software Design Patterns

- In 1987, software engineers began experimenting with the notion of applying design patterns to the challenges confronted in the design of large-scale software systems.
- This led to the formalization of a large number of software design patterns that have been shown to greatly aid in the development of complex software systems.

Software Design Patterns

- In 1987, software engineers began experimenting with the notion of applying design patterns to the challenges confronted in the design of large-scale software systems.
- This led to the formalization of a large number of software design patterns that have been shown to greatly aid in the development of complex software systems.
- Provides developers with a reusable set of proven solutions to generalized problems.

Software Design Patterns

- In 1987, software engineers began experimenting with the notion of applying design patterns to the challenges confronted in the design of large-scale software systems.
- This led to the formalization of a large number of software design patterns that have been shown to greatly aid in the development of complex software systems.
- Provides developers with a reusable set of proven solutions to generalized problems.
- Software design patterns are similar to their architectural counterparts in that they provide software engineers with a language that can be used to discuss software design issues.

Curriculum Design Pattern

THE UNIVERSITY of NEW MEXICO

Definition: (Curricular Design Pattern). A collection of curricular and co-curricular learning activities intentionally structured so as to allow students to attain a set of learning outcomes within a given educational context.

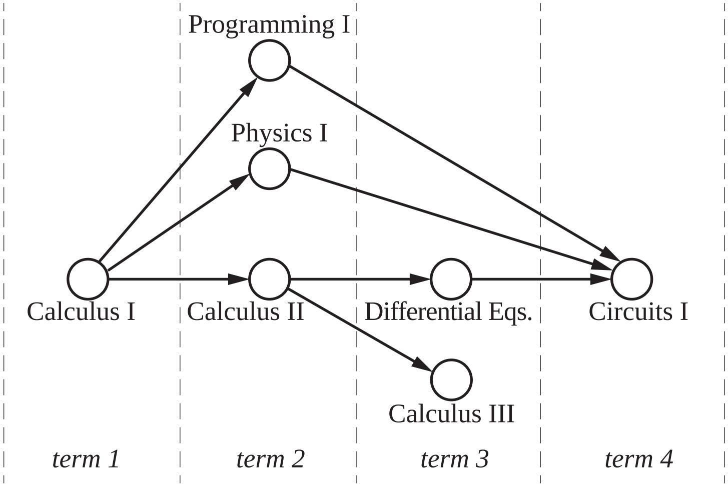
Example: Traditional Circuits I–Calculus Ready

This curricular design pattern is constructed so as to attain a set of learning outcomes that involve the ability to design, build and analyze simple electronic circuits, under the assumption that a student is prepared for Calculus I. Some of the specific learning outcomes are as follows. Students will:

1. Understand the functions of basic electrical circuit elements and sources;
2. Have the ability to apply Ohm's and Kirchhoff's circuit laws in the lumped element model of electrical circuits;
3. Appreciate the consequences of linearity, in particular the principle of superposition and Thevenin and Norton equivalent circuits;
4. Understand the concept of state in a dynamical physical system and have the ability to analyze simple first and second order linear circuits containing memory elements.

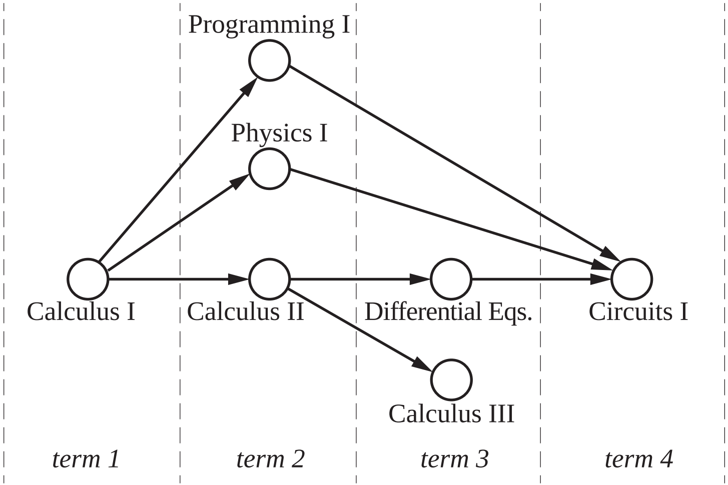
Traditional Circuits I – Calculus Ready

A curricular pattern of courses designed to allow students to attain the learning outcomes:



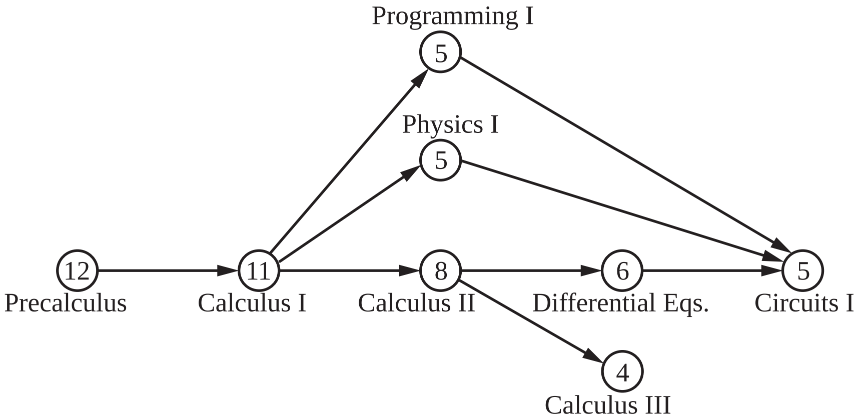
Traditional Circuits I – Calculus Ready

A curricular pattern of courses designed to allow students to attain the learning outcomes:

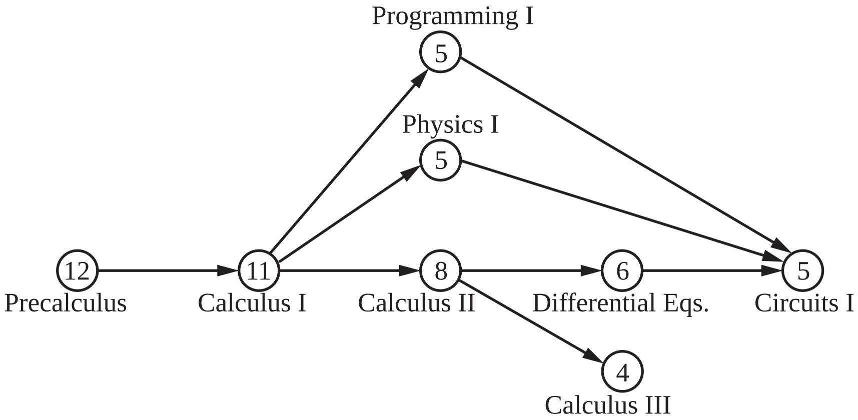


Traditional Circuits I – Not Calculus Ready

If we include Pre-Calculus:



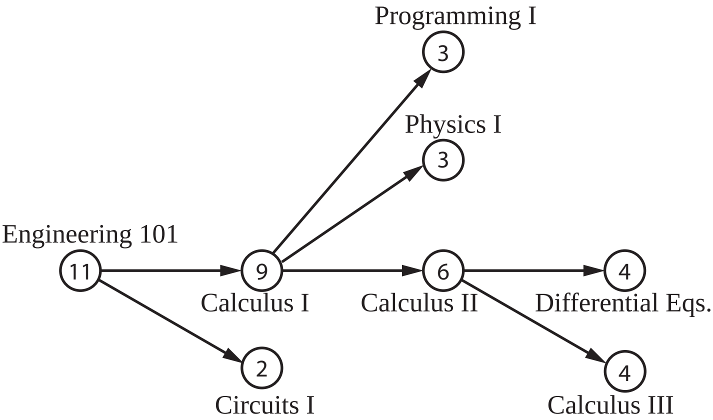
If we include Pre-Calculus:



Structural Complexity = 56

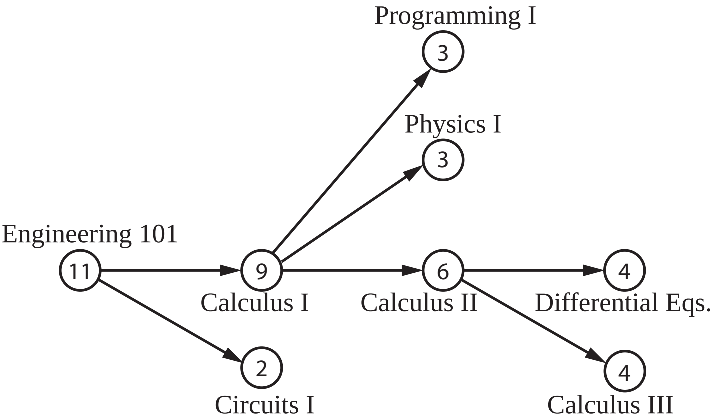
Traditional Circuits I – Alternative

Redesigning the pattern:



Structural Complexity = 41

Redesigning the pattern:



Structural Complexity = 41