Lab No. Final Exam Prep:

Ceng 242

By: David Li

V00818631

Instructor: Daler Rakhmatov

Final Exam: August 5, ECS 130

November 18, 2016

Contents

1 Exam Structure **2**

2 CRT **3**
2.1 Example 2.1 . . . 3
2.2 Example 2.2 . . . 4
2.3 Example 2.3 . . . 5

3 Electric Networks **5**
3.1 Chosen Tree and SubTree for the above problem . . . 6
3.2 Reuse this box later . . . 7
3.3 Checks in Matlab . . . 7
3.4 Other Useless stuff . . . 9

4 Linear Machines **10**
4.1 Finite Fields . . . 10
4.2 Transfer Function simplification . . . 11
4.2.1 Dividing by GCD . . . 13
4.2.2 Block Diagram . . . 13
4.3 Putzer's algorithm . . . 14
4.4 Difference-equation/state-variable models . . . 23

5 Solving HLR/NHLR **27**
5.1 Textbook Recursion Questions . . . 28
5.2 Textbook Recursion Questions 2 . . . 29
5.3 Introduction to OGF(Ordinary Generating Functions) . . . 30
5.4 OGF Tables . . . 31
5.5 Example Problems . . . 40
5.5.1 Fibonacci numbers . . . 40
5.5.2 Towers Of Hanoi . . . 40
5.5.3 Solution . . . 42
5.5.4 Recurrence Equation . . . 42
5.6 Another Problem . . . 43
5.7 Problem 5 . . . 44
5.8 Another Quality Box . . . 44
5.8.1 Another Good Box . . . 45
5.9 Last Recursion Problem . . . 45
5.9.1 P set 2 . . . 46
5.9.2 Q1 . . . 47
5.9.3 Q2 . . . 47
5.9.4 Q3 . . . 48

6 Max-flow network problem **49**
6.1 Minimum capacity and Maximum Flow . . . 49

7 Knapsack problem **53**
7.1 Greedy Heuristic . . . 53
7.1.1 Another Reasonably good box . . . 54
7.2 Branch And Bound . . . 54
7.2.1 Simple Example . . . 56
7.3 Description . . . 56
7.4 Ceng 242 Slides for Knapsack Problem . . . 57

8 ILP/network-flow problem formulations **63**
8.1 Network-flow . . . 66

List of Figures

1 Chinese Remainder Theorem . . . 3
2 Example Circuit: Click for more Info . . . 5
3 Corresponding Graph . . . 6
4 Tree And CoTree . . . 6
5 Example Graph and Tree Click for more Info . . . 9
6 Edmonds Karp . . . 49
7 Slides Example 7.1. . . . 50
8 Example 7.1 . . . 51
9 Example 7.2 Network Flow Example Link . . . 51
10 Example 7.2 **part 1** . . . 52
11 Exaample 7.2 **part 2** . . . 52
12 Network Flow Example . . . 52
13 Example 7.3 . . . 52
14 Example 7.4 **part 1** . . . 53
15 Example 7.4 **part 2** . . . 53
16 Go find more network flow problems . . . 53
17 Branch And Bound Example from Ceng 242 HW . . . 55
18 Simple Branch And Bound . . . 56

List of Scripts

1 Provided answers and Fundamental Matrices . . . 7
2 Calculations . . . 8
3 Practically useless . . . 8
4 Finite Field computation using Matlab function gfDeconv . . . 11

1 Exam Structure

1. Chinese remainder theorem (CRT) = 5 points
2. Electric networks = 5 points (matrices, KCL, KVL)
3. Linear machines = 10 points (difference-equation/state-variable models, Putzer's algorithm, transfer function simplification, implementation)
4. Solving HLR/NHLR = 10 points
5. ILP/network-flow problem formulations = 5 points
6. Solving max-flow network problem = 5 points (Edmonds-Karp algorithm)
7. Solving knapsack problem = 10 points (greedy heuristic, branch-and-bound method)

2 CRT

Theorem (CRT)

Let $\{m_i \in \mathbb{N} \mid i \in I_n\}$ *be a set of* n **pairwise coprime** *positive integers, i.e.,* $\gcd(m_j, m_k) = 1$ *for all* $j, k \in I_n$ *whenever* $j \neq k$. *Also, let* $M = \prod_{i=1}^{n} m_i$.

There exists a **bijection** $\psi : \mathbb{Z}_M \mapsto \mathbb{Z}_{m_1} \times \mathbb{Z}_{m_2} \times ... \times \mathbb{Z}_{m_n}$ *such that for every* $[x]_M \in \mathbb{Z}_M$,

$$\psi([x]_M) = ([y_1]_{m_1}, [y_2]_{m_2}, ..., [y_n]_{m_n}),$$

where we let $y_i = x \bmod m_i$ *for all* $i \in I_n$.

The **inverse** *of* ψ *is a bijection* $\psi^{-1} : \mathbb{Z}_{m_1} \times \mathbb{Z}_{m_2} \times ... \times \mathbb{Z}_{m_n} \mapsto \mathbb{Z}_M$ *given by the following formula:*

$$\psi^{-1}([y_1]_{m_1}, [y_2]_{m_2}, ..., [y_n]_{m_n}) = [y_1^*]_M \oplus [y_2^*]_M \oplus ... \oplus [y_n^*]_M,$$

which produces $[x]_M$ *mapped to* $([y_1]_{m_1}, [y_2]_{m_2}, ..., [y_n]_{m_n})$. *For all* $i \in I_n$, *we let* $y_i^* = (y_i \cdot \frac{M}{m_i} \cdot \lambda_i) \bmod M$, *where* $[\lambda_i]_{m_i} = [\frac{M}{m_i} \bmod m_i]_{m_i}^{-1}$.

Figure 1: Chinese Remainder Theorem

2.1 Example 2.1

Prove the following: $\psi^{-1}([2]_5 \ , \ [3]_7) = \psi[17]_{35}$
Answer on Next Page

$$[\lambda]_1 = [\frac{35}{5} mod 5]_5^{-1} = [7 mod 5]_5^{-1} = [3]_5$$
$$[\lambda]_2 = [\frac{35}{7} mod 7]_7^{-1} = [5 mod 7]_7^{-1} = [3]_7$$
$$y_1^\star = y_1 \bullet \frac{35}{5} \bullet 3 = 2 \bullet 7 \bullet 3 = 42$$
$$y_2^\star = y_2 \bullet \frac{35}{7} \bullet 3 = 3 \bullet 5 \bullet 3 = 45$$
$$[y_1^\star]_{35} \oplus [y_2^\star]_{35} = [42 + 45]_{35} = [17]_{35}$$

2.2 Example 2.2

To compute

$$[17 \times 17]_{35}$$

Remember that

$$\psi[17]_{35} = \psi^{-1}([2]_5 \ , \ [3]_7)$$

So now

$$\psi[17 \times 17]_{35} = \psi^{-1}([2 \times 2]_5 \ , \ [3 \times 3]_7)$$
$$\psi^{-1}([4]_5 \ , \ [9]_7)$$

Apply CRT

$$\psi^{-1}([4]_5 \ , \ [9]_7) = \psi([x]_{35})$$
$$[\lambda]_1 = \left[\frac{35}{5} \text{mod } 5\right]_5^{-1} = [7 \text{mod} 5]_5^{-1} = [3]_5$$
$$[\lambda]_2 = \left[\frac{35}{7} \text{mod} 7\right]_7^{-1} = [5 \text{mod} 7]_7^{-1} = [3]_7$$
$$y_1^\star = y_1 \bullet \frac{35}{5} \bullet 3 = 4 \bullet 7 \bullet 3 = 84$$
$$y_2^\star = y_2 \bullet \frac{35}{7} \bullet 3 = 9 \bullet 5 \bullet 3 = 135$$

$$[y_1^\star]_{35} \oplus [y_2^\star]_{35} = [84 + 135]_{35} = [219]_{35} = [219]_{35} = [9]_{35}$$

2.3 Example 2.3

$$\psi^{-1}([6]_7\ ,\ [4]_8) = \psi[x]_{56} \tag{1}$$

$$[\lambda]_1 = \left[\frac{56}{7}\text{mod } 7\right]_7^{-1} = [8\text{mod}7]_7^{-1} = [1]_7 \tag{2}$$

$$[\lambda]_2 = \left[\frac{56}{8}\text{mod } 8\right]_8^{-1} = [7\text{mod}8]_8^{-1} = [7]_8 \tag{3}$$

$$y_1^\star = y_1 \times \frac{M}{m_1} \times \lambda_1 = 6 \times 8 \times 1 = 48 \tag{4}$$

$$y_2^\star = y_2 \times \frac{M}{m_2} \times \lambda_1 = 4 \times 7 \times 7 = 196 \tag{5}$$

$$[y_1^\star]_{56} \oplus [y_2^\star]_{56} = [48 + 196]_{56} = [244]_{56} = [20]_{35} \tag{6}$$

3 Electric Networks

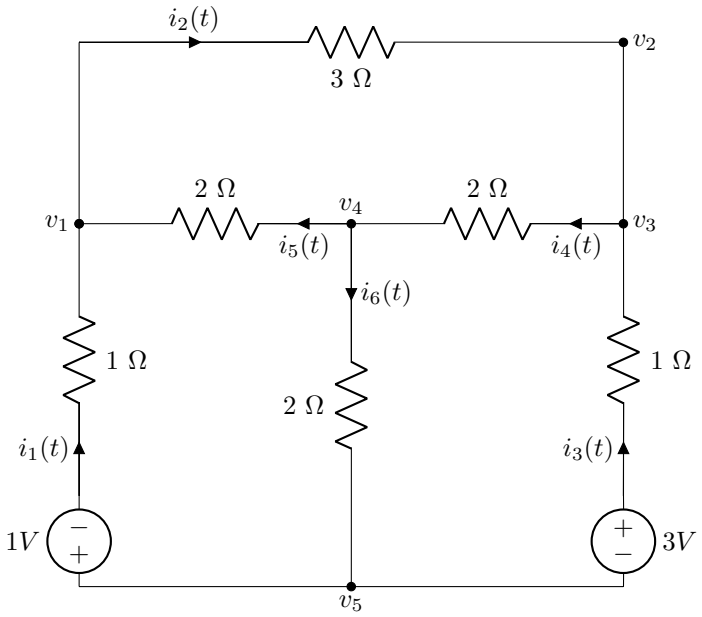


Figure 2: Example Circuit: Click for more Info

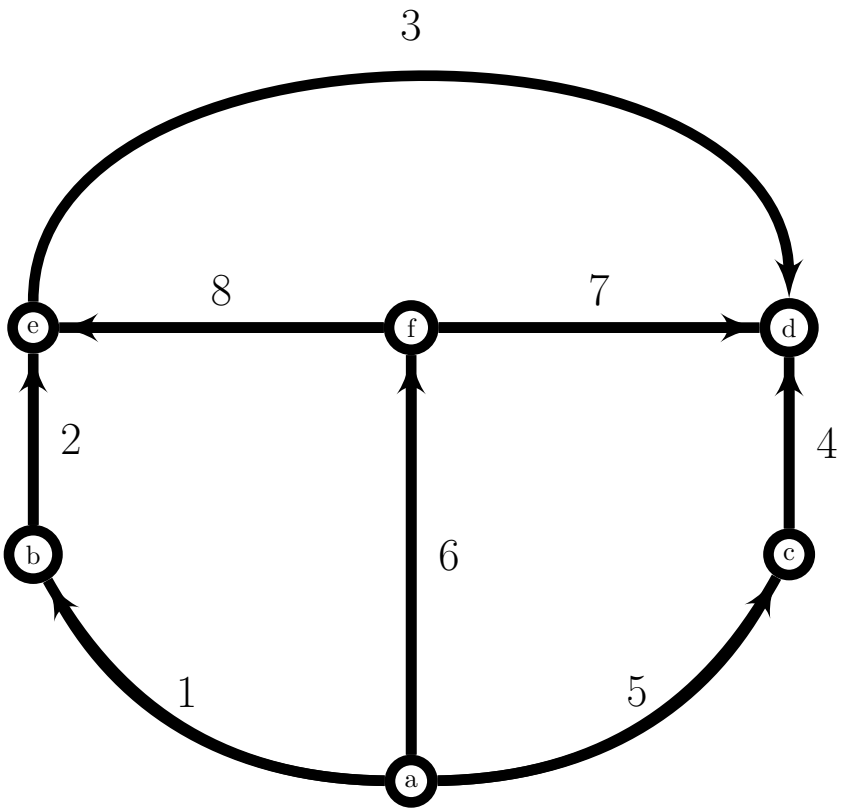


Figure 3: Corresponding Graph

3.1 Chosen Tree and SubTree for the above problem

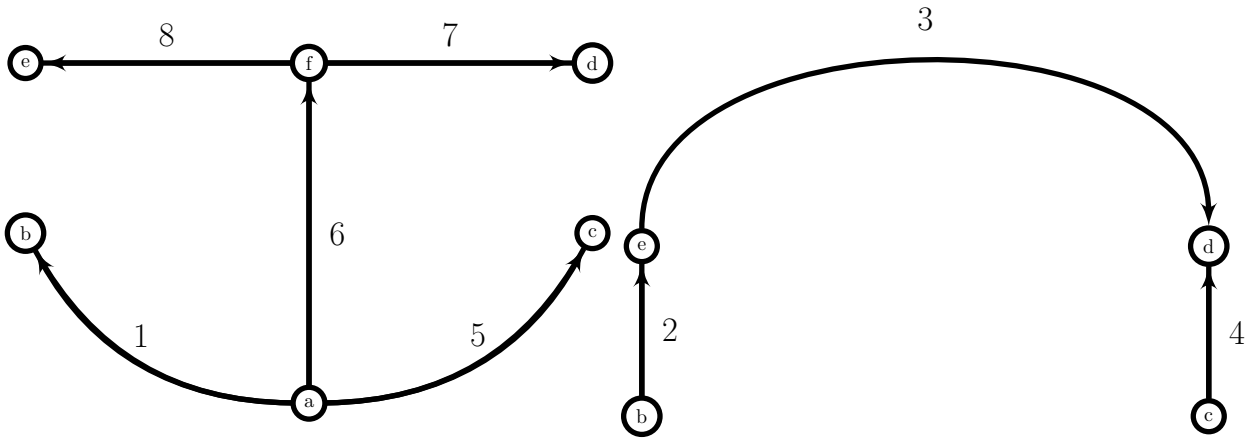


Figure 4: Tree And CoTree

B_f and Q_F with edge ordering

$$\text{Edge Ordering} \quad e_1 \quad e_2 \quad e_3 \quad e_4 \quad e_5 \quad e_6 \quad e_7 \quad e_8$$

$$Q_f = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 & 1 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 1 \end{bmatrix} \quad B_f = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & -1 & -1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \end{bmatrix}$$

3.2 Reuse this box later

B_f and Q_F with edge ordering

$$\text{Edge Ordering} \quad e_1 \quad e_2 \quad e_3 \quad e_4 \quad e_5 \quad e_6 \quad e_7 \quad e_8$$

$$Q_f = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 & 1 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 1 \end{bmatrix} \quad B_f = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & -1 & -1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \end{bmatrix}$$

3.3 Checks in Matlab

Script 1: Provided answers and Fundamental Matrices

```
%% Verification Electric network Example
Bf=[1  1  0  0  0 -1 -1  0;
    0  0  1  0  0  0  1 -1;
    0  0  0  1  1  1  0  1;]
%% provided voltage answer
v= [-1; 0.93; 1.85; 1.22; -3; 0.57; -0.64; 1.21;]

Qf= [1  0  0  0  0 -1  0  0;
     0  1  0  0  0  0  0 -1;
     0  0  1  0  0  1  0 -1;
```

```
      0  0  0  1  0   1 -1   0;
      0  0  0  0  1   0  1  -1;]
%% provided current answer
i = [0.93;   1.22;  0.29;  -0.32;  0.60;  0.93;  0.62;  1.22;]
```

Script 2: Calculations

```
Bf * v

ans =

   1.0e-15 *

    0.1110
         0
         0

Qf*i

ans =

         0
         0
         0
   -0.0100
         0
```

Script 3: Practically useless

```
Bft= [1  0 -1 -1  0; 0  0  0  1  -1; 0  1  1  0  1;]

Bft =

     1     0    -1    -1     0
     0     0     0     1    -1
     0     1     1     0     1
```

Bft= [1 0 −1 −1 0; 0 0 0 1 −1; 0 1 1 0 1;]

Qft =

$$\begin{bmatrix} -1 & 0 & 0 \\ 0 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & -1 & 0 \\ 0 & 1 & -1 \end{bmatrix}$$

3.4 Other Useless stuff

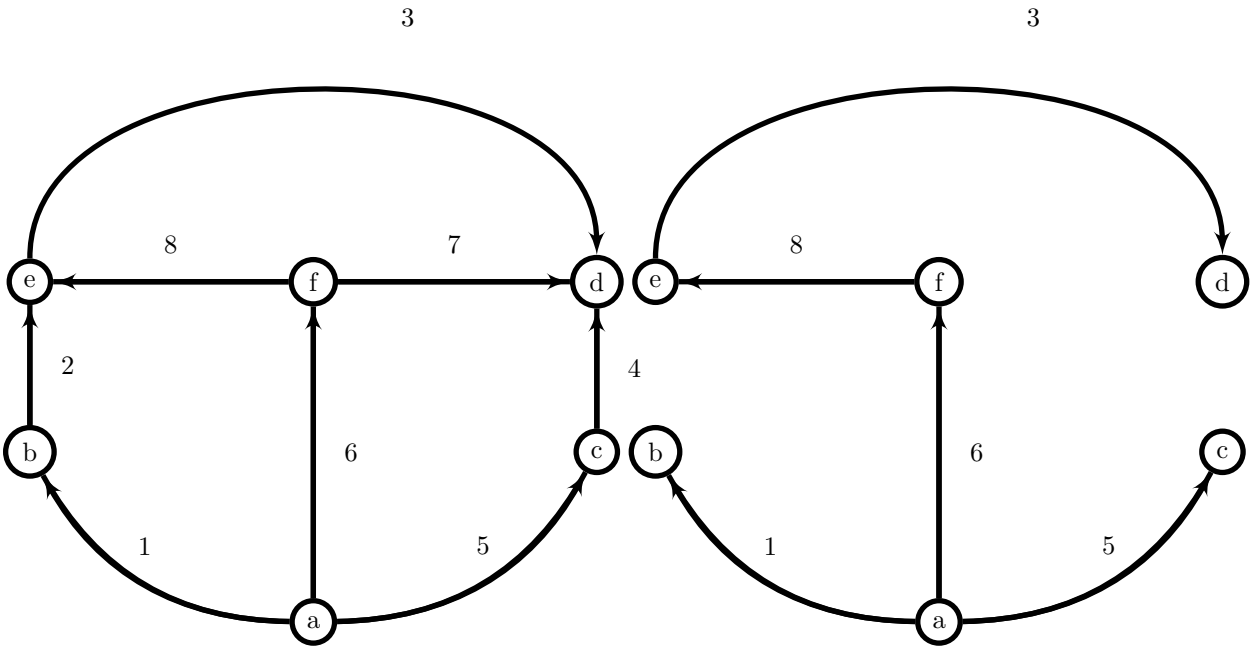


Figure 5: Example Graph and Tree Click for more Info

CutSet Matrix

Tree Branch	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8
1	1	−1						
3			1	−1				1
5				−1	1			
6		1		−1		1		1
7		1		−1			1	1

Finite fields

Theorem

Let F be a **finite field**, *i.e., a field with a finite number of elements. Then $|F| = p^n$ for some* **prime** *p and $n \in \mathbb{N}$.*

Theorem

Finite fields of the same size are **isomorphic**.

Thus, given any prime p and $n \in \mathbb{N}$, the following statements are true.

- **Existence:** There exists some finite field F with p^n elements;
- **Uniqueness:** Such F is unique up to isomorphism.

A finite field F of size p^n is often written as $GF(p^n)$ and called a **Galois field** of **order** p^n and **characteristic** p, meaning that p is the *smallest positive integer* such that $\underbrace{x + x + \ldots + x}_{p \text{ terms}} = 0$ for every $x \in GF(p^n)$.

For example, $\mathbb{Z}_p$ with respect to $\oplus$ and $\otimes$ is $GF(p^n)$, where $n = 1$.

4.2 Transfer Function simplification

Long division on polynomials ($Z_5[x]$):

Script 4: Finite Field computation using Matlab function gfDeconv

```
%% Self Example 1
A = [4 1 4 1]
B = [1 0 1]
[q r]=gfdeconv(A,B,5)

%% output goes here
q =
     4     1
r =
     0

%% self created example 2
C = [3 2 3 1 3];
D = [1 2 1 2];
[q r]=gfdeconv(C,D,5)

%% self created example 3 and has been checked
aold = [1 1 0 1 1 0 0 1]
bold = [1 0 1 1 0 1 1]
factor1 = [1 1 1 0 1 0 0 0]
factor2 = [1 1 1 0 1 0 0]
anew = gfconv(aold,factor1)
bnew = gfconv(bold,factor2)
acheck = gfdeconv(anew,factor1)
bcheck = gfdeconv(bnew,factor2)
%% self created example 4 and created supported diagram
aold = [1 0 1]
bold = [1 0 1 1]
factor1 = [1 1 1]
factor2 = [1 1 1 0]
anew = gfconv(aold,factor1)
```

```
bnew = gfconv(bold, factor2)
acheck = gfdeconv(anew, factor1)
bcheck = gfdeconv(bnew, factor2)
```

Solving $T(D) = \frac{b(D)}{a(D)} = \frac{1+D+D^3+D^4}{1+D+D^5}$ Using finite Field Division

$$T(D) = \frac{b(D)}{a(D)} = \frac{1 + D + D^3 + D^4}{1 + D + D^5} \tag{7}$$

$$\begin{array}{r} \\ D^4 + D^3 + D + 1 \\ \\ \\ \\ \end{array} \overline{\begin{array}{r} D + 1 \\ \big) D^5 + 0D^4 + 0D^3 + 0D^2 + D + 1 \\ \underline{D^5 + \ \ D^4 + 0D^3 + \ \ D^2 + D} \\ D^4 + 0D^3 + D^2 + 1 \\ \underline{D^4 + D^3 + 0D^2 + D + 1} \\ D^3 + D^2 + D \end{array}}$$

$$a(D) = r_0(D)\dot{q}_1(D) = r_0(D) \times (D + 1) + (D^3 + D^2 + D)$$

$$r_1(D) = D^3 + D^2 + D \tag{8}$$

Computing: $r_2(D)$

$$\frac{r_0(D)}{r_1(D)} = \frac{1 + D + D^3 + D^4}{D^3 + D^2 + D}$$

$$\begin{array}{r} \\ D^3 + D^2 + D \\ \\ \\ \end{array} \overline{\begin{array}{r} D \\ \big) D^4 + D^3 + 0D^2 + D + 1 \\ \underline{D^4 + D^3 + D^2} \\ D^2 + D + 1 \end{array}}$$

$$r_2(D) = D^2 + D + 1 \tag{9}$$

Computing: $r_3(D)$

$$\frac{r_1(D)}{r_2(D)} = \frac{D^3 + D^2 + D}{D^2 + D + 1}$$

$$
\begin{array}{r}
D \\
D^2+D+1 \;\overline{)\,D^3+D^2+D+0} \\
\underline{D^3+D^2+D} \\
0
\end{array}
$$

$$r_3(D) = 0 \tag{10}$$

4.2.1 Dividing by GCD

$$\text{Final Answer } T(D) = \frac{\left(\frac{b(D)}{GCD}\right)}{\left(\frac{a(D)}{GCD}\right)} = \frac{1+D^2}{1+D^2+D^3}$$

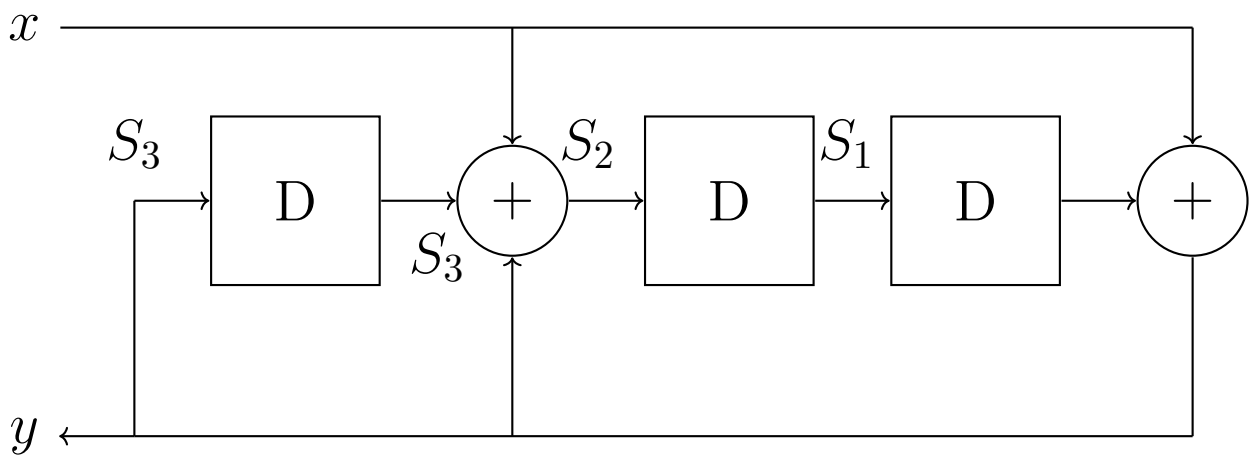
$$b_0 = 1 \quad b_1 = 0 \quad b_2 = 1$$

$$-a_0 = -0 = 0 \quad -a_1 = -1 = 1 \quad -a_2 = -1 = 1$$

Final Answer

$$\text{Final Answer } T(D) = \frac{\left(\frac{b(D)}{GCD}\right)}{\left(\frac{a(D)}{GCD}\right)} = \frac{1+D^2}{1+D^2+D^3} \tag{11}$$

4.2.2 Block Diagram



4.3 Putzer's algorithm

Website example: Putzer's algorithm as in Ceng 242 Slides.
Used OGF(Ordinary Generating Functions)

$$A = \begin{bmatrix} 0 & 1 \\ -2 & -3 \end{bmatrix}$$

$$A - \lambda I = \begin{bmatrix} -\lambda & 1 \\ -2 & -3-\lambda \end{bmatrix}$$

$$g(\lambda) = det(A - \lambda I) = det\left(\begin{bmatrix} -\lambda & 1 \\ -2 & -3-\lambda \end{bmatrix}\right) = \lambda^2 + 3\lambda + 2 \tag{12}$$

Factoring gives $(\lambda+2)(\lambda+1)$ Eigenvalues are $\lambda_1 = -1 \quad \lambda_2 = -2$ $M_0 = I$

$$\mu_1(t) = \lambda_1^t = (-1)^t \tag{13}$$

$$\mu_2(t) = \sum_{r=0}^{t-1} \lambda_2^{t-1-r} \cdot \mu_1(r) = \sum_{r=0}^{t-1} = (-2)^{t-1-r} \cdot (-1)^r$$

$$= (-2)^{t-1} \sum_{r=0}^{t-1} \left(\frac{1}{2}\right)^r = (-2)^{t-1} \cdot (2 - (2^{1-t}))$$

$$= -(-1)^t(2^t - 1) = (-1)^t - (-2)^t \tag{14}$$

See Page 13 for more info

$$A^t = \mu_1(t) \cdot I + \mu_2(t) \cdot M_1$$

$$= (-1)^t \cdot \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + ((-1)^t - (-2)^t) \cdot \begin{bmatrix} -1 & 1 \\ -2 & -2 \end{bmatrix}$$

$$A^t = \begin{bmatrix} (-2)^t) & (-1)^t - (-2)^t) \\ -2(-1)^t + (-2)^{t+1} & -(-1)^t + (-2)^{t+1} \end{bmatrix} \tag{15}$$

Linear MIMO machine equations

$$\begin{aligned} \boldsymbol{s}(t+1) &= \boldsymbol{A}\boldsymbol{s}(t) + \boldsymbol{B}\boldsymbol{x}(t), \\ \boldsymbol{y}(t) &= \boldsymbol{C}\boldsymbol{s}(t) + \boldsymbol{D}\boldsymbol{x}(t). \end{aligned}$$

Note that the equation for $\boldsymbol{s}(t)$ is the **first-order NHLR**, and it requires one initial condition, namely the *initial state* $\boldsymbol{s}(0)$. Thus, a linear MIMO machine is completely specified by $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$, $\boldsymbol{D}$, and $\boldsymbol{s}(0)$.

The machine's state and output at time $t > 0$ can be computed as follows:

$$\begin{aligned} \boldsymbol{s}(t) &= \boldsymbol{A}^t\boldsymbol{s}(0) + \textstyle\sum_{r=0}^{t-1} \boldsymbol{A}^{t-1-r}\boldsymbol{B}\boldsymbol{x}(r), \\ \boldsymbol{y}(t) &= \boldsymbol{C}\boldsymbol{A}^t\boldsymbol{s}(0) + \textstyle\sum_{r=0}^{t-1} \boldsymbol{C}\boldsymbol{A}^{t-1-r}\boldsymbol{B}\boldsymbol{x}(r) + \boldsymbol{D}\boldsymbol{x}(t). \end{aligned}$$

How do we calculate $\boldsymbol{A}^t$ efficiently?

Calculating $\boldsymbol{A}^t$

Problem:
Given a matrix $\boldsymbol{A} = [a_{ij}]_{k \times k}$ and some $t \in \mathbb{N}$, calculate $\boldsymbol{A}^t$, letting $\boldsymbol{A}^0 = \boldsymbol{I}$ where $\boldsymbol{I}$ denotes the **identity matrix**.

Solution:
Use the *characteristic polynomial* of **A** and the *Cayley-Hamilton theorem*.

The **characteristic equation** of $\boldsymbol{A}$ is defined as $\det(\boldsymbol{A} - \lambda \cdot \boldsymbol{I}) = 0$, where $\det(\boldsymbol{A} - \lambda \cdot \boldsymbol{I}) = \lambda^k + h_{k-1} \cdot \lambda^{k-1} + ... + h_1 \cdot \lambda + h_0 = g(\lambda)$. We call $g(\lambda)$ the **characteristic polynomial** of $\boldsymbol{A}$, whose coefficients $h_0, h_1, ..., h_{k-1}$ are obtained as a result of computing $\det(\boldsymbol{A} - \lambda \cdot \boldsymbol{I})$.

It can be shown that the **roots** of the characteristic equation $g(\lambda) = 0$ are the **eigenvalues** of $\boldsymbol{A}$.

Cayley-Hamilton theorem

Theorem (Cayley-Hamilton)

Every $k \times k$ matrix $\boldsymbol{A}$ satisfies its own characteristic equation, i.e.,

$$\boldsymbol{A}^k + h_{k-1} \cdot \boldsymbol{A}^{k-1} + ... + h_1 \cdot \boldsymbol{A} + h_0 \cdot \boldsymbol{I} = \boldsymbol{0},$$

where $\boldsymbol{0} = [0]_{k \times k}$, and $h_0, h_1, ..., h_{k-1}$ are coefficients of the characteristic polynomial $g(\lambda) = \det(\boldsymbol{A} - \lambda \cdot \boldsymbol{I}) = \lambda^k + h_{k-1} \cdot \lambda^{k-1} + ... + h_1 \cdot \lambda + h_0$.

Consequently, if $\lambda_1, \lambda_2, ..., \lambda_k$ are eigenvalues of $\boldsymbol{A}$, then:

$$\prod_{j=1}^{k} (\boldsymbol{A} - \lambda_j \cdot \boldsymbol{I}) = \boldsymbol{0}.$$

Important: The Cayley-Hamilton theorem tells us that every $k \times k$ matrix $\boldsymbol{A}^t$ can be represented as a **linear combination** of $\boldsymbol{A}^{k-1}$, $\boldsymbol{A}^{k-2}$, ..., $\boldsymbol{A}^2$, $\boldsymbol{A}$, and $\boldsymbol{I}$, where $t, k \in \mathbb{N}$ and $t \geq k$.

Putzer's algorithm

Putzer's algorithm calculates $\boldsymbol{A}^t$ at $t \geq k$. It applies the Cayley-Hamilton theorem to k eigenvalues of $\boldsymbol{A}$ (denoted by $\lambda_1, \lambda_2, ..., \lambda_k$) as follows:

1. Let $\boldsymbol{M}_0 \leftarrow \boldsymbol{I}$, and let $\boldsymbol{M}_1 \leftarrow \boldsymbol{A} - \lambda_1 \cdot \boldsymbol{I}$.

2. For each index $j = 2, 3, .., k-1$, calculate $\boldsymbol{M}_j = (\boldsymbol{A} - \lambda_j \cdot \boldsymbol{I}) \cdot \boldsymbol{M}_{j-1}$.
   - **Note:** For $j = k$, we have $\boldsymbol{M}_k = \boldsymbol{0}$, where $\boldsymbol{0} = [0]_{k \times k}$.

3. Given $t \geq k$, let $\mu_1(t) \leftarrow \lambda_1^t$, and let $\mu_j(t) \leftarrow \sum_{r=0}^{t-1} \lambda_j^{t-1-r} \cdot \mu_{j-1}(r)$ for each index $j = 2, 3, ..., k$.
   - **Note:** $\mu_j(t)$ satisfies the recursion $\mu_j(t+1) - \lambda_j \cdot \mu_j(t) = \mu_{j-1}(t)$.

4. Let $\boldsymbol{A}^t \leftarrow \sum_{j=1}^{k} \mu_j(t) \cdot \boldsymbol{M}_{j-1}$.

Example: applying Putzer's algorithm I

Let a linear MIMO machine over $\mathbb{Q}$ have the following 2×2 state matrix:

$$\boldsymbol{A} = \begin{bmatrix} 1 & 1 \\ -2 & 4 \end{bmatrix} \longrightarrow \boldsymbol{A} - \lambda \cdot \boldsymbol{I} = \begin{bmatrix} 1-\lambda & 1 \\ -2 & 4-\lambda \end{bmatrix}.$$

Its characteristic polynomial is $g(\lambda) = \det(\boldsymbol{A} - \lambda \cdot \boldsymbol{I}) = (\lambda - 3) \cdot (\lambda - 2)$. The equation $g(\lambda) = 0$ has two roots $\lambda_1 = 3$ and $\lambda_2 = 2$. Therefore,

$$\begin{aligned} \mu_1(t) &= \lambda_1^t = 3^t, \\ \mu_2(t) &= \textstyle\sum_{r=0}^{t-1} \lambda_2^{t-1-r} \cdot \mu_1(r) = 2^{t-1} \cdot \sum_{r=0}^{t-1} \left(\frac{3}{2}\right)^r \\ &= 2^{t-1} \cdot \frac{1 - \left(\frac{3}{2}\right)^t}{1 - \frac{3}{2}} = 3^t - 2^t. \end{aligned}$$

Useful formula:

$$\textstyle\sum_{r=0}^{t-1} c^r = 1 + c + c^2 + \ldots + c^{t-1} = \begin{cases} t, & c = 1; \\ \frac{1-c^t}{1-c}, & c \neq 1. \end{cases}$$

Example: applying Putzer's algorithm II

We also have:

$$\boldsymbol{M}_1 = \boldsymbol{A} - \lambda_1 \cdot \boldsymbol{I} = \begin{bmatrix} 1 & 1 \\ -2 & 4 \end{bmatrix} - \begin{bmatrix} 3 & 0 \\ 0 & 3 \end{bmatrix} = \begin{bmatrix} -2 & 1 \\ -2 & 1 \end{bmatrix}.$$

Consequently, we obtain the following closed-form expression for $\boldsymbol{A}^t$:

$$\begin{aligned} \boldsymbol{A}^t &= \mu_1(t) \cdot \boldsymbol{I} + \mu_2(t) \cdot \boldsymbol{M}_1 \\ &= 3^t \cdot \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + (3^t - 2^t) \cdot \begin{bmatrix} -2 & 1 \\ -2 & 1 \end{bmatrix} \\ &= \begin{bmatrix} 2^{t+1} - 3^t & 3^t - 2^t \\ 2^{t+1} - 2 \cdot 3^t & 2 \cdot 3^t - 2^t \end{bmatrix}. \end{aligned}$$

Another example: applying Putzer's algorithm I

Let a linear MIMO machine over $\mathbb{Q}$ have the following 3×3 state matrix:

$$\boldsymbol{A} = \begin{bmatrix} 4 & 1 & 2 \\ 0 & 2 & -4 \\ 0 & 1 & 6 \end{bmatrix} \longrightarrow \boldsymbol{A} - \lambda \cdot \boldsymbol{I} = \begin{bmatrix} 4-\lambda & 1 & 2 \\ 0 & 2-\lambda & -4 \\ 0 & 1 & 6-\lambda \end{bmatrix}.$$

Its characteristic polynomial is $g(\lambda) = \det(\boldsymbol{A} - \lambda \cdot \boldsymbol{I}) = (4-\lambda) \cdot (\lambda - 4)^2$. The equation $g(\lambda) = 0$ has three roots $\lambda_1 = \lambda_2 = \lambda_3 = 4$. Therefore,

$$\begin{aligned} \mu_1(t) &= \lambda_1^t = 4^t, \\ \mu_2(t) &= \textstyle\sum_{r=0}^{t-1} \lambda_2^{t-1-r} \cdot \mu_1(r) = \sum_{r=0}^{t-1} 4^{t-1-r} \cdot 4^r = t \cdot 4^{t-1}, \\ \mu_3(t) &= \textstyle\sum_{r=0}^{t-1} \lambda_3^{t-1-r} \cdot \mu_2(r) = \sum_{r=0}^{t-1} 4^{t-1-r} \cdot (r \cdot 4^{r-1}) = \frac{t^2-t}{2} \cdot 4^{t-2}. \end{aligned}$$

Another example: applying Putzer's algorithm II

We also have:

$$\boldsymbol{M}_1 = \boldsymbol{A} - \lambda_1 \cdot \boldsymbol{I} = \begin{bmatrix} 0 & 1 & 2 \\ 0 & -2 & -4 \\ 0 & 1 & 2 \end{bmatrix}, \; \boldsymbol{M}_2 = (\boldsymbol{A} - \lambda_2 \cdot \boldsymbol{I}) \cdot \boldsymbol{M}_1 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

Consequently, we obtain the following closed-form expression for $\boldsymbol{A}^t$:

$$\begin{aligned} \boldsymbol{A}^t &= \mu_1(t) \cdot \boldsymbol{I} + \mu_2(t) \cdot \boldsymbol{M}_1 + \mu_3(t) \cdot \boldsymbol{M}_2 \\ &= 4^t \cdot \boldsymbol{I} + t \cdot 4^{t-1} \cdot \boldsymbol{M}_1 + \tfrac{t^2-t}{2} \cdot 4^{t-2} \cdot \boldsymbol{M}_2 \\ &= \begin{bmatrix} 4^t & t \cdot 4^{t-1} & 2 \cdot t \cdot 4^{t-1} \\ 0 & 4^t - 2 \cdot t \cdot 4^{t-1} & -t \cdot 4^{t-1} \\ 0 & t \cdot 4^{t-1} & 4^t + 2 \cdot t \cdot 4^{t-1} \end{bmatrix}. \end{aligned}$$

4.4 Difference-equation/state-variable models

$$
\begin{aligned}
s(t+1) &= As(t) + Bx(t) \\
y(t) &= Cs(t) + Dx(t) \\
H(z^{-1}) &= C(z \cdot I - A)^{-1}B + D
\end{aligned}
$$

$$s(t+1) = \begin{bmatrix} 0 & 1 \\ -4 & 4 \end{bmatrix} \cdot s(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \cdot x(t) \quad (16)$$

$$y(t) = [-7 \quad 8] \cdot s(t) + [2] \cdot x(t) \quad (17)$$

$$(z \cdot I - A)^{-1} = \frac{1}{\det(zI - A)} \cdot \mathrm{adj}(zI - A) = \frac{1}{z^2 - 4z + 4} \cdot \begin{bmatrix} z-4 & 1 \\ -4 & z \end{bmatrix}$$

$$
\begin{aligned}
H(z^{-1}) &= \frac{1}{z^2 - 4z + 4} \cdot [-7 \quad 8] \cdot \begin{bmatrix} z-4 & 1 \\ -4 & z \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 1 \end{bmatrix} + [2] \\
&= \left[\frac{1 + 2z^2}{z^2 - 4z + 4}\right] = \left[\frac{z^{-2} + 2}{1 - 4z^{-1} + 4z^{-2}}\right] \qquad (18)
\end{aligned}
$$

$$s(t+1) = \begin{bmatrix} 0 & -4 \\ 1 & 4 \end{bmatrix} \cdot s(t) + \begin{bmatrix} -7 \\ 8 \end{bmatrix} \cdot x(t)$$

$$y(t) = [0 \quad 1] \cdot s(t) + [2] \cdot x(t)$$

$$(z \cdot I - A)^{-1} = \frac{1}{\det(zI - A)} \cdot \mathrm{adj}(zI - A) = \frac{1}{z^2 - 4z + 4} \cdot \begin{bmatrix} z-4 & -4 \\ 1 & z \end{bmatrix}$$

$$
\begin{aligned}
H(z^{-1}) &= \frac{1}{z^2 - 4z + 4} \cdot [0 \quad 1] \cdot \begin{bmatrix} z-4 & -4 \\ 1 & z \end{bmatrix} \cdot \begin{bmatrix} -7 \\ 8 \end{bmatrix} + [2] \qquad (19) \\
&= \left[\frac{1 + 2z^2}{z^2 - 4z + 4}\right] = \left[\frac{z^{-2} + 2}{1 - 4z^{-1} + 4z^{-2}}\right] \qquad (20)
\end{aligned}
$$

State-variable model

Any kth-order discrete-time linear system (SISO or MIMO) over $\mathbb{C}$ can be specified using a **state-variable model** (involving matrices $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$, $\boldsymbol{D}$):

$$\begin{aligned} \boldsymbol{s}(t+1) &= \boldsymbol{A}\boldsymbol{s}(t) + \boldsymbol{B}\boldsymbol{x}(t), \\ \boldsymbol{y}(t) &= \boldsymbol{C}\boldsymbol{s}(t) + \boldsymbol{D}\boldsymbol{x}(t), \end{aligned}$$

where $\mathbf{s}(t)$ is a k-element state vector, $\boldsymbol{x}(t)$ is an input vector, and $\boldsymbol{y}(t)$ is an output vector.

The state-variable model has two special representations for SISO systems: **controllable canonical form** and **observable canonical form**.

State-variable model: controllable canonical form

Difference-Equation Model: $\sum_{j=0}^{k} a_j \cdot y(t-j) = \sum_{j=0}^{k} b_j \cdot x(t-j)$

State-Variable Model — Controllable Canonical Form:

$$\boldsymbol{A} = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 1 \\ -a_k & -a_{k-1} & -a_{k-2} & \cdots & -a_2 & -a_1 \end{bmatrix}, \qquad \boldsymbol{B} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ 1 \end{bmatrix},$$

$$\boldsymbol{C} = \begin{bmatrix} b_k - b_0 \cdot a_k & b_{k-1} - b_0 \cdot a_{k-1} & \cdots & b_2 - b_0 \cdot a_2 & b_1 - b_0 \cdot a_1 \end{bmatrix},$$

$$\boldsymbol{D} = [b_0].$$

Note: $\det(\boldsymbol{A} - \lambda \cdot \boldsymbol{I}) = (-1)^k(\lambda^k + a_1 \cdot \lambda^{k-1} + ... + a_{k-1} \cdot \lambda + a_k)$.

State-variable model: observable canonical form

Difference-Equation Model: $\sum_{j=0}^{k} a_j \cdot y(t-j) = \sum_{j=0}^{k} b_j \cdot x(t-j)$

State-Variable Model — Observable Canonical Form:

$$\boldsymbol{A} = \begin{bmatrix} 0 & 0 & \cdots & 0 & 0 & -a_k \\ 1 & 0 & \cdots & 0 & 0 & -a_{k-1} \\ 0 & 1 & \cdots & 0 & 0 & -a_{k-2} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 & -a_2 \\ 0 & 0 & \cdots & 0 & 1 & -a_1 \end{bmatrix}, \qquad \boldsymbol{B} = \begin{bmatrix} b_k - b_0 \cdot a_k \\ b_{k-1} - b_0 \cdot a_{k-1} \\ b_{k-2} - b_0 \cdot a_{k-2} \\ \vdots \\ b_2 - b_0 \cdot a_2 \\ b_1 - b_0 \cdot a_1 \end{bmatrix},$$

$$\boldsymbol{C} = \begin{bmatrix} 0 & 0 & \cdots & 0 & 0 & 1 \end{bmatrix},$$

$$\boldsymbol{D} = [b_0].$$

Note: $\det(\boldsymbol{A} - \lambda \cdot \boldsymbol{I}) = (-1)^k(\lambda^k + a_1 \cdot \lambda^{k-1} + ... + a_{k-1} \cdot \lambda + a_k)$.

5 Solving HLR/NHLR

All solved Problems in this section are located here.
Skip to Page 37 for examples of HLR and NHLR

Linear recursions with constant coefficients

Let F be a field. Given k initial values $c_0, c_1, ..., c_{k-1} \in F$ and coefficients $a_1, a_2, ..., a_k \in F$, where $a_k \neq 0$, we define a **kth-order homogeneous linear recursion (HLR)** as follows:

$$\begin{cases} u_0 = c_0, \quad u_1 = c_1, \quad u_2 = c_2, \quad ..., \quad u_{k-1} = c_{k-1} & \text{(initial conditions);} \\ u_{n+k} + a_1 \cdot u_{n+k-1} + ... + a_{k-1} \cdot u_{n+1} + a_k \cdot u_n = 0 & \text{(for every } n \in \mathbb{Z}_+\text{).} \end{cases}$$

Similarly, a **kth-order nonhomogeneous linear recursion (NHLR)** is defined as follows:

$$\begin{cases} u_0 = c_0, \quad u_1 = c_1, \quad ..., \quad u_{k-1} = c_{k-1} & \text{(initial conditions);} \\ u_{n+k} + a_1 \cdot u_{n+k-1} + ... + a_k \cdot u_n = g(n) & \text{(for every } n \in \mathbb{Z}_+\text{),} \end{cases}$$

where function $g : \mathbb{Z}_+ \mapsto F$ maps every argument $n \in \mathbb{Z}_+$ to some unique value $g(n) \in F$.

5.1 Textbook Recursion Questions

Test Yourself

1. A recursive definition for a sequence consists of a _____ and _____.
2. A recurrence relation is an equation that defines each later term of a sequence by reference to _____ in the sequence.
3. Initial conditions for a recursive definition of a sequence consist of one or more of the _____ of the sequence.
4. To solve a problem recursively means to divide the problem into smaller subproblems of the same type as the initial problem, to suppose _____, and to figure out how to use the supposition to _____.
5. A crucial step for solving a problem recursively is to define a _____ in terms of which the recurrence relation and initial conditions can be specified.

Exercise Set 5.6

Find the first four terms of each of the recursively defined sequences in 1–8.

1. $a_k = 2a_{k-1} + k$, for all integers $k \geq 2$
$a_1 = 1$

2. $b_k = b_{k-1} + 3k$, for all integers $k \geq 2$
$b_1 = 1$

3. $c_k = k(c_{k-1})^2$, for all integers $k \geq 1$
$c_0 = 1$

4. $d_k = k(d_{k-1})^2$, for all integers $k \geq 1$
$d_0 = 3$

5. $s_k = s_{k-1} + 2s_{k-2}$, for all integers $k \geq 2$
$s_0 = 1,\ s_1 = 1$

6. $t_k = t_{k-1} + 2t_{k-2}$, for all integers $k \geq 2$
$t_0 = -1,\ t_1 = 2$

7. $u_k = ku_{k-1} - u_{k-2}$, for all integers $k \geq 3$
$u_1 = 1,\ u_2 = 1$

8. $v_k = v_{k-1} + v_{k-2} + 1$, for all integers $k \geq 3$
$v_1 = 1,\ v_2 = 3$

9. Let $a_0, a_1, a_2, \ldots$ be defined by the formula $a_n = 3n + 1$, for all integers $n \geq 0$. Show that this sequence satisfies the recurrence relation $a_k = a_{k-1} + 3$, for all integers $k \geq 1$.

10. Let $b_0, b_1, b_2, \ldots$ be defined by the formula $b_n = 4^n$, for all integers $n \geq 0$. Show that this sequence satisfies the recurrence relation $b_k = 4b_{k-1}$, for all integers $k \geq 1$.

11. Let $c_0, c_1, c_2, \ldots$ be defined by the formula $c_n = 2^n - 1$ for all integers $n \geq 0$. Show that this sequence satisfies the recurrence relation

$$c_k = 2c_{k-1} + 1.$$

12. Let $s_0, s_1, s_2, \ldots$ be defined by the formula $s_n = \dfrac{(-1)^n}{n!}$ for all integers $n \geq 0$. Show that this sequence satisfies the recurrence relation

$$s_k = \frac{-s_{k-1}}{k}.$$

13. Let $t_0, t_1, t_2, \ldots$ be defined by the formula $t_n = 2 + n$ for all integers $n \geq 0$. Show that this sequence satisfies the recurrence relation

$$t_k = 2t_{k-1} - t_{k-2}.$$

14. Let $d_0, d_1, d_2, \ldots$ be defined by the formula $d_n = 3^n - 2^n$ for all integers $n \geq 0$. Show that this sequence satisfies the recurrence relation

$$d_k = 5d_{k-1} - 6d_{k-2}.$$

H 15. For the sequence of Catalan numbers defined in Example 5.6.4, prove that for all integers $n \geq 1$,

$$C_n = \frac{1}{4n+2}\binom{2n+2}{n+1}.$$

16. Use the recurrence relation and values for the Tower of Hanoi sequence $m_1, m_2, m_3, \ldots$ discussed in Example 5.6.5 to compute m_7 and m_8.

17. *Tower of Hanoi with Adjacency Requirement*: Suppose that in addition to the requirement that they never move a larger disk on top of a smaller one, the priests who move the disks of the Tower of Hanoi are also allowed only to move disks one by one from one pole to an *adjacent* pole. Assume poles A and C are at the two ends of the row and pole B is in the middle. Let

$$a_n = \left[\begin{array}{l}\text{the minimum number of moves}\\ \text{needed to transfer a tower of } n\\ \text{disks from pole } A \text{ to pole } C\end{array}\right].$$

a. Find a_1, a_2, and a_3. b. Find a_4.
c. Find a recurrence relation for $a_1, a_2, a_3, \ldots$.

18. *Tower of Hanoi with Adjacency Requirement*: Suppose the same situation as in exercise 17. Let

$$b_n = \left[\begin{array}{l}\text{the minimum number of moves}\\ \text{needed to transfer a tower of } n\\ \text{disks from pole } A \text{ to pole } B\end{array}\right].$$

a. Find b_1, b_2, and b_3. **b.** Find b_4.

5.2 Textbook Recursion Questions 2

4. Let $b_0, b_1, b_2, \ldots$ be the sequence defined by the explicit formula

$$b_n = C \cdot 3^n + D(-2)^n \quad \text{for all integers } n \geq 0,$$

where C and D are real numbers.

a. Find C and D so that $b_0 = 0$ and $b_1 = 5$. What is b_2 in this case?

b. Find C and D so that $b_0 = 3$ and $b_1 = 4$. What is b_2 in this case?

5. Let $a_0, a_1, a_2, \ldots$ be the sequence defined by the explicit formula

$$a_n = C \cdot 2^n + D \quad \text{for all integers } n \geq 0,$$

where C and D are real numbers. Show that for any choice of C and D,

$$a_k = 3a_{k-1} - 2a_{k-2} \quad \text{for all integers } k \geq 2.$$

6. Let $b_0, b_1, b_2, \ldots$ be the sequence defined by the explicit formula

$$b_n = C \cdot 3^n + D(-2)^n \quad \text{for all integers } n \geq 0,$$

where C and D are real numbers. Show that for any choice of C and D,

$$b_k = b_{k-1} + 6b_{k-2} \quad \text{for all integers } k \geq 2.$$

7. Solve the system of equations in Example 5.8.4 to obtain

$$C = \frac{1+\sqrt{5}}{2\sqrt{5}} \text{ and } D = \frac{-(1-\sqrt{5})}{2\sqrt{5}}.$$

In each of 8–10: (a) suppose a sequence of the form $1.t.t^2.t^3 \ldots t^n \ldots$ where $t \neq 0$, satisfies the given recurrence relation (but not necessarily the initial conditions), and find all possible values of t: (b) suppose a sequence satisfies the given initial conditions as well as the recurrence relation, and find an explicit formula for the sequence.

8. $a_k = 2a_{k-1} + 3a_{k-2}$, for all integers $k \geq 2$
$a_0 = 1, a_1 = 2$

9. $b_k = 7b_{k-1} - 10b_{k-2}$, for all integers $k \geq 2$
$b_0 = 2, b_1 = 2$

10. $c_k = c_{k-1} + 6c_{k-2}$, for all integers $k \geq 2$
$c_0 = 0, c_1 = 3$

In each of 11–16 suppose a sequence satisfies the given recurrence relation and initial conditions. Find an explicit formula for the sequence.

11. $d_k = 4d_{k-2}$, for all integers $k \geq 2$
$d_0 = 1,\ d_1 = -1$

12. $e_k = 9e_{k-2}$, for all integers $k \geq 2$
$e_0 = 0,\ e_1 = 2$

13. $r_k = 2r_{k-1} - r_{k-2}$, for all integers $k \geq 2$
$r_0 = 1,\ r_1 = 4$

14. $s_k = -4s_{k-1} - 4s_{k-2}$, for all integers $k \geq 2$
$s_0 = 0,\ s_1 = -1$

15. $t_k = 6t_{k-1} - 9t_{k-2}$, for all integers $k \geq 2$
$t_0 = 1,\ t_1 = 3$

***H* 16.** $s_k = 2s_{k-1} + 2s_{k-2}$, for all integers $k \geq 2$
$s_0 = 1,\ s_1 = 3$

17. Find an explicit formula for the sequence of exercise 39 in Section 5.6

18. Suppose that the sequences $s_0, s_1, s_2, \ldots$ and $t_0, t_1, t_2, \ldots$ both satisfy the same second-order linear homogeneous recurrence relation with constant coefficients:

$$s_k = 5s_{k-1} - 4s_{k-2} \quad \text{for all integers } k \geq 2,$$
$$t_k = 5t_{k-1} - 4t_{k-2} \quad \text{for all integers } k \geq 2.$$

Show that the sequence $2s_0 + 3t_0, 2s_1 + 3t_1, 2s_2 + 3t_2, \ldots$ also satisfies the same relation. In other words, show that

$$2s_k + 3t_k = 5(2s_{k-1} + 3t_{k-1}) - 4(2s_{k-2} + 3t_{k-2})$$

for all integers $k \geq 2$. Do *not* use Lemma 5.8.2.

19. Show that if r, s, a_0, and a_1 are numbers with $r \neq s$, then there exist unique numbers C and D so that

$$C + D = a_0$$
$$Cr + Ds = a_1.$$

20. Show that if r is a nonzero real number, k and m are distinct integers, and a_k and a_m are any real numbers, then there exist unique real numbers C and D so that

$$Cr^k + kDr^k = a_k$$
$$Cr^m + lDr^m = a_m.$$

***H* 21.** Prove Theorem 5.8.5 for the case where the values of C and D are determined by a_0 and a_1.

Exercises 22 and 23 are intended for students who are familiar with complex numbers.

22. Find an explicit formula for a sequence $a_0, a_1, a_2, \ldots$ that satisfies

$$a_k = 2a_{k-1} - 2a_{k-2} \quad \text{for all integers } k \geq 2$$

with initial conditions $a_0 = 1$ and $a_1 = 2$.

23. Find an explicit formula for a sequence $b_0, b_1, b_2, \ldots$ that satisfies

$$b_k = 2b_{k-1} - 5b_{k-2} \quad \text{for all integers } k \geq 2$$

with initial conditions $b_0 = 1$ and $b_1 = 1$.

5.3 Introduction to OGF(Ordinary Generating Functions)

Polynomials are a special case of ordinary generating functions corresponding to finite sequences or equivalently sequences that vanish after a certain point. Click to go to Wikipedia and get more info.

Ordinary generating functions

Since $U(x) = \sum_{n=0}^{\infty} u_n \cdot x^n$ is simply an expression representing an *infinite sequence* $(u_0, u_1, ..., u_n, ...)$ of commutative ring elements, it is also called the **ordinary generating function (OGF)** for that sequence. Below are two examples illustrating the OGF concept.

Consider $U(x) = \sum_{n=0}^{\infty} u_n \cdot x^n \in \mathbb{R}[[x]]$, where $u_n = 1/n!$. Our $U(x)$ is the OGF for the sequence $(\frac{1}{0!}, \frac{1}{1!}, ..., \frac{1}{n!}, ...)$. Moreover, $U(x)$ can be viewed as the series expansion of the **exponential function** $\exp(x)$, where $x \in \mathbb{R}$.

Consider a polynomial $x + 1 \in \mathbb{Z}[x]$, and let $m > 1$ be some integer. We know that $(x+1)^m = \sum_{n=0}^{m} \binom{m}{n} \cdot x^n$, where $\binom{m}{n} = \frac{m!}{n!(m-n)!}$ are so-called **binomial coefficients**. Therefore, $U(x) = (x+1)^m$ can be viewed as the OGF for the sequence $(\binom{m}{0}, \binom{m}{1}, ..., \binom{m}{m}, 0, 0, 0, ...)$.

5.4 OGF Tables

These tables are extremely useful in solving problems

More examples of sequences and their OGFs I

Sequence $(u_0, u_1, ..., u_n, ...)$, where c is some constant	Expression for u_n	OGF $U(x)$
$c, c, c, c, c, ...$	$u_n = c$	$\frac{c}{1-x}$
$c, -c, c, -c, c, ...$	$u_n = (-1)^n c$	$\frac{c}{1+x}$
$c, 0, c, 0, c, ...$	$u_n = \frac{(-1)^n+1}{2} c$	$\frac{c}{1-x^2}$
$c, 2c, 3c, 4c, 5c, ...$	$u_n = (n+1)c$	$\frac{c}{(1-x)^2}$
$0, c, 2c, 3c, 4c, ...$	$u_n = nc$	$\frac{c \cdot x}{(1-x)^2}$
$1, c, c^2, c^3, c^4, ...$	$u_n = c^n$	$\frac{1}{1-c \cdot x}$
$0, c, 2c^2, 3c^3, 4c^4, ...$	$u_n = nc^n$	$\frac{c \cdot x}{(1-c \cdot x)^2}$
$1, 2c, 3c^2, 4c^3, 5c^4, ...$	$u_n = (n+1)c^n$	$\frac{1}{(1-c \cdot x)^2}$
$1, -2c, 3c^2, -4c^3, 5c^4, ...$	$u_n = (-1)^n(n+1)c^n$	$\frac{1}{(1+c \cdot x)^2}$

Examples of HLR and NHLR

Consider the second-order HLR $u_{n+2} - 5 \cdot u_{n+1} + 6 \cdot u_n = 0$ over $\mathbb{Q}$, with $u_0 = 0$ and $u_1 = 1$ (i.e., we have $a_1 = -5$, $a_2 = 6$, $c_0 = 0$, $c_1 = 1$). This recursion specifies a certain sequence $(u_0, u_1, ..., u_n, ...)$, whose OGF is as follows:

$$U(x) = \frac{x}{1 - 5 \cdot x + 6 \cdot x^2} \quad \longrightarrow \quad u_n = -2^n + 3^n.$$

Consider the following NHLR over $\mathbb{Q}$: $u_{n+2} - 3 \cdot u_{n+1} + 2 \cdot u_n = n$, with $u_0 = u_1 = 1$. The corresponding OGF is given by

$$U(x) = \frac{x^3}{(1-x)^2 \cdot (1 - 3 \cdot x + 2 \cdot x^2)} + \frac{1 - 2 \cdot x}{1 - 3 \cdot x + 2 \cdot x^2},$$

which yields $u_n = 2^n - \frac{n^2+n}{2}$.

Question: Given some NHLR or HLR, how do we find its OGF $U(x)$ and the corresponding expression for u_n?

Alternative method for solving NHLRs I

A general NHLR solution can be written as $u_n = u_n^{(h)} + u_n^{(p)}$, where $u_n^{(h)}$ is the **homogeneous solution** of $u_{n+k} + a_1 \cdot u_{n+k-1} + ... + a_k \cdot u_n = 0$, and $u_n^{(p)}$ is a **particular solution** of $u_{n+k} + a_1 \cdot u_{n+k-1} + ... + a_k \cdot u_n = g(n)$.

We can obtain $u_n^{(h)}$ using the **method of charactestic equations**, to be covered later. To determine $u_n^{(p)}$, we need to examine *trial solutions* that depend on the nature of $g(n)$.

Special Case of $g(n)$, where $m \in \mathbb{N}$	Trial Solution $u_n^{(p)}$, where $A_0, A_1, ..., A_m$ are constants
c (constant)	A (constant)
c^n	$A_0 \cdot c^n$
$n^m c$	$(A_0 + nA_1 + ... + n^m A_m) \cdot c$
$n^m c^n$	$(A_0 + nA_1 + ... + n^m A_m) \cdot c^n$

Alternative method for solving NHLRs II

Rule: If any term of $u_n^{(p)}$ also appears in $u_n^{(h)}$, then we must use $n^z u_n^{(p)}$ as our trial solution, where $z \in \mathbb{Z}_+$ is the smallest positive integer, such that $n^z u_n^{(p)}$ and $u_n^{(h)}$ have *no terms in common*.

Example: Consider NHLR $u_{n+2} - 4 \cdot u_{n+1} + 4 \cdot u_n = 5 \cdot 4^n + 3 \cdot 2^n$, whose homogeneous solution is $u_n^{(h)} = A_0 \cdot 2^n + nA_1 \cdot 2^n$. Given $g_1(n) = 5 \cdot 4^n$, we obtain $u_n^{(p)_1} = B \cdot 4^n$. We also have $u_n^{(p)_2} = C \cdot 2^n$ for $g_2(n) = 3 \cdot 2^n$, but we must use $n^2(C \cdot 2^n)$, because $C \cdot 2^n$ or $n(C \cdot 2^n)$ would be in common with $A_0 \cdot 2^n$ or $nA_1 \cdot 2^n$ in $u_n^{(h)}$. Thus, $u_n^{(p)} = u_n^{(p)_1} + u_n^{(p)_2} = B \cdot 4^n + n^2 C \cdot 2^n$.

Example

Consider NHLR $u_{n+2} - 3 \cdot u_{n+1} + 2 \cdot u_n = n$ over $\mathbb{Q}$, with $u_0 = u_1 = 1$. Its *homogeneous solution* is $u_n^{(h)} = A \cdot 2^n + B$, and its trial *particular solution* is in the form $u_n^{(p)} = n(C_0 + nC_1)$, where n is present to ensure that C_0 is no longer in common with B in $u_n^{(h)}$. Thus,

$$u_n = u_n^{(h)} + u_n^{(p)} = A \cdot 2^n + B + nC_0 + n^2 C_1.$$

To determine unknown coefficients A, B, C_0, and C_1, we can use the initial conditions ($u_0 = 1$ and $u_1 = 1$), as well as the values of $u_2 = 1$ and $u_3 = 2$ produced by our NHLR for $n = 0$ ($u_2 - 3 \cdot u_1 + 2 \cdot u_0 = 0 \longrightarrow u_2 = 1$) and $n = 1$ ($u_3 - 3 \cdot u_2 + 2 \cdot u_1 = 1 \longrightarrow u_3 = 2$).

$$\begin{cases} u_0 = 1 = A + B, & \text{(initial condition)} \\ u_1 = 1 = 2 \cdot A + B + C_0 + C_1, & \text{(initial condition)} \\ u_2 = 1 = 4 \cdot A + B + 2C_0 + 4C_1, & \text{(recursion: } n = 0\text{)} \\ u_3 = 2 = 8 \cdot A + B + 3C_0 + 9C_1. & \text{(recursion: } n = 1\text{)} \end{cases}$$

Hence, $A = 1$, $B = 0$, and $C_0 = C_1 = -\frac{1}{2}$, which yields $u_n = 2^n - \frac{n^2+n}{2}$.

Method of characteristic equations

Given some HLR $u_{n+k} + a_1 \cdot u_{n+k-1} + ... + a_{k-1} \cdot u_{n+1} + a_k \cdot u_n = 0$, we define its **characteristic polynomial** as shown below, where $a_0 = 1$:

$$t(y) = \sum_{j=0}^{k} a_{k-j} \cdot y^j = y^k + a_1 \cdot y^{k-1} + ... + a_{k-1} \cdot y + a_k.$$

Observe that we can obtain $a(x)$ from $t(y)$ by replacing y^j with x^{k-j} for every $j \in \{0, 1, ..., k\}$.

When the **characteristic equation** $t(y) = 0$ has exactly k roots, we can obtain the closed-form expression for u_n directly, without using its OGF. For NHLRs, such roots lead to a closed-form expression for $u_n^{(h)}$.

Note: If $a_1, a_2, ..., a_k$ are complex numbers, i.e. $t(y) \in \mathbb{C}[y]$, the equation $t(y) = 0$ is guaranteed to have exactly k roots in $\mathbb{C}[y]$. This statement is also known as the "*Fundamental Theorem of Algebra*".

Characteristic roots

Theorem (Characteristic Roots)

Let $t(y)$ be a characteristic polynomial of some kth-order HLR. Suppose the equation $t(y) = 0$ has distinct roots $\lambda_1, \lambda_2, ..., \lambda_s$ with multiplicities $m_1, m_2, ..., m_s$. If $\sum_{j=1}^{s} m_j = k$, then the closed-form expression for our HLR-defined sequence $(u_0, u_1, ..., u_n, ...)$ is $u_n = \sum_{j=1}^{s} p_j(n) \cdot \lambda_j^n$, where $p_j(n) = C_{j,0} + nC_{j,1} + ... + n^{m_j-1} C_{j,m_j-1}$, and $C_{j,0}, C_{j,1}, ..., C_{j,m_j-1}$ are appropriate constants. Note that $n^m C$ means $\underbrace{C + C + ... + C}_{n^m \text{ terms}}$.

The above theorem is applicable whenever the characteristic equation of our kth-order HLR has exactly k roots, which is always the case for the field $\mathbb{C}$. When working in some other field (e.g., $\mathbb{R}$, $\mathbb{Q}$, $\mathbb{Z}_p$), we may still use this theorem, provided that our characteristic equation has exactly k roots. Otherwise, we have to work out partial fraction decomposition of the HLR's OGF.

Example I

Consider $u_{n+2} - 5 \cdot u_{n+1} + 6 \cdot u_n = 0$ over $\mathbb{Q}$ (i.e., $k = 2$); its characteristic polynomial is $t(y) = y^2 - 5 \cdot y + 6$. The equation $t(y) = 0$ has two distinct roots $\lambda_1 = 2$ and $\lambda_2 = 3$ with multiplicities $m_1 = 1$ and $m_2 = 1$. Since we have $\sum_{j=1}^{2} m_j = 2 = k$, we can use λ_1 and λ_2 as our characteristic roots:

$$u_n = p_1(n) \cdot 2^n + p_2(n) \cdot 3^n,$$

where $p_1(n) = C_{1,0}$ and $p_2(n) = C_{2,0}$.

Suppose the initial conditions are $u_0 = 0$ and $u_1 = 1$. Then, we obtain the following system of two equations:

$$\begin{array}{rcl} u_0 = 0 & = & p_1(0) \cdot 2^0 + p_2(0) \cdot 3^0 = C_{1,0} + C_{2,0}, \\ u_1 = 1 & = & p_1(1) \cdot 2^1 + p_2(1) \cdot 3^1 = C_{1,0} \cdot 2 + C_{2,0} \cdot 3. \end{array}$$

The solution is $C_{1,0} = -1$ and $C_{2,0} = 1$. Therefore, $u_n = -2^n + 3^n$.

Example II

Let us attempt to solve our previous HLR using the method of generating functions. We multiply both sides by x^{n+2} and rewrite:

$$\sum_{n=0}^{\infty} u_{n+2} \cdot x^{n+2} - 5 \cdot \sum_{n=0}^{\infty} u_{n+1} \cdot x^{n+2} + 6 \cdot \sum_{n=0}^{\infty} u_n \cdot x^{n+2} = 0.$$

Next, we need to bring $U(x) = \sum_{n=0}^{\infty} u_n \cdot x^n$ into play:

- $\sum_{n=0}^{\infty} u_{n+2} \cdot x^{n+2} = U(x) - u_0 - u_1 \cdot x = U(x) - x$,
- $5 \cdot \sum_{n=0}^{\infty} u_{n+1} \cdot x^{n+2} = 5 \cdot x \cdot (U(x) - u_0) = 5 \cdot x \cdot U(x)$,
- $6 \cdot \sum_{n=0}^{\infty} u_n \cdot x^{n+2} = 6 \cdot x^2 \cdot U(x)$.

Putting it all together, we obtain:

$$U(x) - x - 5 \cdot x \cdot U(x) + 6 \cdot x^2 \cdot U(x) = 0 \longrightarrow U(x) = \frac{x}{1 - 5 \cdot x + 6 \cdot x^2}.$$

Hence,

$$U(x) = \frac{x}{(1 - 2 \cdot x) \cdot (1 - 3 \cdot x)} = \frac{-1}{1 - 2 \cdot x} + \frac{1}{1 - 3 \cdot x} \longrightarrow u_n = -2^n + 3^n.$$

5.5 Example Problems

Problem set 1 located Here.
Problem set 2 located Here at pg 100.

5.5.1 Fibonacci numbers

The Fibonacci numbers are generated by setting $F_0 = 0$, $F_1 = 1$, and then using the recursive formula $f_n = f_{n-1} + f_{n-2}$. Find a general formula for the Fibonacci sequence. It may be better to solve this problem by relying upon the characteristic equation.

$$
\begin{aligned}
f_{n+2} - f_{n+1} - f_n &= 0 \\
\sum_{n=0}^{\infty} f_{n+2} \cdot x^{n+2} - \sum_{n=0}^{\infty} f_{n+1} \cdot x^{n+2} - \sum_{n=0}^{\infty} f_n \cdot x^{n+2} &= 0 \\
(F(x) - F_0 - F_1) - x(F(x) - F_0) - x^2 F(x) &= 0 \\
F(x) - xF(x) - x^2 F(x) &= 1 \\
F(x)[-x^2 + -x + 1] &= 1 \\
F(x) &= \frac{1}{[-x^2 - x + 1]} \\
F(x) &= \frac{1}{\left(\frac{1+\sqrt{5}}{2} \cdot x\right)\left(\frac{1-\sqrt{5}}{2} \cdot x\right)} \\
\text{Simplifing} \mapsto \quad &= \frac{1}{\frac{1}{2}(1+\sqrt{5} \cdot x)\frac{1}{2}(1-\sqrt{5} \cdot x)} \\
\text{Applying OGF} \mapsto \quad f_n &= \frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^n - \frac{1}{\sqrt{5}}\left(\frac{1-\sqrt{5}}{2}\right)^n
\end{aligned}
$$

$$\boxed{f_n = \frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^n - \frac{1}{\sqrt{5}}\left(\frac{1-\sqrt{5}}{2}\right)^n}$$

5.5.2 Towers Of Hanoi

More information on Wikipedia. The objective of the puzzle is to move the

entire stack to another rod, obeying the following simple rules:

1. Only one disk can be moved at a time.
2. Each move consists of taking the upper disk from one of the stacks and placing it on top of another stack i.e. a disk can only be moved if it is the uppermost disk on a stack.
3. No disk may be placed on top of a smaller disk.

n is the number of disks.

Solution Over Here.

$$\begin{cases} h_0 = 0 \text{ initial condition} \\ h_n = 2h_{n-1} + 1 \text{ recurrence} \end{cases}$$

$$\begin{aligned} h_n x^n &= 2h_{n-1}x^n + x^n \\ \sum_{n=1}^{\infty} h_n \cdot x^n &= \sum_{n=1}^{\infty} h_{n-1} \cdot x^n + \sum_{n=1}^{\infty} x^n \\ \sum_{n=1}^{\infty} h_n \cdot x^n &= 2x \sum_{n=1}^{\infty} h_{n-1} \cdot x^{n-1} + \sum_{n=1}^{\infty} x^n \\ H(x) - h_0 &= 2xH(x) + \frac{x}{1-x} \\ H(x)[1-2x] &= \frac{x}{1-x} \\ H(x) = \frac{x}{(1-x)[1-2x]} &= \frac{1}{1-2x} - \frac{1}{1-x} \\ h_n &= 2^n - 1 \end{aligned}$$

TOWERS OF HANOI RECURSION PROBLEM SOLUTION

$$h_n = 2^n - 1 \tag{21}$$

TOWERS OF HANOI RECURSION PROBLEM SOLUTION

$$\begin{cases} h_0 = 0 \text{ initial condition} \\ h_n = 2h_{n-1} + 1 \text{ recurrence} \end{cases}$$

$$h_n = 2^n - 1 \tag{22}$$

More information on Wikipedia. The objective of the puzzle is to move the entire stack to another rod, obeying the following simple rules:

1. Only one disk can be moved at a time.
2. Each move consists of taking the upper disk from one of the stacks and placing it on top of another stack i.e. a disk can only be moved if it is the uppermost disk on a stack.
3. No disk may be placed on top of a smaller disk.

5.5.3 Solution

$$\begin{aligned} h_n x^n &= 2h_{n-1}x^n + x^n \\ \sum_{n=1}^{\infty} h_n \cdot x^n &= \sum_{n=1}^{\infty} h_{n-1} \cdot x^n + \sum_{n=1}^{\infty} x^n \\ \sum_{n=1}^{\infty} h_n \cdot x^n &= 2x \sum_{n=1}^{\infty} h_{n-1} \cdot x^{n-1} + \sum_{n=1}^{\infty} x^n \\ H(x) - h_0 &= 2xH(x) + \frac{x}{1-x} \\ H(x)[1-2x] &= \frac{x}{1-x} \\ H(x) = \frac{x}{(1-x)[1-2x]} &= \frac{1}{1-2x} - \frac{1}{1-x} \\ h_n &= 2^n - 1 \end{aligned}$$

5.5.4 Recurrence Equation

Find the solution for the recurrence relation

Problem Statement

$$\begin{Bmatrix} x_n = 6x_{n-1} - 9x_{n-2} \\ x_0 = 2 \\ x_1 = 3 \end{Bmatrix}$$

$$x_n - 6x_{n-1} + 9x_{n-2} = 0 \tag{23}$$

$$\text{Finding Characteristic Equation} \Rightarrow \quad r^2 - 6r + 9 = 0 \tag{24}$$

$$\text{Factoring} \rightsquigarrow \quad (r-3)^2 = 0 \tag{25}$$

$$x_n = C_1 3^n + C_2 n 3^n \tag{26}$$

$$\text{Using initial Condition} \quad x_0 = 2 \longrightarrow \quad x_0 = 2 = C_1$$

$$\text{Using initial Condition} \quad x_1 = 3 \longrightarrow \quad x_1 = 3 = 2(3)^1 + C_2(1)3^1$$

$$C_2 = -1$$

$$\boxed{x_n = 2 \cdot 3^n - n \cdot 3^n} \tag{27}$$

$$\begin{cases} x_n = \frac{1}{2}p_{n+1} - \frac{1}{2}p_{n-1} \\ p_0 = 0 \\ p_{100} = 1 \end{cases}$$

5.6 Another Problem

Another Generic Fancy Boxed Math equation

$$\text{Characteristic Equation} \longrightarrow r^2 - 2r + 1 = 0$$

$$\text{Factoring} \longrightarrow (r-1)^2$$

$$p_n = c_1 + c_2 n$$

$$\text{Applying the boundary conditions} \quad p_0 = 0 \; and \; p_{100} = 1$$

$$p_n = \frac{n}{100}, \qquad 0 \leq n \leq 100 \tag{28}$$

5.7 Problem 5

PROBLEM STATEMENT

$$\begin{cases} a_n = 10a_{n-1} - 25a_{n-2} + 8 \cdot 5^n \\ a_0 = 6 \\ a_1 = 10 \end{cases}$$

5.8 Another Quality Box

STEPS TO SOLVE PROBLEM

$$a_n - 10x_{n-1} + 25a_{n-2} = 8 \cdot 5^n$$

$$\sum_{n=0}^{\infty} a_{n+2} \cdot x^{n+2} - 10 \sum_{n=0}^{\infty} a_{n+1} \cdot x^{n+2} + 25 \sum_{n=0}^{\infty} a_n \cdot x^{n+2} = 8 \sum_{n=0}^{\infty} x^{n+2} 5^n$$

$$\sum_{n=0}^{\infty} a_{n+2} \cdot x^{n+2} - 10x \sum_{n=0}^{\infty} a_{n+1} \cdot x^{n+1} + 25x^2 \sum_{n=0}^{\infty} a_n \cdot x^n = 8x^2 \sum_{n=0}^{\infty} x^n 5^n$$

Continued

$$A(x) - a_0 - a_1 - 10x(A(x) - a_0) + 25x^2 A(x) = 8x^2 \cdot \frac{1}{1 - 5x}$$

$$A(x) - 6 - 10 - 10x(A(x) - 6) + 25x^2 A(x) = \frac{8x^2}{1 - 5x}$$

$$A(x)[1 - 10x + 25x^2] = \frac{8x^2}{1 - 5x} + 16 - 60x$$

$$A(x)[(5x - 1)^2] = \frac{8x^2}{1 - 5x} + 16 - 60x$$

$$A(x) = \frac{8x^2}{(1 - 5x)(5x - 1)^2} + \frac{16}{(5x - 1)^2} - \frac{60x}{(5x - 1)^2}$$

Partial Fractions

$$\frac{8x^2}{(1-5x)^3} = -\frac{8}{25(1-5x)} - \frac{16}{(1-5x)^2} - \frac{8}{25(1-5x)^3}$$

Partial Fractions 2

$$\frac{60x}{(5x-1)^2} = \frac{12}{5(5x-1)} + \frac{12}{5(5x-1)^2}$$

5.8.1 Another Good Box

OGF

$$A(x) = -\frac{8}{25(1-5x)} - \frac{16}{(1-5x)^2} - \frac{8}{25(1-5x)^3} + \frac{16}{(1-5x)^2} + \frac{12}{5(5x-1)} + \frac{12}{5(5x-1)^2}$$

$$a_n = \frac{-8}{25}5^n - 16(n+1)5^n - \frac{8}{25}(n+1)5^{2n} + 16 \cdot 5^n + \frac{12}{5}5^n + \frac{12}{5}(n+1)5^n$$

Answer Should be

$$x_n = (4n^2 - 8n + 6)5^n \tag{29}$$

Problem Statement

5.9 Last Recursion Problem

Solve for a_n given that:

$$u_n = 3u_{n-1} + 10u_{n-2} + 7 \cdot 5^n$$
$$u_0 = 4$$
$$u_1 = 3$$

Problem Work Wrong

$$u_n - 3u_{n-1} - 10u_{n-2} = 7 \cdot 5^n$$

$$\text{Characteristic Equation} = r^2 - 3r - 10 = 0 \iff (r-5)(r+2) = 0$$

$$r_1 = 5 \quad r_2 = -2$$

$$\text{Trial Solution is:} \quad u_n = An^2 5^n$$

$$\text{Putting it back into the equation} \quad A = 5$$

$$u_n^{(p)} = n \cdot 5^{n+1}$$

$$u_n^{(h)} = A_0 \cdot 5^n + A_1 \cdot (-2)^n$$

$$\text{Using Initial Condition} \quad u_n^{(h)} = -2 \cdot 5^n + 6 \cdot (-2)^n$$

$$u_n = u_n^{(h)} + u_n^{(p)} = n \cdot 5^{n+1} - 2 \cdot 5^n + 6 \cdot (-2)^n$$

Problem set 2 located Here at pg 100.

5.9.1 P set 2

$$a_n - 3a_{n-1} = 2^n$$

$$\sum_{n=0}^{\infty} a_{n+1} \cdot x^{n+1} - 3x \sum_{n=0}^{\infty} a_n \cdot x^n = x \sum_{n=0}^{\infty} 2^n \cdot x^n$$

$$A(x) - a_0 - 3x \cdot A(x) = x \frac{1}{1-2x}$$

$$[1-3x]A(x) = \frac{x}{1-2x} + a_0$$

$$A(x) = \frac{x}{(1-2x)(1-3x)} + \frac{a_0}{1-3x}$$

Applying Partial Fractions to $\dfrac{x}{(1-2x)(1-3x)} \longrightarrow \dfrac{-2}{1-2x} + \dfrac{3}{1-3x}$

$$A(x) = \frac{-2}{1-2x} + \frac{3}{1-3x} + \frac{a_0}{1-3x}$$

$$a_n = -2(2)^n + (3+a_0)3^n$$

5.9.2 Q1

Solving the Recursion Leads to

$$a_n = (\alpha)3^n - (2)^{n+1} \tag{30}$$

5.9.3 Q2

$$a_n - 3a_{n-1} = 3^n$$

$$\sum_{n=0}^{\infty} a_{n+1} \cdot x^{n+1} - 3x \sum_{n=0}^{\infty} a_n \cdot x^n = x \sum_{n=0}^{\infty} 3^n \cdot x^n$$

$$A(x) - a_0 - 3x \cdot A(x) = x\frac{1}{1-3x}$$

$$[1-3x]A(x) = \frac{x}{1-3x} + a_0$$

$$A(x) = \frac{x}{(1-3x)^2} + \frac{a_0}{1-3x}$$

Partial Fractions of $\quad \dfrac{x}{(1-3x)^2} = \dfrac{A}{(1-3x)} + \dfrac{B}{(1-3x)^2}$

Applying Partial Fractions to $\quad \dfrac{x}{(1-3x)^2} \longrightarrow \quad \dfrac{-1}{3(1-3x)} + \dfrac{1}{3(1-3x)^2}$

$$A(x) = \frac{-1}{3(1-3x)} + \frac{1}{3(1-3x)^2} + \frac{a_0}{1-3x}$$

Using the Correct OGF $\longrightarrow \quad a_n = -\dfrac{1}{3}(3)^n + n \cdot (3)^n + a_0 \cdot (3)^n$

Solving the Recursion Leads to

$$a_n = (\alpha)3^n + n(3)^n, \qquad where\ \alpha = a_0 - \frac{1}{3} \tag{31}$$

5.9.4 Q3

$$\begin{cases} a_n = 2a_{n-1} - a_{n-2} + 2n \\ a_0 = \alpha \\ a_1 = \beta \end{cases}$$

$$a_n - 2a_{n-1} + a_{n-2} = 2n$$

$$\sum_{n=0}^{\infty} a_{n+2} \cdot x^{n+2} - \sum_{n=0}^{\infty} a_{n+1} \cdot x^{n+2} + \sum_{n=0}^{\infty} a_n \cdot x^{n+2} = x^2 \sum_{n=0}^{\infty} x^n 2n$$

$$[A(x) - a_0 - a_1] - 2x[A(x) - a_0] + x^2 A(x) = x^2 \frac{2 \cdot x}{(1-x)^2}$$

$$[1 - 2x + x^2]A(x) = \frac{2 \cdot x^3}{(1-x)^2} - 2\alpha x + \alpha + \beta$$

$$[(1-x)^2]A(x) = \frac{2 \cdot x^3}{(1-x)^2} - 2\alpha x + \alpha + \beta$$

$$A(x) = \frac{2 \cdot x^3}{(1-x)^2(1-x)^2} - \frac{2\alpha x}{(1-x)^2} + \frac{\alpha}{(1-x)^2} + \frac{\beta}{(1-x)^2}$$

$$A(x) = \frac{2 \cdot x^3}{(1-x)^4} - \frac{2\alpha x}{(1-x)^2} + \frac{\alpha}{(1-x)^2} + \frac{\beta}{(1-x)^2}$$

Applying Partial Fractions to $\frac{2 \cdot x^3}{(1-x)^4} \longrightarrow \frac{2}{1-x} + \frac{6}{(1-x)^2} + \frac{6}{(1-x)^3} + \frac{2}{(1-x)^4}$

Applying Partial Fractions to $\frac{2 \cdot \alpha x}{(1-x)^2} \longrightarrow \frac{2 \cdot \alpha}{1-x} + \frac{2 \cdot \alpha}{(1-x)^2}$

$$A(x) = \frac{2}{1-x} + \frac{6}{(1-x)^2} + \frac{6}{(1-x)^3} + \frac{2}{(1-x)^4} + \frac{2 \cdot \alpha}{1-x} + \frac{2 \cdot \alpha}{(1-x)^2} + \frac{\alpha}{(1-x)^2} + \frac{\beta}{(1-x)^2}$$

$$a_n = 2 + (n+1)6 + 6 \cdot (n+1)6 + [(n+1) * 2]^2 + 2\alpha + (n+1) \cdot 2\alpha + (n+1) \cdot \alpha + (n+1)\beta$$

Note that this answer is probably wrong and requires simplification or the use of the correct OGF to get to the right answer

Right answer in the box

$$a_n = \alpha + \beta n + n^2 + \frac{1}{3}n^3 \tag{32}$$

6 Max-flow network problem

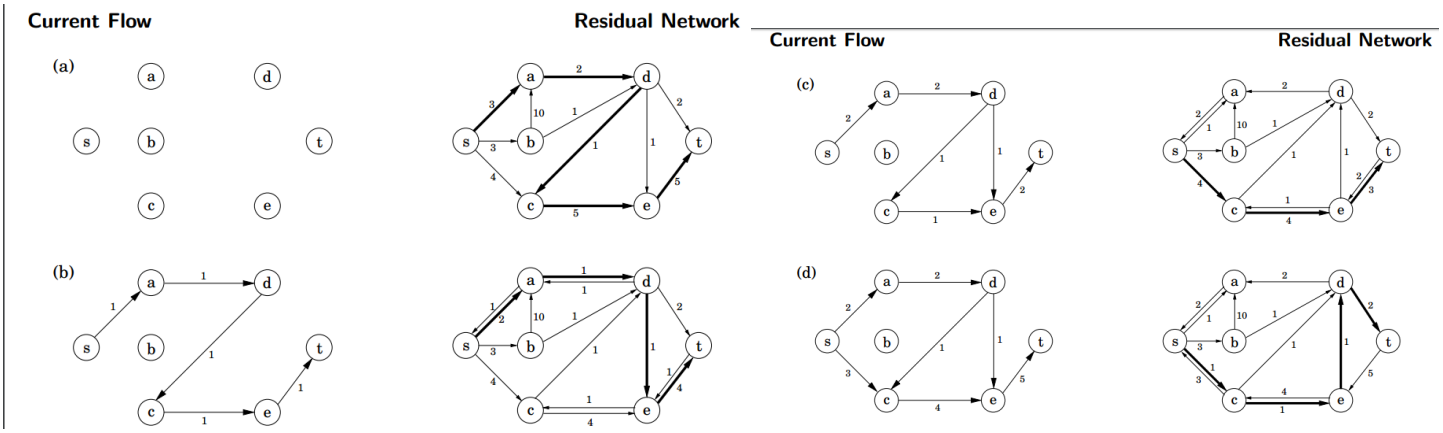
Edmonds-Karp algorithm

1. Initialize: let $\boldsymbol{X_f} \leftarrow \mathbf{0}$ and $\mathcal{N}^*(\boldsymbol{X_f}) \leftarrow \mathcal{N}$;
2. Using BFS, find a path Π from source s to sink t in $\mathcal{N}^*(\boldsymbol{X_f})$;
3. **while** $\Pi \neq \emptyset$ **do**:
   - 3.1. Augment $\delta = \min\{c^*_{jk} \mid (v_j, v_k) \in \Pi\}$ units of flow through every arc along Π, and update $\boldsymbol{X_f}$ accordingly;
   - 3.2. Update residual network $\mathcal{N}^*(\boldsymbol{X_f})$;
   - 3.3. Using BFS, find a path Π from source s to sink t in $\mathcal{N}^*(\boldsymbol{X_f})$;

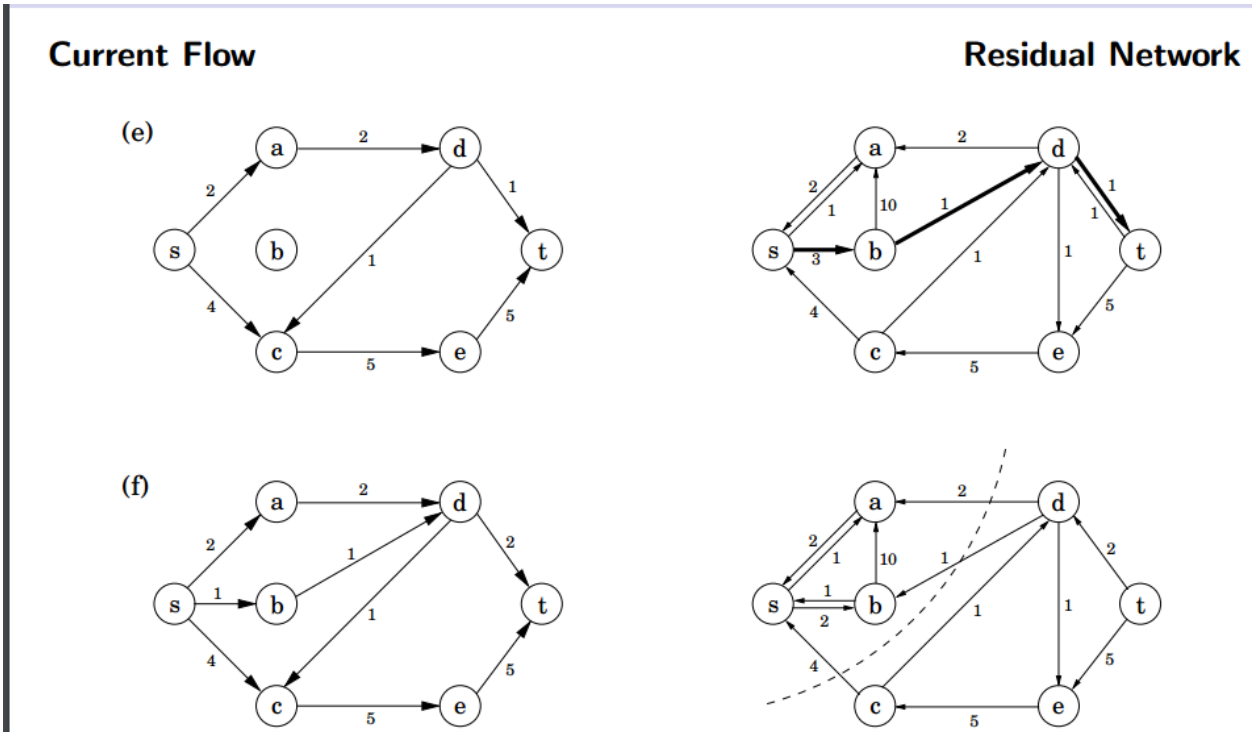
Figure 6: Edmonds Karp

6.1 Minimum capacity and Maximum Flow

Among all possible s-t cuts in a given network N = (V, E, u), a cut with a minimum capacity determines a maximum flow from s to t in N . We require that the inflow carried by v_k's incoming arcs be the same as the outflow carried by v_k's outgoing arcs. In other words, every network node, except for s and t, must conserve the flow.



(a) (b)



(c)

Figure 7: Slides Example 7.1.

E5: The current flow f in the network G shown at the top in Figure 11.1.6 has value 9. The corresponding residual network G_f is shown at the bottom.

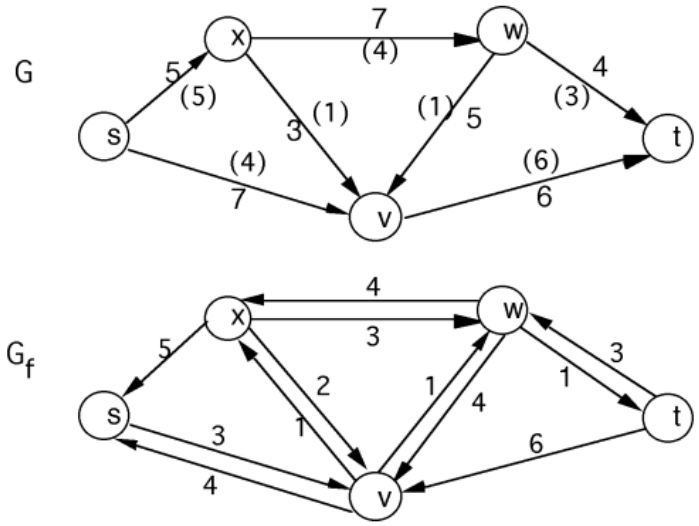


Figure 11.1.6 **A network G with flow f and its residual network G_f.**

Figure 8: Example 7.1

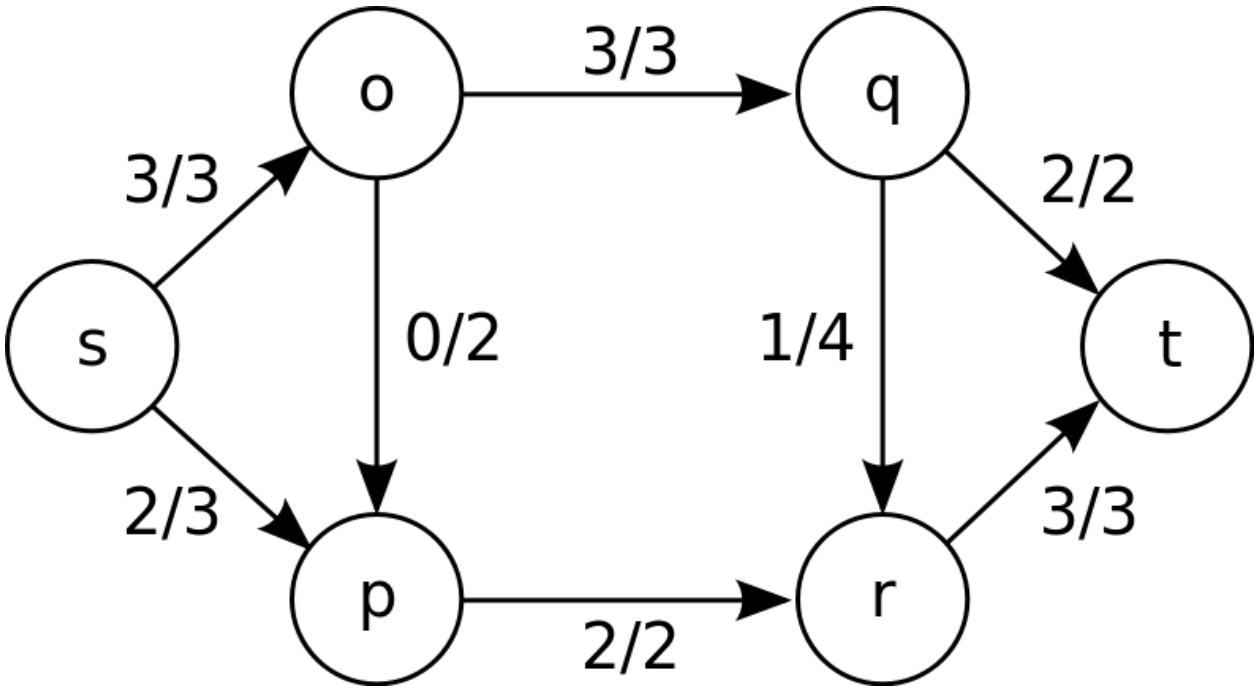
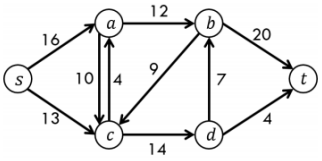


Figure 9: Example 7.2 Network Flow Example Link

- Capacities



- Maximum flow (of 23 total units)

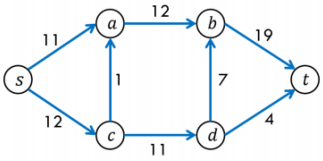
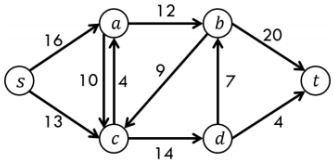


Figure 10: Example 7.2 **part 1**

- Capacities



- Minimum Cut (red edges are removed)

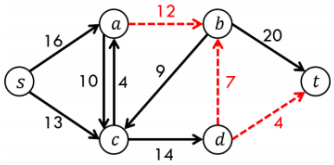


Figure 11: Exaample 7.2 **part 2**

Figure 12: Network Flow Example

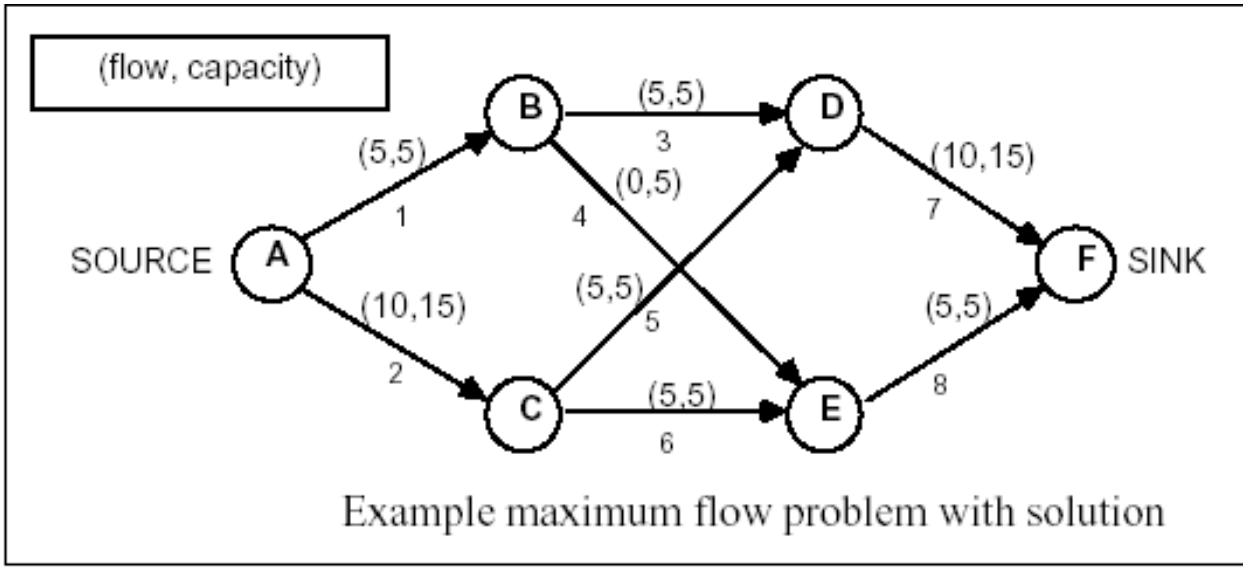


Figure 13: Example 7.3

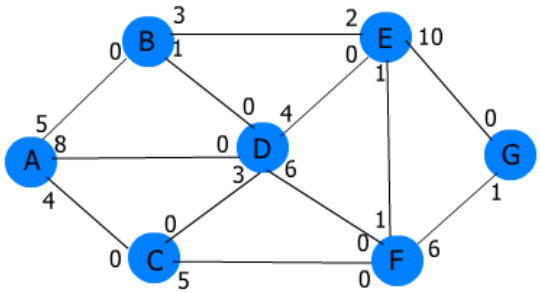


Figure 14: Example 7.4 **part 1**

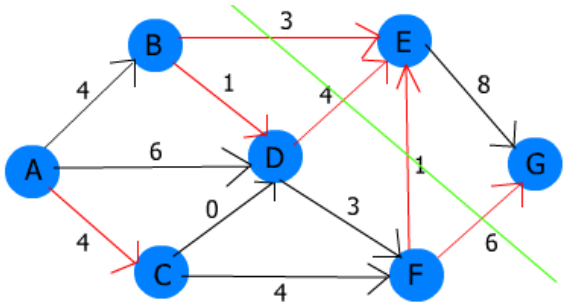


Figure 15: Example 7.4 **part 2**

Figure 16: Go find more network flow problems

7 Knapsack problem

Remember that w_i is the COST of the ith object.
C is equal to the capacity and or maximum weight
v_i is the VALUE of the ith object. Try to maximum stuff.

7.1 Greedy Heuristic

Link to the Simple Problem

$$N = 3 \quad w = [3 \;\; 8 \;\; 5]^T \quad v = [4 \;\; 6 \;\; 5]^T \quad C^* = C = 8$$

Steps to solve the problem

$$p_i = \left[\frac{v_1}{w_1} \;\; \frac{v_2}{w_2} \;\; \frac{v_3}{w_3}\right] = \left[\frac{4}{3} \;\; \frac{6}{8} \;\; \frac{5}{5}\right] = [1.33 \;\; 0.75 \;\; 1]$$

1. Select p_1 and take out $w_1 \rightarrow \quad C^* = C^* - 3 = 5$

2. Select p_3 and take out $w_3 \rightarrow \quad C^* = C^* - 5 = 0$

3. End the Process $\rightarrow$ Optimal Solution $\rightarrow \quad x^* = [1 \;\; 0 \;\; 1]^T$

Link to the Moderate Problem

$$N = 5 \quad w = [12 \;\; 1 \;\; 2 \;\; 1 \;\; 4]^T \quad v = [4 \;\; 2 \;\; 2 \;\; 1 \;\; 10]^T \quad C = 15$$

Steps to solve the problem

$$p_i = \left[\frac{v_1}{w_1} \;\; \frac{v_2}{w_2} \;\; \frac{v_3}{w_3} \;\; \frac{v_4}{w_4} \;\; \frac{v_5}{w_5}\right] = \left[\frac{4}{12} \;\; \frac{2}{1} \;\; \frac{2}{2} \;\; \frac{1}{1} \;\; \frac{10}{4}\right] = [0.333 \;\; 2 \;\; 1 \;\; 1 \;\; 2.5]$$

1. Select p_5 and take out $w_5 \rightarrow \quad C^* = C^* - w_5 = 15 - 4 = 11$
2. Select p_2 and take out $w_2 \rightarrow \quad C^* = C^* - w_2 = 11 - 1 = 10$
3. Select p_3 and take out $w_3 \rightarrow \quad C^* = C^* - w_3 = 10 - 2 = 8$
4. Select p_4 and take out $w_4 \rightarrow \quad C^* = C^* - w_4 = 8 - 1 = 7$
5. End the Process $\rightarrow$ Optimal Solution $\rightarrow \quad x^* = [0 \;\; 1 \;\; 1 \;\; 1 \;\; 1]^T$

Link to the Hard Problem

7.1.1 Another Reasonably good box

Answer Using Greedy Heuristic

$$x^* = [0 \;\; 1 \;\; 0 \;\; 0 \;\; 1 \;\; 1 \;\; 1]^T$$

7.2 Branch And Bound

The knapsack problem or rucksack problem is a problem in combinatorial optimization: Given a set of items, each with a weight and a value, determine the number of each item to include in a collection so that the total weight is

less than or equal to a given limit and the total value is as large as possible. It derives its name from the problem faced by someone who is constrained by a fixed-size knapsack and must fill it with the most valuable items.

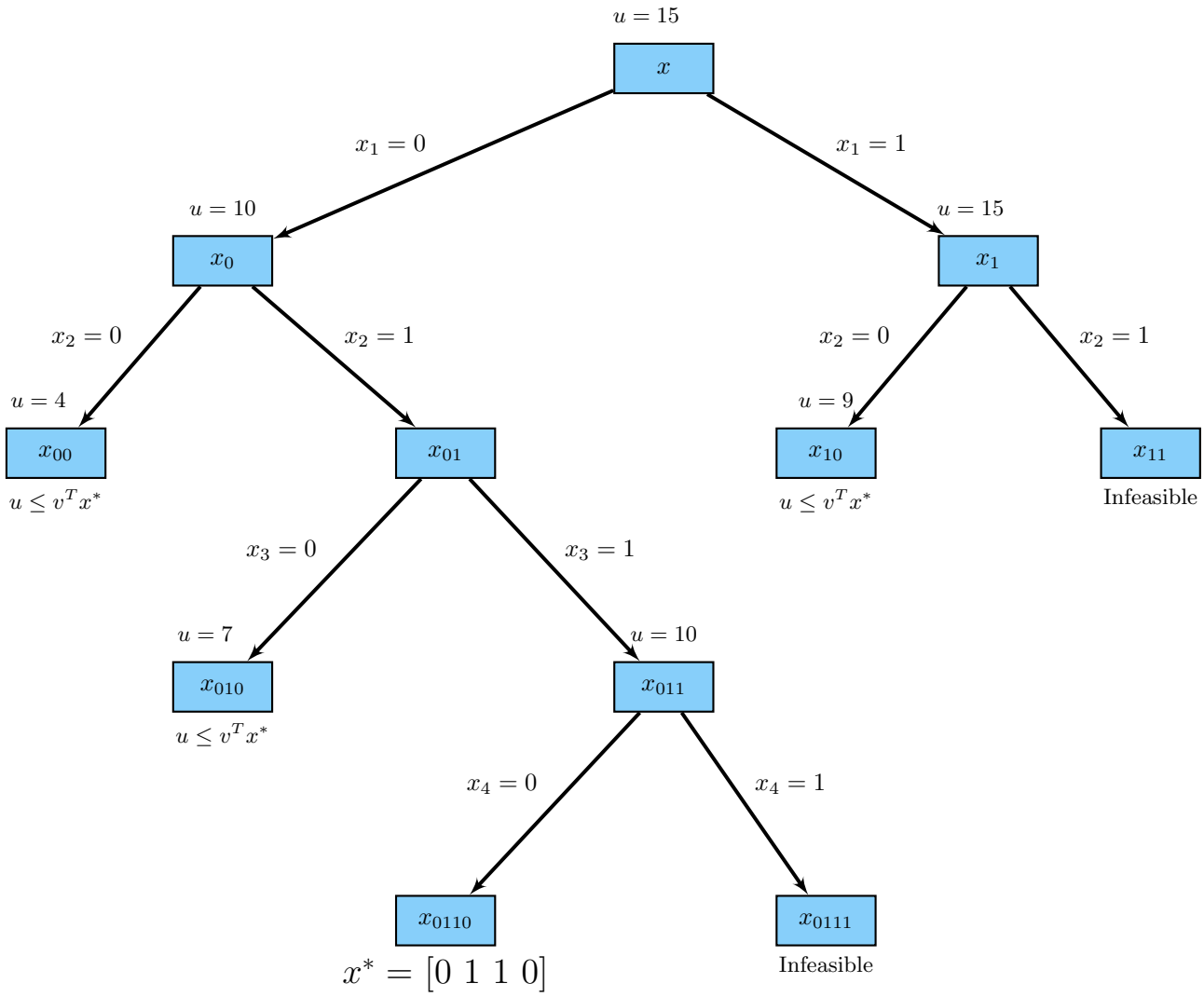


Figure 17: Branch And Bound Example from Ceng 242 HW

7.2.1 Simple Example

Steps to solve the problem

$$p_i = \begin{bmatrix} \frac{v_1}{w_1} & \frac{v_2}{w_2} & \frac{v_3}{w_3} & \frac{v_4}{w_4} & \frac{v_5}{w_5} \end{bmatrix} = \begin{bmatrix} \frac{4}{12} & \frac{2}{1} & \frac{2}{2} & \frac{1}{1} & \frac{10}{4} \end{bmatrix} = [0.333 \quad 2 \quad 1 \quad 1 \quad 2.5]$$

Skip to Page 52 for the full solution

Optimal Solution $\rightarrow \quad x^* = [0 \quad 1 \quad 1 \quad 1 \quad 1]^T$

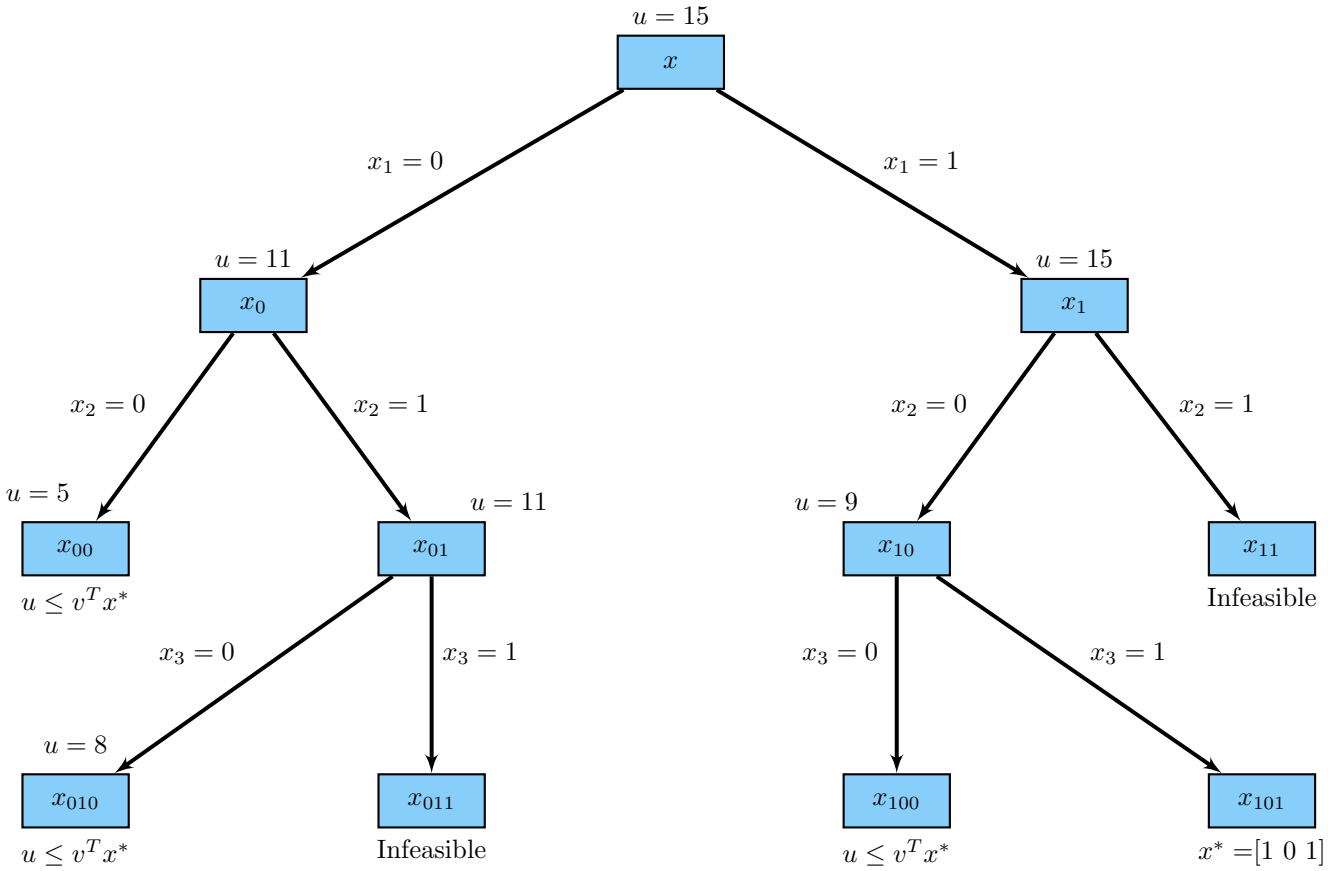


Figure 18: Simple Branch And Bound

7.3 Description

7.4 Ceng 242 Slides for Knapsack Problem

These slides are using in understanding the knapsack problem

Solving 0-1 knapsack problem

Recall two basic versions of the **0-1 knapsack problem**, where $C \in \mathbb{Z}_+$ and $\boldsymbol{v}, \boldsymbol{w} \in \mathbb{Z}_+^N$:

- $\max\{\boldsymbol{v}^\top \boldsymbol{x} \mid \boldsymbol{w}^\top \boldsymbol{x} \leq C,\ \boldsymbol{x} \in \mathbb{B}^N\}$,
- $\min\{\boldsymbol{w}^\top \boldsymbol{x} \mid \boldsymbol{v}^\top \boldsymbol{x} \geq C,\ \boldsymbol{x} \in \mathbb{B}^N\}$.

There are three basic approaches to solving this problem:

- **Greedy heuristic** of complexity $O(N \log N)$, with no guarantee of solution optimality;
- **Branch-and-bound method** of complexity $O(2^N)$, with a guarantee of solution optimality;
- **Dynamic programming (DP) method** of complexity $O(CN)$, with a guarantee of solution optimality.

Greedy heuristic: $\max\{\boldsymbol{v}^\top \boldsymbol{x} \mid \boldsymbol{w}^\top \boldsymbol{x} \leq C,\, \boldsymbol{x} \in \mathbb{B}^N\}$

Let $C^* = C$ be the residual knapsack capacity.

Step 1:
For every $n \in \{1, 2, ..., N\}$, set object *priorities* $p_n = \frac{v_n}{w_n}$. Initialize $\boldsymbol{x}$ to $\mathbf{0}$.

Step 2:
Determine $p_k = \max\{p_n \mid x_n = 0, w_n \leq C^*, n = 1, 2, ..., N\}$ and let $x_k = 1$, $C^* \leftarrow C^* - w_k$. In other words, select the highest-priority knapsack-fitting object, put it in the knapsack, and update the residual knapsack capacity.

Repeat **Step 2** as long as $\{p_n \mid x_n = 0, w_n \leq C^*, n = 1, 2, ..., N\} \neq \emptyset$.

Branch-and-bound method I

The **branch-and-bound method** uses **implicit enumeration** (as opposed to explicit enumeration performed by an exhaustive search): it detects and rejects infeasible and suboptimal solutions without fully computing them.

Let $\tilde{\boldsymbol{x}}$ be a complete feasible solution of some *minimization problem*, let $\boldsymbol{x}^*$ be the *best complete feasible solution found so far*, and let $f(\tilde{\boldsymbol{x}})$ and $f(\boldsymbol{x}^*)$ denote their respective objective values. Instead of computing complete $\tilde{\boldsymbol{x}}$ and then checking if $f(\tilde{\boldsymbol{x}})$ is an improvement over $f(\boldsymbol{x}^*)$, we find a *lower bound* $\ell \leq f(\tilde{\boldsymbol{x}})$. If $\ell \geq f(\boldsymbol{x}^*)$, we reject $\tilde{\boldsymbol{x}}$; else, we proceed by branching out to related **subproblems**. If our computed feasible solution $\tilde{\boldsymbol{x}}$ is such that $f(\boldsymbol{x}^*) > f(\tilde{\boldsymbol{x}})$, we let $\boldsymbol{x}^* \leftarrow \tilde{\boldsymbol{x}}$. Once every feasible solution has been either fully computed or rejected, we output $\boldsymbol{x}^*$ as our *optimal solution*.

Branch-and-bound method II

Although the branch-and-bound method has *exponential complexity* in the worst case, it often terminates much sooner than an exhaustive search, but it may require a significant amount of memory.

To illustrate basic principles of the branch-and-bound method, we shall use some generic 0-1 ILP problem X as an example. Suppose our problem X involves *minimization* of some objective function $f(\boldsymbol{x})$, where $\boldsymbol{x} \in \mathbb{B}^N$ is our binary decision vector $[\ x_1\ x_2\ \ldots\ x_N\ ]^\top$ for X.

Let $\mathcal{S}$ be a set of **active (unsolved) subproblems** of X. Each subproblem shall be denoted by $X_{x_1 x_2 \ldots x_{n-1}}$ to indicate that its $n-1$ decision variables, written as $[\ x_1\ x_2\ \ldots\ x_{n-1}\ ]^\top$, have been determined and fixed, whereas its remaining decision variables, written as $[\ \widehat{x}_n\ \widehat{x}_{n+1}\ \ldots\ \widehat{x}_N\ ]^\top$, are still to be resolved. Next, we present an outline of the branch-and-bound method.

Branch-and-bound method III

1. Find a feasible solution $\boldsymbol{x}^*$ (for example, use a greedy heuristic). Let $\mathcal{S} \leftarrow \{X\}$.
2. Remove some $X_{x_1 x_2 \ldots x_{n-1}}$ from $\mathcal{S}$. If $\ell(X_{x_1 x_2 \ldots x_{n-1}}) \geq f(\boldsymbol{x}^*)$, reject it; otherwise, expand it into *new* active subproblems: $X_{x_1 x_2 \ldots x_{n-1} 0}$ (fixed decision $x_n = 0$) and $X_{x_1 x_2 \ldots x_{n-1} 1}$ (fixed decision $x_n = 1$).
3. If $X_{x_1 x_2 \ldots x_{n-1} 0}$ has no feasible solutions, reject it; otherwise, if $n < N$, let $\mathcal{S} \leftarrow \mathcal{S} \cup \{X_{x_1 x_2 \ldots x_{n-1} 0}\}$. The case of $n = N$ means that we have obtained a *complete feasible solution* $\tilde{\boldsymbol{x}} = [\, x_1 \; x_2 \; \ldots \; x_{N-1} \; 0 \,]^\top$ that must be compared to $\boldsymbol{x}^*$: if $f(\boldsymbol{x}^*) > f(\tilde{\boldsymbol{x}})$, let $\boldsymbol{x}^* \leftarrow \tilde{\boldsymbol{x}}$.
4. Apply step 3 to $X_{x_1 x_2 \ldots x_{n-1} 1}$.
5. If $\mathcal{S} \neq \emptyset$, go to step 2; otherwise, output $\boldsymbol{x}^*$ and stop.

Example: $\min\{\boldsymbol{w}^\top \boldsymbol{x} \mid \boldsymbol{v}^\top \boldsymbol{x} \geq C,\ \boldsymbol{x} \in \mathbb{B}^4\}$

$\boldsymbol{v} = [\,5\ 5\ 4\ 2\,]^\top \qquad \boldsymbol{w} = [\,5\ 6\ 3\ 1\,]^\top \qquad C = 9$

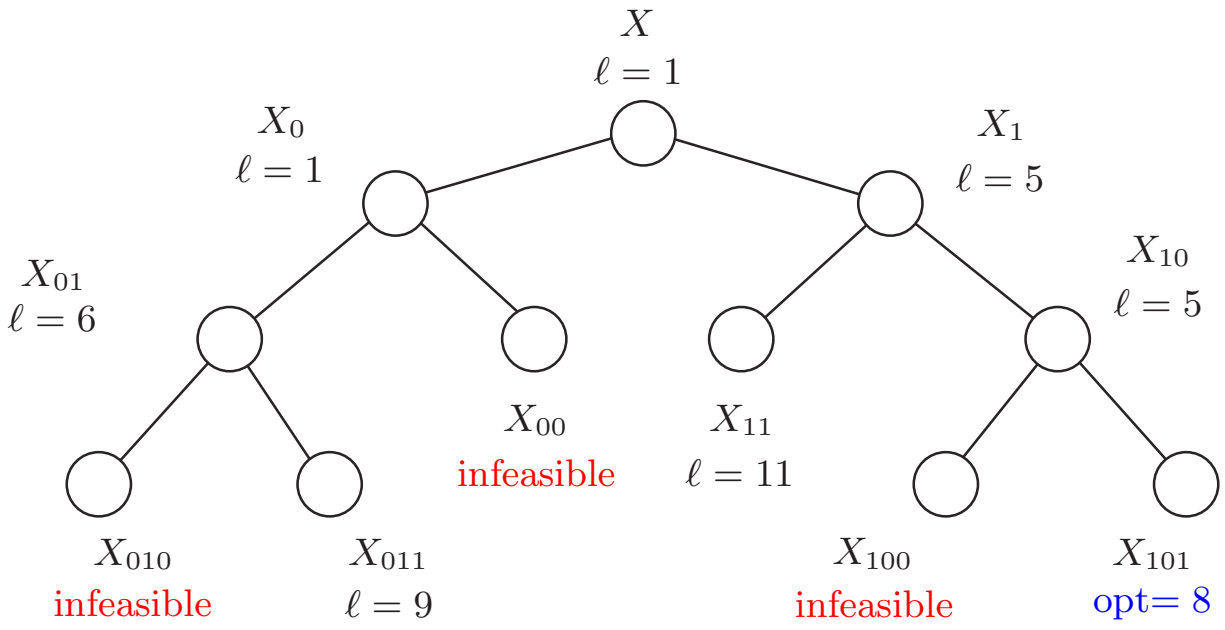


Figure: Solving a 0-1 knapsack problem with the branch-and-bound method. Our subproblems are labeled X_{x_1}, $X_{x_1x_2}$, $X_{x_1x_2x_3}$. Optimal solution is $\boldsymbol{x}^* = [\,1\ 0\ 1\ 0\,]^\top$.

* Graphics from *Mathematics of Operational Research* by Weber (web).

8 ILP/network-flow problem formulations

These slides are useful in understanding how to formulate Integer linear programming problems

LP: Standard form Φ_1

Let: $\boldsymbol{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$, $\boldsymbol{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}$, $\boldsymbol{c} = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix}$, $\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$.

Linear Program Φ_1: $\min\{\boldsymbol{c}^\top \boldsymbol{x} \mid \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, \boldsymbol{x} \geq \boldsymbol{0}\}$

- INPUT: $\boldsymbol{A} = [a_{jk}]_{m \times n}$, $\boldsymbol{b} = [b_j]_{m \times 1}$, $\boldsymbol{c} = [c_k]_{n \times 1}$.
- OUTPUT: $\boldsymbol{x} = [x_k]_{n \times 1}$.
- OBJECTIVE: Minimize $\boldsymbol{c}^\top \boldsymbol{x}$.
- CONSTRAINTS: $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ and $\boldsymbol{x} \geq \boldsymbol{0}$.

LP: Standard form Φ_2

Let: $\boldsymbol{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$, $\boldsymbol{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}$, $\boldsymbol{c} = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix}$, $\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$.

Linear Program Φ_2: $\max\{\boldsymbol{c}^\top \boldsymbol{x} \mid \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}, \boldsymbol{x} \geq \boldsymbol{0}\}$

- INPUT: $\boldsymbol{A} = [a_{jk}]_{m \times n}$, $\boldsymbol{b} = [b_j]_{m \times 1}$, $\boldsymbol{c} = [c_k]_{n \times 1}$.
- OUTPUT: $\boldsymbol{x} = [x_k]_{n \times 1}$.
- OBJECTIVE: Maximize $\boldsymbol{c}^\top \boldsymbol{x}$.
- CONSTRAINTS: $\boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}$ and $\boldsymbol{x} \geq \boldsymbol{0}$.

Description of ILP problem

A Linear Programming (LP) problem is of the form minimize $c^T x$ subject to $Ax = b$ where A (a rational $m \times n$ matrix), c (a rational n-vector) and b (a rational m-vector) are given and x is an n-vector to be determined. In other words, we try to find the minimum of a linear function over a feasible set defined by a finite number of linear constraints. It can be shown that a problem with linear equalities or "<= "linear inequalities can always be put in the above form, implying that this formulation is more general than it might look. An Integer Linear Programming (ILP) problem is obtained from an LP problem by requiring that all entries of the solution vector x are integer. LP problems are "easy" to solve (they are in the complexity class P), whereas ILP problems are, in general, difficult (they are NP-hard).

8.1 Network-flow

These slides begin the description of how to formulate a network flow problem

Maximum network flow problem: alternative formulation I

Given a network $\mathcal{N} = (V, E, \boldsymbol{u})$, we can find a maximum flow from source s to sink t in $O(|V||E|^2)$ time using the **labeling algorithm** (developed by Fulkerson and Ford, improved by Edmonds and Karp).

Before presenting key ideas behind this algorithm, we introduce a **capacity matrix** $\boldsymbol{C_u} = [c_{jk}]_{N\times N}$ and a **flow matrix** $\boldsymbol{X_f} = [x_{jk}]_{N\times N}$, where $N = |V|$:

$$c_{jk} = \begin{cases} u_i, & \text{if there is an arc } e_i = (v_j, v_k) \in E; \\ 0, & \text{otherwise.} \end{cases}$$

$$x_{jk} = \begin{cases} f_i, & \text{if there is an arc } e_i = (v_j, v_k) \in E; \\ 0, & \text{otherwise.} \end{cases}$$

Essentially, we have distributed the elements of the capacity vector $\boldsymbol{u}$ and the flow vector $\boldsymbol{f}$ throughout the corresponding capacity matrix $\boldsymbol{C_u}$ and the flow matrix $\boldsymbol{X_f}$.

Maximum network flow problem: alternative formulation II

Below is a modified formulation of the maximum network flow problem, using $\boldsymbol{C_u}$ and $\boldsymbol{X_f}$ (instead of $\boldsymbol{u}$ and $\boldsymbol{f}$), where $s = v_1$ and $t = v_N$:

$$\text{maximize} \qquad \sum_{k=2}^{N-1} x_{1k},$$

$$\text{subject to} \qquad \sum_{j=2}^{N-1} x_{ji} = \sum_{k=2}^{N-1} x_{ik}, \qquad \text{for every } i \in \{2, 3, ..., N-1\},$$

$$0 \leq x_{jk} \leq c_{jk}, \quad x_{jk} \in \mathbb{Z}, \qquad \text{for every } j, k \in \{1, 2, ..., N\}.$$

Example: C_u and X_f

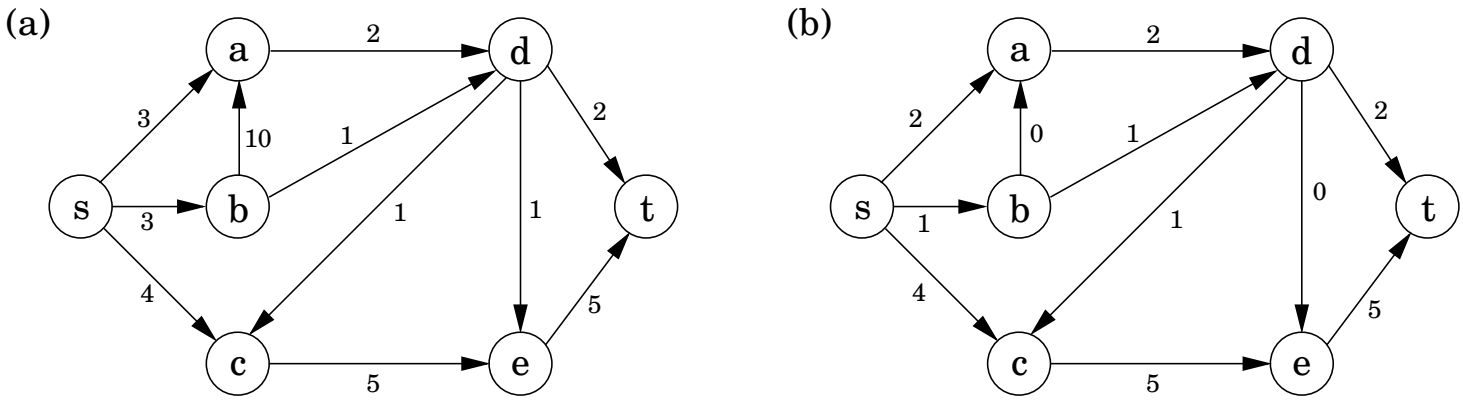


Figure: (a) Capacities, (b) Flows; $\{v_1, v_2, v_3, v_4, v_5, v_6, v_7\} = \{s, a, b, c, d, e, t\}$.

$$C_u = \begin{bmatrix} 0 & 3 & 3 & 4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 10 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 5 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 5 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \geq X_f = \begin{bmatrix} 0 & 2 & 1 & 4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 5 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 5 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

* Graphics from *Algorithms* by Dasgupta *et al* (web).