

Contents

1	Introduction	2
2	Background	3
2.1	Discrete Uniform Distribution	3
2.2	Law of Total Expectation	3
2.3	k -ary Tree	4
3	Analysis	5
3.1	k -ary Tree for Trial Rule	5
3.2	Trial Rule from Probability Distribution	8
3.3	Trial Number Expectation	10
3.4	Node Merging	15
3.5	Finding Minimum $\mathbf{E}[X]$	17
3.6	k -ary Numeral System for Computation	21
4	Implementation	24
4.1	Development Environment	24
4.2	Calculator	24
4.3	Printer	27
5	Discussion	30
5.1	Minimum $\mathbf{E}[X]$ with Non-Maximum c	30
5.2	Converging Minimum $\mathbf{E}[X]$	32
6	Conclusion and Further Research	34

1 Introduction

한국콘텐츠진흥원에서는 매년 게임콘텐츠 제작지원 사업을 진행하고 있다. 2021년 게임콘텐츠 제작지원 사업안내서를 살펴보면 제작지원 대상 분야로 ‘보드게임’이 신시장창출형 부문에 포함되어 있다는 것[1]을 확인할 수 있는데, 이를 통해 최근까지도 보드게임에 대해 시장창출이 가능한 수준의 지속적인 수요가 존재한다는 사실을 알 수 있다. 또한 보드게임은 기획 및 개발의 진입장벽이 컴퓨터 프로그래밍이 필수적으로 요구되는 전자게임에 비하면 낮은 편이며, 이를 통해 게임 콘텐츠 창작을 목적으로 한 보드게임 제작 활동이 꾸준히 이루어지고 있음을 짐작할 수 있다.

프랑스의 사상가 로제 카이와(Roger Caillois)는 그의 저서에서 인간이 놀이로부터 재미를 느끼는 근본적인 원리를 4가지 범주로 정리하여 제시하였다.[2] 4가지 범주 중 하나인 알레아(Alea)는 우연, 운을 기반으로 하는 놀이를 포함하는 범주이며 확률적으로 이득이나 손해를 보는 구조를 통해 인간이 재미를 느낄 수 있다는 사실을 나타낸다. 이는 오늘날까지도 게임 기획 및 제작에 있어 큰 영향을 미치고 있고, 실제로 일일히 나열하기 어려울 정도로 수많은 게임에서 확률적, 무작위적 요소를 사용하고 있다.

전자기기를 이용하여 실행하고 즐기는 전자게임과 달리 보드게임은 현실에서 실제 사물을 정해진 규칙에 따라 조작하는 형태로 즐기게 되는데, 이는 보드게임 창작자가 알레아의 실현, 다시 말해 확률적 요소의 구현에 있어 큰 어려움을 느끼도록 만드는 원인이 된다. 7% 확률로 당첨이 되는 상황을 구현해야 한다고 생각해보자. 전자게임에서는 난수생성을 활용하는 방식으로 해당 상황을 쉽게 구현할 수 있지만, 보드게임에서는 그렇지 않다. 별도의 전자기기를 준비물로 두지 않고 동전이나 카드, 주사위 등 보드게임에서 주로 사용하는 도구만을 가지고 이러한 상황을 구현하는 것은 간단한 일이 아니다.

이 논문에서는 전자기기를 준비물로 두지 않고 주사위 등의 한정된 도구만 가지고서 확률적 요소를 구현하는 방법에 대해서 다룬다. 해당 상황을 수학적으로 분석하여 문제를 정의하고 이를 해결할 것이며, 도출된 결과를 바탕으로 프로그램을 제작한 뒤 이를 사용해본 결과에 대해서도 다룬다.

2 Background

2.1 Discrete Uniform Distribution

육면체 주사위를 던져서 1부터 6까지의 숫자 중 하나를 얻는 상황을 생각해보자. 주사위를 던져 얻은 숫자를 이산확률변수(discrete random variable) X 로 둘 수 있으며, 주사위의 각 면이 나올 확률이 모두 동일하여 X 가 1부터 6까지 각각의 값이 될 확률이 모두 동일한 경우 X 가 이산균등분포(discrete uniform distribution)을 따른다고 한다.

발생확률이 동일한 k 개의 근원사건(elementary event)으로 구성된 이산표본공간(discrete sample space)에서 각 근원사건에 대해 1부터 k 까지 차례대로 번호를 붙이고, 한 번의 시행 결과 발생한 사건의 번호를 확률변수 X 로 두었을 때, X 가 따르는 이산균등분포는 다음과 같이 나타낼 수 있다.

$$\Pr(X = x) = \frac{1}{k} \quad (x = 1, \dots, k)$$

X 가 위의 이산균등분포를 따른다는 것을 줄여서 다음과 같이 나타낸다.

$$X \sim \text{unif}\{1, k\}$$

2.2 Law of Total Expectation

이산확률변수 X 의 기댓값(expectation)은 다음과 같다.

$$\mathbf{E}[X] = \sum_x x \Pr(X = x)$$

위 정의에서와 같이 표현되는 시그마 기호는 X 의 범위 안에 포함되는 가능한 모든 x 에 대해 적용되는 합을 의미한다.

사건 B 가 발생했다는 전제 하에서 사건 A 가 발생할 조건부확률(conditional probability)은 다음과 같다.

$$\Pr(A | B) = \frac{\Pr(A \cap B)}{\Pr(B)}$$

조건부확률의 정의를 이용하여 조건부기댓값(conditional expectation)을 정의할 수 있다. 사건 A 가 발생했다는 전제 하에서 이산확률변수 X 의 조건부기댓값은 다음과 같다.

$$\mathbf{E}[X | A] = \sum_x x \Pr(X = x | A)$$

그리고 전체 기댓값의 법칙(Law of Total Expectation)에 따르면 기댓값을 조건부기댓값들의 합으로 나타낼 수 있다.

Theorem (Law of Total Expectation). 사건 E_1, \dots, E_n 에 대해 $\{E_1, \dots, E_n\}$ 이 표본공간 S 를 분할할 때 이산확률변수 X 에 대해,

$$\mathbf{E}[X] = \sum_{i=1}^n \Pr(E_i) \mathbf{E}[X|E_i]$$

Proof. $\{E_1, \dots, E_n\}$ 이 표본공간 S 를 분할한다는 것은 다음이 성립한다는 의미이다.

1. $i \neq j$ 이고 $1 \leq i, j \leq n$ 일 때, $E_i \cap E_j = \emptyset$
2. $E_1 \cup \dots \cup E_n = S$

따라서 다음과 같다.

$$\begin{aligned} \sum_{i=1}^n \Pr(E_i) \mathbf{E}[X|E_i] &= \sum_{i=1}^n \Pr(E_i) \left(\sum_x x \Pr(X=x|E_i) \right) \\ &= \sum_{i=1}^n \Pr(E_i) \left(\sum_x x \frac{\Pr(X=x \cap E_i)}{\Pr(E_i)} \right) \\ &= \sum_{i=1}^n \left(\sum_x x \Pr(X=x \cap E_i) \right) \\ &= \sum_x \left(\sum_{i=1}^n x \Pr(X=x \cap E_i) \right) \\ &= \sum_x x \Pr(X=x \cap S) = \sum_x x \Pr(X=x) \\ &= \mathbf{E}[X] \end{aligned}$$

□

2.3 k -ary Tree

k 진 트리(k -ary tree)란 각각의 노드가 최대 k 개의 자식 노드를 가질 수 있는 트리 자료구조를 뜻한다. k 진 트리의 모든 노드가 0개 또는 k 개의 자식 노드를 갖는 경우 이를 정 k 진 트리(full k -ary tree)라고 한다. 그리고 포화 k 진 트리(perfect k -ary tree)는 모든 leaf 노드들의 깊이(depth)가 동일하면서 leaf 노드를 제외한 모든 노드가 k 개의 자식 노드를 가지는 k 진 트리를 뜻한다. 노드의 깊이란 해당 노드에서 root 노드까지 도달하기 위한 간선의 개수를 뜻한다. leaf 노드는 자식 노드의 수가 0이므로, 모든 포화 k 진 트리는 정 k 진 트리에 속한다.

트리 전체의 높이(height)란 트리 최하단에 위치하는 leaf 노드의 깊이를 의미한다. 위에 서 제시한 정의에 의해, 높이가 h 인 포화 k 진 트리의 최하단 leaf 노드들의 전체 개수는 k^h

개가 된다. Figure 1과 같이 높이가 3인 포화 4진 트리를 나타내는 그림을 그려보면 최하단에 위치한 leaf 노드의 개수가 총 $4^3 = 64$ 개임을 확인할 수 있다.

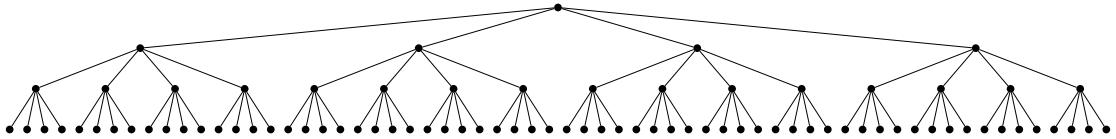


Figure 1: 높이가 3인 포화 4진 트리

3 Analysis

3.1 k -ary Tree for Trial Rule

어떤 동전을 던졌을 때 앞면, 뒷면이 나올 확률이 각각 $\frac{1}{2}$ 로 같다고 하자. 이 동전을 던져서 앞면이 나오는 사건과 뒷면이 나오는 사건에 각각 1, 2의 번호를 붙이고, 동전을 한 번 던졌을 때 발생하는 사건의 번호를 확률변수 X 로 두면 X 는 다음과 같이 이산균등분포를 따른다.

$$X \sim \text{unif}\{1, 2\}$$

동전을 반복적으로 던지면 매 시행마다 X 의 값을 얻을 수 있다. 이제 어떤 정 이진 트리 (full binary tree)가 주어졌을 때, 해당 트리의 root 노드에서 시작하여 leaf 노드에 도달할 때까지 매번 동전을 던지는 시행을 통해 얻는 X 의 값에 따라 대응되는 간선으로 이동하며 한 단계씩 내려가는 규칙을 따른다고 생각해보자.

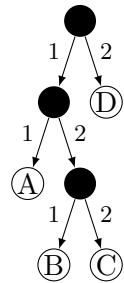


Figure 2: A부터 D까지 총 4개의 leaf 노드를 가지는 정 이진 트리

Figure 2의 트리는 A부터 D까지 총 4개의 leaf 노드를 가진다. 이 트리에서 앞에서 설명한 규칙을 따라 이동하면 최종적으로 4개의 leaf 노드 중 하나에 도달하게 될 것이다. root 노드에서 시작하여 각 leaf 노드에 도달할 확률은 Table 1과 같다.

도달한 leaf 노드	A	B	C	D	계
확률	$\left(\frac{1}{2}\right)^2$	$\left(\frac{1}{2}\right)^3$	$\left(\frac{1}{2}\right)^3$	$\frac{1}{2}$	1

Table 1: Figure 2의 root 노드에서 시작하여 각 leaf 노드에 도달할 확률을 나타낸 표

도달한 leaf 노드가 C인 경우 root 노드로 돌아가 이동을 처음부터 다시 시작하고, 그 외의 경우(A, B, D)에는 이동을 종료하는 것으로 규칙을 정하자. 그러면 최종적으로 이동을 종료하였을 때, 반드시 A, B, D 중 하나의 leaf 노드에서 멈추게 된다. root 노드에서 출발하여 이동을 A, B에서 종료하는 사건을 E_1 , D에서 종료하는 사건을 E_2 라고 하자. E_1 이 발생하려면 root 노드에서 출발한 뒤 바로 A, B에 도달하거나, C에 도달하여 처음부터 다시 시작하는 것을 1회 이상 반복한 뒤 마지막에 A, B에 도달하여야 한다. 따라서 E_1 이 발생할 확률은 다음과 같이 구할 수 있다.

$$\begin{aligned} \Pr(E_1) &= \left(\frac{1}{4} + \frac{1}{8}\right) + \left(\frac{1}{4} + \frac{1}{8}\right) \left(\frac{1}{8}\right) + \left(\frac{1}{4} + \frac{1}{8}\right) \left(\frac{1}{8}\right)^2 + \dots \\ &= \sum_{i=0}^{\infty} \frac{3}{8} \left(\frac{1}{8}\right)^i = \frac{3}{8} \left(\frac{1}{1 - \frac{1}{8}}\right) = \frac{3}{7} \end{aligned}$$

같은 방식으로 E_2 가 발생할 확률도 계산할 수 있다.

$$\begin{aligned} \Pr(E_2) &= \frac{1}{2} + \frac{1}{2} \left(\frac{1}{8}\right) + \frac{1}{2} \left(\frac{1}{8}\right)^2 + \dots \\ &= \sum_{i=0}^{\infty} \frac{1}{2} \left(\frac{1}{8}\right)^i = \frac{1}{2} \left(\frac{1}{1 - \frac{1}{8}}\right) = \frac{4}{7} \end{aligned}$$

이산확률변수 Y 에 대하여 사건 E_1 가 발생하면 $Y = 1$, 사건 E_2 가 발생하면 $Y = 2$ 라고 정의하자. 그러면 Y 의 확률분포표는 Table 2와 같다.

Y	1	2	계
$\Pr(Y = y)$	$\frac{3}{7}$	$\frac{4}{7}$	1

Table 2: E_1 발생 시 1, E_2 발생 시 2인 이산확률변수 Y 의 확률분포표

Figure 2의 트리에서 각 leaf 노드를 A, B 대신 E_1 , C 대신 Re , D 대신 E_2 로 표현하면 Figure 3과 같고, 해당 노드에 도달한 순간 발생하는 사건을 나타낼 수 있다. 이 트리에서는 E_1, E_2 가 대응되는 노드에 도달할 시 대응되는 해당 사건이 발생하고, Re 가 대응되는 노드에 도달할 시 처음부터 다시 시작한다고 이야기할 수 있다.

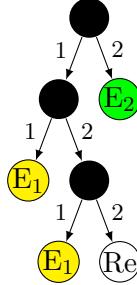


Figure 3: Figure 2의 트리에서 leaf 노드를 E_1, E_2, Re 로 표현한 트리

지금까지의 내용을 정리해보자. $X \sim \text{unif}\{1, 2\}$ 인 X 를 반복적으로 구하여 root 노드에서 시작하여 leaf 노드에 도달할 때까지 대응되는 간선을 통해 자식 노드로 이동하는 규칙을 따른다고 할 때, 각 leaf 노드에 도달하는 경우 발생하는 사건이 정해져 대응되어 있는 Figure 3의 정 이진 트리가 주어지면 해당 트리로부터 Table 2가 나타내는 확률분포를 계산을 통해 구할 수 있다.

매 순간마다 현재 어떤 노드에 있는 지에 따라 X 를 구하는 시행을 해야하는지 아닌지 나타내고 있다는 의미에서 Figure 3의 정 이진 트리를 ‘시행 규칙’이라고 부르겠다. 이를 일반화하여 정의하면 다음과 같다.

Definition. 트리 T 가 다음을 만족할 때 T 를 **시행 규칙**이라고 한다.

1. T 는 정 k 진 트리이다.
2. T 의 모든 노드마다 자식 방향으로의 간선들에 1부터 k 까지의 값이 중복되지 않게 대응되어 있다.
3. T 의 모든 leaf 노드에는 E_i 꼴 또는 Re 가 대응된다. (i 는 자연수)

Definition. 시행 규칙 T 로부터 구한 확률분포란 다음과 같이 정의된다.

T 가 정 k 진 트리일 때, T 의 root 노드에서 시작하여 leaf 노드에 도달할 때까지 $X \sim \text{unif}\{1, k\}$ 인 X 를 구하여 나온 값에 따라 대응되는 자식 노드 방향 간선을 통해 이동하는 것을 반복한다. Re 가 대응되어 있는 leaf 노드에 도달할 경우 root 노드로 돌아가 처음부터 다시 시작한다. E_i 가 대응되어 있는 leaf 노드에 도달할 경우 이동을 종료하며 이때 발생한 사건은 E_i 라고 한다. (i 는 자연수) 최종적으로 이동을 종료하였을 때 발생한 사건이 E_i 일 때 $Y = i$ 인 이산확률변수 Y 를 정의한다. 이때 Y 가 따르는 확률분포가 T 로부터 구한 확률분포가 된다.

시행 규칙으로부터 확률분포를 구하는 과정을 생각해보면, 시행 규칙 상에서 깊이가 같은 leaf 노드들에 대응된 값을 서로 교환해도 해당 시행 규칙으로부터 동일한 확률분포를 구할 수 있다는 걸 알 수 있다.

Theorem 1. 시행 규칙 T 에서 깊이가 같은 leaf 노드들에 대응된 값을 서로 교환하여 얻은 시행 규칙을 T' 라고 할 때, T' 로부터 구할 수 있는 확률분포는 T 로부터 구할 수 있는 확률분포와 동일하다.

Proof. 정 k 진 트리에서 깊이가 d 로 같은 leaf 노드들은 root 노드에서 시작하여 도달할 확률이 $(\frac{1}{k})^d$ 로 모두 동일하기 때문에, 서로 대응된 값을 교환해도 확률분포를 구하는 과정에 영향을 주지 않는다. \square

Theorem 1을 Figure 3의 시행 규칙에 적용해보면, 깊이 3인 두 leaf 노드에 대해 대응값 E_1 과 Re 를 서로 맞바꾼 뒤 계산해도 Table 2가 나온다는 것을 알 수 있다.

3.2 Trial Rule from Probability Distribution

앞에서 시행 규칙 T 로부터 확률분포를 구하는 과정을 보였다. 그렇다면 반대로 특정 이산확률 분포에 대해 해당 확률분포를 구할 수 있는 시행 규칙을 만들어 낼 수 있을까? 이산확률변수가 가질 수 있는 값들의 수가 유한하고, 모든 확률이 유리수라면 이 역시 가능하다는 것을 예를 통해 살펴보자. 이산확률변수 Y 의 확률분포표가 Table 3과 같다고 하자.

Y	1	2	3	계
$\Pr(Y = y)$	$\frac{1}{6}$	$\frac{1}{3}$	$\frac{1}{2}$	1

Table 3: 1, 2, 3일 확률이 각각 $\frac{1}{6}, \frac{1}{3}, \frac{1}{2}$ 인 이산확률변수 Y 의 확률분포표

모든 확률이 유리수이므로 각각의 확률의 비인 $\frac{1}{6} : \frac{1}{3} : \frac{1}{2}$ 와 동일한 자연수비가 존재하며 이 중 $1 : 2 : 3$ 를 택하겠다. 그리고 이 자연수비를 구성하는 숫자들의 합 $1 + 2 + 3 = 6$ 에 대해 $2^h \geq 6$ 을 만족하는 h 값들 중에서 3을 택하겠다. 3을 높이로 하는 완전 이진 트리를 그리면 해당 트리의 leaf 노드 개수는 총 $2^3 = 8$ 개이다. 이제 8개의 leaf 노드 중 왼쪽부터 차례대로 E_1, E_2, E_3 을 각각 1, 2, 3개의 노드에 대응시키고 나머지 노드에는 ‘ Re ’를 대응시키며, 각 노드마다 가지는 자식 방향 간선 2개에 1, 2를 대응시키면 Figure 4와 같은 트리가 된다.

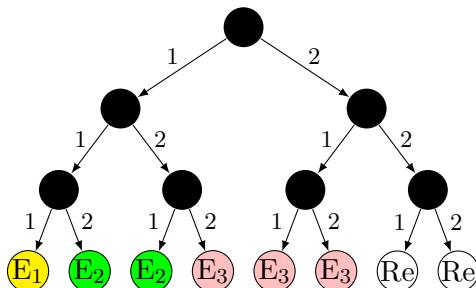


Figure 4: Table 3으로부터 만들어 낸 시행 규칙의 정의를 만족하는 트리

Figure 4의 트리는 시행 규칙의 정의를 만족하므로 확률분포를 구할 수 있다. 우선 root 노드에서 출발하여 leaf 노드에 도달할 때까지 이동을 하는 경우 모든 leaf 노드가 도달 확률이 $\frac{1}{8}$ 로 같다. 따라서 root 노드에서 출발하여 E_1, E_2, E_3, Re 가 대응된 노드에 도달하게 될 확률은 각각 $\frac{1}{8}, \frac{2}{8}, \frac{3}{8}, \frac{2}{8}$ 이 된다. 이제 3.1에서 설명했던 것과 마찬가지 방법으로 E_1, E_2, E_3 가 발생할 확률을 각각 구해보면 다음과 같다.

$$\begin{aligned}\Pr(E_1) &= \sum_{i=0}^{\infty} \frac{1}{8} \left(\frac{1}{4}\right)^i = \frac{1}{8} \left(\frac{1}{1-\frac{1}{4}}\right) = \frac{1}{6} \\ \Pr(E_2) &= \sum_{i=0}^{\infty} \frac{1}{4} \left(\frac{1}{4}\right)^i = \frac{1}{4} \left(\frac{1}{1-\frac{1}{4}}\right) = \frac{1}{3} \\ \Pr(E_3) &= \sum_{i=0}^{\infty} \frac{3}{8} \left(\frac{1}{4}\right)^i = \frac{3}{8} \left(\frac{1}{1-\frac{1}{4}}\right) = \frac{1}{2}\end{aligned}$$

이산확률변수 Y 에 대하여 사건 E_y 가 발생하면 $Y = y$ 라고 정의하고 확률분포를 구하면, Table 3과 동일한 확률분포를 얻을 수 있다. 지금까지의 내용을 일반화하면 다음과 같다.

Theorem 2. 자연수 n , 유리수 p_1, \dots, p_n ($p_1 + \dots + p_n = 1$)에 대하여 다음과 같은 확률분포

$$\Pr(X = i) = p_i \quad (i = 1, \dots, n)$$

가 주어졌을 때, 이 확률분포를 구할 수 있는 정 k 진 트리인 시행 규칙 T 는 다음과 같은 과정을 통해 만들 수 있다.

1. $p_1 : \dots : p_n$ 와 동일한 자연수비 $z_1 : \dots : z_n$ 를 구한다.
2. $k^h \geq \sum_{j=1}^n z_j$ 를 만족하는 자연수 h 를 구한다.
3. 높이가 h 인 완전 k 진 트리 T 를 그리고, T 의 각 노드마다 자식 방향으로의 간선들에 1부터 k 까지의 값을 중복되지 않게 대응시킨다.
4. T 의 leaf 노드들에 E_1, \dots, E_n, Re 를 각각 $z_1, \dots, z_n, k^h - \sum_{j=1}^n z_j$ 개 대응시킨다.

Proof. T 는 높이가 h 인 완전 k 진 트리이므로 정 k 진 트리이며 k^h 개의 leaf 노드를 갖는다. 또한 3.과 4.의 과정으로 인해 시행 규칙의 정의를 만족한다. 따라서 T 로부터 확률분포를 구할 수 있다. root 노드에서 leaf 노드에 도달할 때까지 이동을 할 경우 모든 leaf 노드에 대해 도달 확률이 $\frac{1}{k^h}$ 로 같다. 따라서 root 노드에서 출발하여 E_1, \dots, E_n, Re 가 대응된 노드에 도달하게 될 확률은 각각 $\frac{z_1}{k^h}, \dots, \frac{z_n}{k^h}, \frac{k^h - \sum_{j=1}^n z_j}{k^h}$ 이다.

이제 E_i ($i = 1, \dots, n$)가 발생할 확률을 계산하면 다음과 같다.

$$\begin{aligned}\Pr(E_i) &= \sum_{j'=0}^{\infty} \frac{z_i}{k^h} \left(\frac{k^h - \sum_{j=1}^n z_j}{k^h} \right)^{j'} = \frac{z_i}{k^h} \left(\frac{1}{1 - \frac{k^h - \sum_{j=1}^n z_j}{k^h}} \right) = \frac{z_i}{k^h} \left(\frac{k^h}{\sum_{j=1}^n z_j} \right) \\ &= \frac{z_i}{\sum_{j=1}^n z_j}\end{aligned}$$

$p_1 : \dots : p_n = z_1 : \dots : z_n$ |므로 $z_j = cp_j$ 로 표현할 수 있고, 또 $\sum_{j=1}^n p_j = 1$ |므로

$$\Pr(E_i) = \frac{z_i}{\sum_{j=1}^n z_j} = \frac{cp_i}{\sum_{j=1}^n cp_j} = \frac{cp_i}{c \sum_{j=1}^n p_j} = \frac{p_i}{\sum_{j=1}^n p_j} = p_i$$

따라서 이산화률변수 X 에 대하여 사건 E_i 가 발생하면 $X = i$ 라고 정의하면 X 는 처음에 주어졌던 확률분포를 따르게 된다. \square

Theorem 2은 처음에 제시했던 문제인 ‘보드게임에서 한정된 도구로 확률적 요소의 구현을 어떻게 할 것인가’에 대해 기본적인 해결책을 제공한다. 해당 정리의 과정을 통해, 사용 결값이 이산균등분포를 따르는 도구가 있을 때 해당 도구만 사용하여 원하는 확률 분포를 구할 수 있는 시행 규칙을 만들 수 있다. 보드게임 창작자는 사용 결값이 이산균등분포를 따르는 도구인 동전, 주사위 등과 필요한 시행 규칙을 게임의 구성 요소로 제공함으로써 사용자가 필요할 때 확률적 요소를 이용하도록 만들 수 있다.

3.3 Trial Number Expectation

사용자가 주사위를 굴리며 실제로 시행 규칙을 사용하는 상황을 생각해보자. 우선 시행 규칙의 root 노드에서 이동을 시작한다. 주사위를 굴리고 나온 눈의 수에 따라 대응되는 간선을 통해 자식 노드로 이동한다. leaf 노드에 도달할 때까지 이를 반복한다. 만약 leaf 노드에 도달했는 데 Re가 대응되어 있는 노드라면 처음부터 다시 시작해야 한다. 사용자는 주사위를 굴리고, 이동하는 것을 반복하다 E_i 꼴이 대응되어 있는 leaf 노드에 도달하고 나서야 어떤 사건이 발생했는지 결정하고 주사위를 굴리는 것을 멈출 수 있다.

만약 어떤 사건이 발생했는지 결정될 때까지 주사위를 굴리는 횟수가 너무 많아진다면 시간적인 비용이 발생하게 되고 이는 사용자에게 분명 좋지 못한 경험이 될 것이다. 따라서 어떤 시행 규칙이 주어졌을 때, 사용 결값이 이산균등분포를 따르는 도구를 사용하는 시행을 평균적으로 몇 번 실시해야만 어떤 사건이 발생했는지 결정되는가를 살펴볼 필요가 있다.

간단한 표현을 위해, 지금부터 ‘사용 결跬값이 이산균등분포를 따르는 도구’를 ‘시행 도구’라고 하고 ‘어떤 사건이 발생했는지 결정’되는 것을 ‘최종 결과가 결정’되는 것이라고 하겠다. 보드게임의 주사위나 동전 등은 시행 도구이고, 이를 던져서 무슨 면이 나왔는 지 관측하는 행위는 시행 도구를 사용하는 시행이며, 이를 통해 시행 규칙 상에서 이동을 진행하다가 최종적으로 어떤 사건 E_i 가 발생하면 최종 결과가 결정되는 것이다.

시행 규칙이 주어지고, 최종 결과가 결정될 때까지 시행 도구를 사용하는 시행의 횟수를 이산확률변수 X 로 두면 X 의 기댓값 $\mathbf{E}[X]$ 이 바로 시행 도구를 사용하는 시행의 평균 횟수가 된다. 시행 규칙 하나를 예로 들어 $\mathbf{E}[X]$ 를 어떻게 구할 수 있는지 살펴보자.

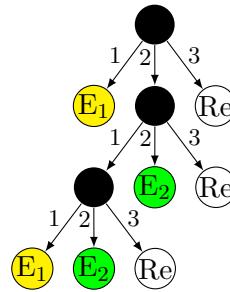


Figure 5: 사용하는 시행 도구 결跬값이 $\text{unif}\{1, 3\}$ 을 따르는 경우의 시행 규칙

Figure 5의 시행 규칙에서 최종 결과가 결정될 때까지 시행 도구를 사용하는 시행의 횟수를 이산확률변수 X 로 두자. X 가 가질 수 있는 값은 모든 자연수이며 각각의 경우에 대해 가능한 시행 도구 결跬값 순서와 확률을 정리하면 Table 4와 같다.

X	1	2	3	...
가능한 결跬값 순서	(1)	(2,1,1) (2,2) (3,1)	(2,1,2) (2,3,1) (3,2,2)	...
$\Pr(X = x)$	$\frac{1}{3}$	$\frac{2}{9}$	$\frac{5}{27}$...

Table 4: 이산확률변수 X 가 시행 도구 사용 횟수일 때 각각의 경우에 대해 정리한 표

시행 규칙 상 Re 의 존재로 인해 x 의 범위가 무한하고, x 가 커질 수록 $\Pr(X = x)$ 을 직접 구하기가 매우 어려워진다. 그러므로 $\mathbf{E}[X]$ 의 값을 구하기 위해선 각각의 경우의 확률을 구하는 대신 다른 방법을 사용하여야 한다.

Figure 5의 시행 규칙에서 leaf 노드의 총 개수는 7개이다. 이 7개의 노드에 번호를 붙이자.

깊이 1, 3인 E_1 대응 노드에는 1, 2번을, 깊이 2, 3인 E_2 대응 노드에는 3, 4번을, 깊이 1, 2, 3인 Re 대응 노드에는 5, 6, 7번을 붙이겠다. root 노드에서 시작하여 맨 처음에 도달하는 leaf 노드의 번호가 i 인 사건을 F_i 라고 하면 전체 기댓값의 법칙에 의해 다음이 성립한다.

$$\mathbf{E}[X] = \sum_{i=1}^7 \Pr(F_i) \mathbf{E}[X|F_i] \quad (1)$$

각각의 leaf 노드의 깊이를 표현하기 위해 다음을 정의하자.

Definition. F_i 가 시행 규칙의 root 노드에서 시작하여 맨 처음 어떤 leaf 노드에 도달하는 사건일 때, $depth(F_i)$ 는 F_i 에서 도달하는 노드의 깊이이다.

(1)에서 $i = 1, 2, 3, 4$ 인 경우는 root 노드에서 시작하여 맨 처음에 도달한 leaf 노드가 Re 가 대응된 노드가 아닌 경우이다. 따라서 도달한 노드의 깊이가 곧 X 의 값이 되므로 다음과 같다.

$$\begin{aligned}\mathbf{E}[X|F_1] &= depth(F_1) \\ \mathbf{E}[X|F_2] &= depth(F_2) \\ \mathbf{E}[X|F_3] &= depth(F_3) \\ \mathbf{E}[X|F_4] &= depth(F_4)\end{aligned}$$

$i = 5, 6, 7$ 인 경우는 root 노드에서 시작하여 맨 처음에 도달한 leaf 노드가 Re 가 대응된 노드인 경우이다. 이 경우에는 다음 정리를 사용할 수 있다.

Theorem 3. F_i 가 시행 규칙의 root 노드에서 시작하여 맨 처음 Re 가 대응된 어떤 leaf 노드에 도달하는 사건일 때,

$$\mathbf{E}[X|F_i] = \mathbf{E}[X] + depth(F_i)$$

Proof.

$$\mathbf{E}[X|F_i] = \sum_{x=1} x \Pr(X = x | F_i)$$

간단한 표현을 위해 $d = depth(F_i)$ 라 하고, $x' = x - d$ 로 두면

$$\sum_{x=1} x \Pr(X = x | F_i) = \sum_{x'=1-d} (x' + d) \Pr(X = x' + d | F_i)$$

F_i 이 발생했다는 것은 시행 규칙의 root 노드에서 시작하여 맨 처음에 깊이가 $depth(F_i)$ 인 Re 가 대응된 노드에 도달했다는 의미이다. 그러므로 F_i 이 발생했다는 전제 하에 X 가 특정 값을

가질 사건의 조건부확률을 구하는 것은 시작 직후 시행 도구를 사용하는 시행을 $depth(F_i)$ 번 실시한 뒤 Re가 대응된 노드에 도달하여 처음부터 다시 시작한 경우에 한정해서 확률을 구하는 것과 같다. 따라서 다음과 같다.

$$\Pr(X = x | F_i) = \begin{cases} 0 & (x \leq depth(F_i)) \\ \Pr(X = x - depth(F_i)) & (x > depth(F_i)) \end{cases}$$

이를 이용하여 정리하면

$$\begin{aligned} \sum_{x'=1-d} (x' + d) \Pr(X = x' + d | F_i) &= \sum_{x'=1} (x' + d) \Pr(X = x' + d | F_i) \\ &= \sum_{x'=1} (x' + d) \Pr(X = x') \\ &= \sum_{x'=1} x' \Pr(X = x') + \sum_{x'=1} d \Pr(X = x') \\ &= \mathbf{E}[X] + d \end{aligned}$$

□

Theorem 3를 사용하면 다음과 같다.

$$\begin{aligned} \mathbf{E}[X|F_5] &= \mathbf{E}[X] + depth(F_5) \\ \mathbf{E}[X|F_6] &= \mathbf{E}[X] + depth(F_6) \\ \mathbf{E}[X|F_7] &= \mathbf{E}[X] + depth(F_7) \end{aligned}$$

이제 (1)을 정리해보자. 간단한 표현을 위해 $p_i = \Pr(F_i)$, $d_i = depth(F_i)$ 라 하면

$$\begin{aligned} \mathbf{E}[X] &= \sum_{i=1}^7 \Pr(F_i) \mathbf{E}[X|F_i] \\ &= p_1 d_1 + p_2 d_2 + p_3 d_3 + p_4 d_4 + p_5 (\mathbf{E}[X] + d_5) + p_6 (\mathbf{E}[X] + d_6) + p_7 (\mathbf{E}[X] + d_7) \end{aligned}$$

$\mathbf{E}[X]$ 항을 좌변으로 모아 정리하면

$$(1 - (p_5 + p_6 + p_7)) \mathbf{E}[X] = p_1 d_1 + p_2 d_2 + p_3 d_3 + p_4 d_4 + p_5 d_5 + p_6 d_6 + p_7 d_7$$

$$\sum_{i=1}^7 p_i = 1 \circ] \text{므로}$$

$$\begin{aligned}\mathbf{E}[X] &= \frac{p_1d_1 + p_2d_2 + p_3d_3 + p_4d_4 + p_5d_5 + p_6d_6 + p_7d_7}{1 - (p_5 + p_6 + p_7)} \\ &= \frac{p_1d_1 + p_2d_2 + p_3d_3 + p_4d_4 + p_5d_5 + p_6d_6 + p_7d_7}{p_1 + p_2 + p_3 + p_4} \\ &= \frac{\sum_{i=1}^7 p_i d_i}{\sum_{i=1}^4 p_i}\end{aligned}$$

이렇게 구한 식에 실제 값을 넣어서 계산하면

$$\begin{aligned}\mathbf{E}[X] &= \frac{\frac{1}{3} \cdot 1 + \frac{1}{27} \cdot 3 + \frac{1}{9} \cdot 2 + \frac{1}{27} \cdot 3 + \frac{1}{3} \cdot 1 + \frac{1}{9} \cdot 2 + \frac{1}{27} \cdot 3}{\frac{1}{3} + \frac{1}{27} + \frac{1}{9} + \frac{1}{27}} \\ &= \frac{39}{14} = 2.7857...\end{aligned}$$

따라서 Figure 5의 시행 규칙이 주어졌을 때, 결과가 결정될 때까지 시행 도구를 사용하는 시행을 평균 2.7857...회 하여야 한다는 것을 알 수 있다. $\mathbf{E}[X]$ 를 나타내는 용어를 정의하고 지금까지의 내용을 일반화하면 다음과 같다.

Definition. 시행 규칙 T 에 대해, root 노드에서 시작하여 결과가 결정될 때까지 시행 도구를 사용한 시행 횟수를 확률변수 X 라고 하자. 이때 $\mathbf{E}[X]$ 를 T 의 시행 횟수 기댓값이라고 한다.

Theorem 4 (Trial Number Expectation). 시행 규칙 T 의 leaf 노드의 수가 l 개이고 그 중 a 개가 Re 가 대응되지 않은 노드라고 하자. Re 가 대응되지 않은 leaf 노드들에는 1부터 a 까지, Re 가 대응된 leaf 노드들에는 그 다음부터 l 까지 번호를 붙인 뒤 T 의 root 노드에서 시작하여 맨 처음에 도달하는 leaf 노드의 번호가 i 인 사건을 F_i 라고 하자. 이때 T 의 시행 횟수 기댓값 $\mathbf{E}[X]$ 는 다음과 같다.

$$\mathbf{E}[X] = \frac{\sum_{i=1}^l \Pr(F_i) \operatorname{depth}(F_i)}{\sum_{i=1}^a \Pr(F_i)}$$

Proof.

$$\begin{aligned}
\mathbf{E}[X] &= \sum_{i=1}^l \Pr(F_i) \mathbf{E}[X|F_i] \\
&= \sum_{i=1}^a \Pr(F_i) \mathbf{E}[X|F_i] + \sum_{i=a+1}^l \Pr(F_i) \mathbf{E}[X|F_i] \\
&= \sum_{i=1}^a \Pr(F_i) \text{depth}(F_i) + \sum_{i=a+1}^l \Pr(F_i) (\mathbf{E}[X] + \text{depth}(F_i))
\end{aligned}$$

$\mathbf{E}[X]$ 항을 좌변으로 모아 정리하면

$$\begin{aligned}
\left(1 - \sum_{i=a+1}^l \Pr(F_i)\right) \mathbf{E}[X] &= \sum_{i=1}^l \Pr(F_i) \text{depth}(F_i) \\
\sum_{i=1}^l \Pr(F_i) &= 1 \circ] \text{므로} \\
\mathbf{E}[X] &= \frac{\sum_{i=1}^l \Pr(F_i) \text{depth}(F_i)}{\sum_{i=1}^a \Pr(F_i)}
\end{aligned}$$

□

Theorem 4의 식에서 분모는 root 노드에서 시작하여 맨 처음에 도달하는 leaf 노드가 Re가 대응되지 않은 노드일 확률을 의미한다. 따라서 어떤 시행 규칙이 주어졌을 때, 시행 규칙의 leaf 노드들의 깊이가 전체적으로 작을 수록 그리고 leaf 노드들 중 Re가 대응되지 않은 노드들의 비중이 클 수록 해당 시행 규칙의 시행 횟수 기댓값이 더 작은 경향을 띠게 된다.

3.4 Node Merging

Figure 4의 시행 규칙을 다시 살펴보면, root 노드에서 시작하는 경우 시행 도구 결괏값 순서가 (2, 1, 1)일 때와 (2, 1, 2)일 때 모두 E_3 이 대응된 노드에 도달한다는 것을 알 수 있다. 즉 처음 두 번의 시행 도구 결괏값 순서가 (2, 1)이라면 그 다음 시행 결과가 1이든 2이든 상관없이 E_3 이 확정적으로 발생하게 된다. 이는 처음 두 번의 시행 도구 결괏값 순서가 (2, 1)이 나왔을 때 그냥 시행을 종료하고 E_3 이 발생한 것으로 해도 다를 바가 없다는 의미이다. 이를 시행 규칙에 반영하여 다시 표현하면 Figure 6과 같다.

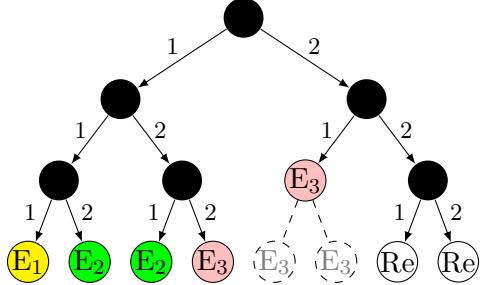


Figure 6: Figure 4의 시행 규칙을 처음 두 번의 시행 도구 결괏값 순서가 (2, 1)인 경우 E_3 대응 노드에 바로 도달하도록 바꾼 시행 규칙

Figure 6의 시행 규칙에서 Re 가 대응되는 노드 2개에 대해서도 동일한 과정을 적용할 수 있다. 이제 이러한 과정을 일반화하여 정의하고, 이 과정을 통해 시행 횟수 기댓값이 얼마나 줄어드는지 정리하면 다음과 같다.

Definition. 정 k 진 트리인 시행 규칙 T 에서 *leaf* 노드 k 개가 부모가 같고 모두 동일한 대응 값을 가지는 경우, 해당 노드들을 모두 제거하여 부모 노드를 새 *leaf* 노드로 만들고 제거된 노드들에 대응되어 있던 값을 새 *leaf* 노드에 대응시키는 과정을 **노드 합치기**라고 한다.

Theorem 5. T' 가 시행 규칙 T 에서 노드 합치기를 통해 얻어진 시행 규칙일 때, T' 로부터 구할 수 있는 확률분포는 T 로부터 구할 수 있는 확률분포와 동일하다.

Proof. 시행 규칙 T 가 정 k 진 트리일 때, 노드 합치기의 대상이 되는 노드는 총 k 개이다. 대상 노드들의 부모 노드까지 처음에 root 노드에서 시작하여 도달할 확률을 p 라고 하면, 대상 노드들 각각에 도달할 확률은 모두 $p \cdot \frac{1}{k}$ 이다. 합치기 전에는 k 개의 *leaf* 노드 각각에 대해 $p \cdot \frac{1}{k}$ 의 확률로 도달하므로 각각의 노드들의 도달 확률의 합이 p 이다. 그리고 해당 노드들의 부모 노드까지 처음에 root 노드에서 시작하여 도달할 확률을 p 라고 하였으므로, 합치기 후의 새 *leaf* 노드 도달 확률은 p 이다. 합치기를 수행한 부분의 도달 확률이 합치기 전과 후에 달라지지 않으므로 시행 규칙에서 확률 분포를 계산하는 과정도 달라지지 않는다. \square

Theorem 6. 시행 규칙 T 의 *leaf* 노드 중 Re 가 대응되지 않은 노드가 a 개라고 하자. 해당 *leaf* 노드들에 1부터 a 까지 번호를 붙이고, T 의 root 노드에서 시작하여 맨 처음에 도달하는 *leaf* 노드의 번호가 i 인 사건을 F_i 라고 하자. T 에서 노드 합치기를 수행할 때, root 노드에서 시작하여 합치기 대상인 노드들의 부모 노드까지 도달할 확률을 p , 노드 합치기 후 얻어진 시행 규칙을 T' 라 하자. T 의 시행 횟수 기댓값을 $\mathbf{E}[X]$, T' 의 시행 횟수 기댓값을 $\mathbf{E}[X']$ 라 할 때

$$\mathbf{E}[X'] = \mathbf{E}[X] - \frac{p}{\sum_{i=1}^a \Pr(F_i)}$$

Proof. Theorem 4를 이용하여 $\mathbf{E}[X]$ 와 $\mathbf{E}[X']$ 각각을 구한 뒤 차를 계산하면 된다. 두 분수 값의 계산 과정에서 분모는 서로 동일하게 구해지는데, 이는 Theorem 4의 식에서 분모가 의미하는 것이 root 노드에서 시작하여 맨 처음 도달하는 leaf 노드가 Re가 대응되지 않은 노드일 확률이기 때문에 노드 합치기를 통해서 바뀌지 않기 때문이다. 분자의 경우 $\mathbf{E}[X]$ 에서는 합치기 대상인 노드들을 의미하는, 확률 $\frac{1}{k} p$ 와 깊이 d (합치기 대상인 노드들의 깊이)가 곱해진 항이 k 개 존재하고 $\mathbf{E}[X']$ 에서는 해당 항들이 사라진 대신 합친 후의 새 leaf 노드를 의미하는, 확률 p 와 깊이 $d-1$ 가 곱해진 항이 1개 존재한다. 그러므로 분자의 차는 $k \frac{1}{k} pd - p(d-1) = p$ 이고, 따라서

$$\mathbf{E}[X] - \mathbf{E}[X'] = \frac{p}{\sum_{i=1}^a \Pr(F_i)}$$

□

Theorem 1에 의해 시행 규칙 상에서 같은 깊이를 가지는 leaf 노드들의 대응값은 해당 노드들 내에서 자유롭게 재배치될 수 있다. 따라서 어떤 시행 규칙이 주어졌을 때, 깊이가 같고 동일한 대응값을 가지는 leaf 노드들이 같은 부모를 가지도록 재배치한 뒤 노드 합치기를 진행함으로써, 구할 수 있는 확률 분포는 동일하고 시행 횟수 기댓값은 더 작은 시행 규칙을 얻을 수 있다. 이 과정을 계속 반복하여 결국 시행 규칙 상에서 같은 깊이를 가지는 leaf 노드끼리 어떻게 재배치하더라도 더 이상 노드 합치기를 할 수 없는 상태가 되었다고 한다면, 이는 처음에 주어진 시행 규칙에 대해 시행 횟수 기댓값을 줄이는 과정을 최대한 수행하였다고 표현할 수 있다.

3.5 Finding Minimum $\mathbf{E}[X]$

Theorem 2의 과정을 따르면 이산확률변수가 유한한 값을 가지며 각 값을 가질 확률이 모두 유리수인 확률분포가 주어졌을 때, 해당 확률분포를 구할 수 있는 시행 규칙을 만들 수 있다. 그런데 이 과정에서 중간에 z_1, \dots, z_n , 그리고 h 를 어떤 값을 취하느냐에 따라서 만들어지는 시행 규칙의 형태가 달라진다. 따라서 이렇게 만들어지는 모든 시행 규칙들 각각에 대해 시행 횟수 기댓값을 줄이는 과정을 최대한 수행하고, 최종적으로 구해지는 시행 횟수 기댓값들을 서로 비교하여 그 중 가장 값이 작은 경우의 z_1, \dots, z_n, h 값 조합을 찾는 과정을 생각해볼 수 있다. 이것은 다시 말해 특정 확률분포를 구할 수 있는 시행 규칙들 중 시행 횟수 기댓값이 가장 작은 시행 규칙을 찾는 과정이다.

Theorem 2에서 z_1, \dots, z_n 의 가능한 모든 조합들 중 $\sum_{j=1}^n z_j$ 이 최소일 때는 z_1, \dots, z_n 의

최대공약수가 1일 때이다. 이 때의 z_1, \dots, z_n 값을 $\alpha_1, \dots, \alpha_n$ 라고 하면, z_1, \dots, z_n 의 가능한 모든 조합은 $c\alpha_1, \dots, c\alpha_n$ ($c = 1, 2, \dots$)로 표현됨을 알 수 있다.

c 가 결정되면 Theorem 2의 2.를 만족하는 h 의 범위가 정해진다. 3.2에서 예로 주어진 확률분포(Table 3)에 대해 지금까지의 내용을 적용하여 정리해보면 다음과 같다.

c	z_1	z_2	z_3	$\sum_{i=1}^3 z_i$	$2^h \geq \sum_{i=1}^3 z_i$ 인 h 의 범위
1	1	2	3	6	$h \geq 3$
2	2	4	6	12	$h \geq 4$
3	3	6	9	18	$h \geq 5$
4	4	8	12	24	$h \geq 5$
5	5	10	15	30	$h \geq 5$
6	6	12	18	36	$h \geq 6$
				⋮	

Table 5: c 에 따른 z_1, z_2, z_3 의 값과 가능한 h 의 범위를 나타낸 표

Table 5에서 $c = 1$ 이고 h 가 최솟값인 3인 경우는 3.2에서 살펴보았으며, Figure 4의 시행 규칙을 얻을 수 있었다. 만약 $c = 1$ 이고 h 가 4인 경우는 다음과 같은 시행 규칙을 얻을 수 있다.

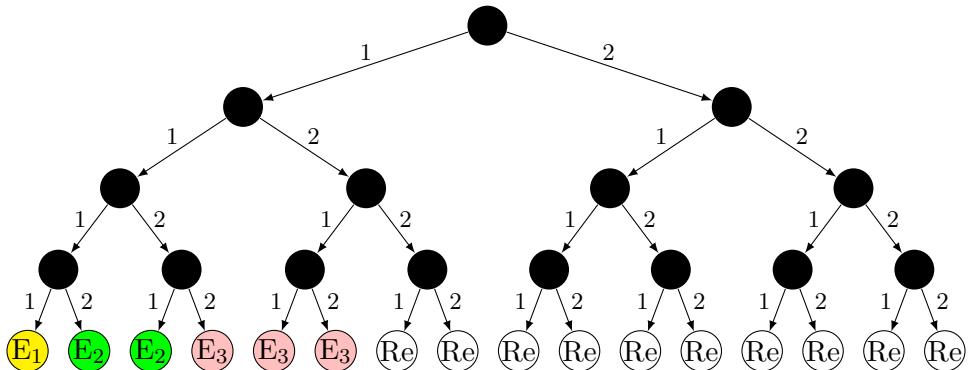


Figure 7: $c = 1, h = 4$ 인 경우 얻을 수 있는 시행 규칙

3.3의 마지막 문단에서 다루었듯이, leaf 노드들의 깊이가 작을 수록 그리고 leaf 노드들 중 Re가 대응되지 않은 노드들의 비중이 클 수록 시행 규칙의 시행 횟수 기댓값이 작은 경향을 띤다. 따라서 Figure 4와 Figure 7의 두 시행 규칙에 대해 시행 횟수 기댓값을 줄이는 과정을 최대한 수행하고 나면, Figure 4의 시행 규칙이 시행 횟수 기댓값이 더 작아진다는 절 형태를 통해 직관적으로 알 수 있다. 다시 말해, Table 5에서 $c = 1$ 인 경우 $h = 3$ 일 때 최종적으로

구해지는 시행 규칙이 $h = 4$ 일 때 최종적으로 구해지는 시행 규칙보다 더 시행 횟수 기댓값이 작다는 것이다. 계산을 통해 이를 일반화하여 정리하면 다음과 같다.

Theorem 7. Theorem 2의 과정에서 특정 z_1, \dots, z_n, h 조합을 선택하여 만들어낸 시행 규칙이 시행 횟수 기댓값을 줄이는 과정을 최대한 수행하여 최종적으로 구해지는 시행 규칙 T 의 시행 횟수 기댓값을 $\mathbf{E}[X]$ 라고 하자. z_1, \dots, z_n 는 그대로 두고 h 만 $h+1$ 로 바꾼 뒤 동일한 과정을 통해 최종적으로 구해지는 시행 규칙 T' 의 시행 횟수 기댓값을 $\mathbf{E}[X']$ 라고 하면 $\mathbf{E}[X']$ 는 $\mathbf{E}[X]$ 보다 항상 크다.

Proof. T 가 k 진 트리라고 하자. Theorem 2의 과정에서 특정 z_1, \dots, z_n, h 조합에 대해 h 만 $h+1$ 로 증가시킨다는 것은 Theorem 2의 과정이 끝난 직후 만들어지는 시행 규칙의 leaf 노드의 수가 k^h 개에서 k^{h+1} 개로 늘어난다는 의미이다. 이때 추가되는 $(k-1)k^h$ 개의 leaf 노드들은 모두 깊이가 $h+1$ 이고 Re가 대응되므로, 시행 규칙의 시행 횟수 기댓값을 줄이는 과정을 최대한 수행하고 나면 깊이가 1인 $k-1$ 개의 Re 대응 노드가 된다. 결론적으로 T' 는 T 의 root 노드에 형제 노드로 $k-1$ 개의 Re 대응 노드를 추가하고 새로운 부모 노드를 붙여서 만든 시행 규칙이다. T' 를 그림으로 나타내면 다음과 같다.

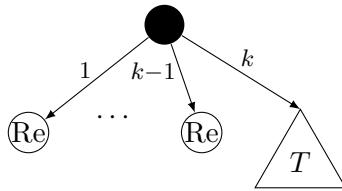


Figure 8: 시행 규칙 T'

T 의 leaf 노드 개수를 l 이라고 하자. T 의 root 노드에서 시작하여 맨 처음에 도달할 확률, 깊이를 p_i, d_i ($i = 1, \dots, l$)라 하고, leaf 노드들 중 Re가 아닌 노드들의 p_i 값의 합을 P 라고 하면 Theorem 4에 의해

$$\mathbf{E}[X] = \frac{\sum_{i=1}^l p_i d_i}{P}$$

Figure 8에서 T' 을 살펴보면 T 의 모든 노드에 대해 root 노드에서 시작하여 맨 처음에 도달할 확률이 $\frac{1}{k}$ 배가 되었고, 깊이는 1 증가한 것을 알 수 있다. 또한 추가된 Re 대응 노드 $k-1$ 개는 root 노드에서 시작하여 맨 처음에 도달할 확률이 $\frac{1}{k}$ 이고 깊이는 1이다. 따라서

$$\mathbf{E}[X'] = \frac{(k-1)\frac{1}{k}1 + \sum_{i=1}^l \frac{1}{k}p_i(d_i + 1)}{\frac{1}{k}P}$$

분자와 분모에 k 를 곱하고 정리하면

$$\begin{aligned}
 \mathbf{E}[X'] &= \frac{k - 1 + \sum_{i=1}^l p_i(d_i + 1)}{P} \\
 &= \frac{k - 1 + \sum_{i=1}^l p_i d_i + \sum_{i=1}^l p_i}{P} \\
 &= \frac{\sum_{i=1}^l p_i d_i + k}{P} \\
 &= \mathbf{E}[X] + \frac{k}{P}
 \end{aligned}$$

□

Theorem 7은 Theorem 2의 과정에서 일단 특정 z_1, \dots, z_n 조합을 선택하고 나면, h 가 커질 수록 최종적으로 구해지는 시행 규칙의 시행 횟수 기댓값도 커진다는 것을 나타낸다. 이는 다시 말해 h 가 최소일 때 시행 횟수 기댓값이 최소가 된다는 것을 뜻한다.

Corollary 7.1. *Theorem 2의 과정에서 특정 z_1, \dots, z_n 조합이 결정되었을 때, h 를 가능한 최솟값을 택하여 시행 규칙을 만든 경우에 해당 시행 규칙에 시행 횟수 기댓값을 줄이는 과정을 최대한 수행하여 최종적으로 구한 시행 규칙의 시행 횟수 기댓값이 최소가 된다.*

Corollary 7.1이 의미하는 것은, 특정 확률분포를 구할 수 있는 시행 규칙들 중 시행 횟수 기댓값이 가장 작은 시행 규칙을 찾는 경우에 Theorem 2의 과정에서 h 가 최소인 경우 하나만 확인해보면 충분하다는 것이다. 예를 들어 Table 5에서 $c = 3$ 일 때는 $h = 5$ 인 경우 하나만 확인하면 충분하다.

시행 횟수 기댓값이 가장 작은 시행 규칙을 찾는 경우에 대해 한 가지 더 확인하고 넘어가자. Theorem 2의 과정에서 z_1, \dots, z_n 가 모두 k 의 배수인 경우, 만들어지는 시행 규칙의 모든 leaf 노드를 k 개씩 모아 노드 합치기를 수행할 수 있다. 노드 합치기를 수행하고 나면 각 대응 값이 대응된 노드 수가 전부 $\frac{1}{k}$ 배이고 높이가 1 작은 새로운 포화 k 진 트리 시행 규칙을 얻을 수 있는데, 이 시행 규칙 또한 Theorem 2의 과정을 통해 만들어질 수 있다. 다시 말해 Table 5에서와 같은 방식으로 c 의 값을 1부터 차례대로 증가시켜가며 Theorem 2를 적용할 경우, c 의 값이 kc_0 일 때와 c_0 일 때 (c_0 는 자연수) 동일한 시행 규칙이 만들어진다. 따라서 c 가 k 의 배수인 경우는 중복이므로 건너뛰어도 된다. Figure 9를 보면 Table 5에서 $c = 2, h = 4$ 인 경우 구해지는 시행 규칙에서 최하단 leaf 노드들에 대해 전부 합치기를 수행한 결과 $c = 1, h = 3$ 인 경우 구해지는 시행 규칙인 Figure 4와 같아지는 것을 확인할 수 있다.

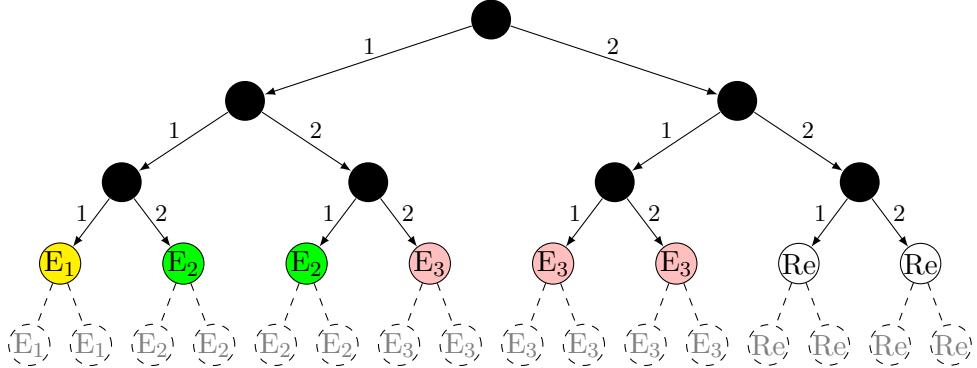


Figure 9: $c = 2, h = 4$ 인 경우 얻을 수 있는 시행 규칙의 최하단 leaf 노드들에 대해 전부 합치기를 수행한 결과

지금까지 특정 확률분포를 구할 수 있는 시행 규칙들 중 시행 횟수 기댓값이 가장 작은 시행 규칙을 찾기 위해 어떤 경우들을 조사하여야 하는지 살펴보았다. 해당 경우들을 차례대로 조사하는 프로그램을 제작하고 보드게임 창작자가 이를 사용한다면, 창작자는 자신이 원하는 확률적 요소를 특정 도구로 구현하는 가장 좋은 방법을 빠르고 간편하게 알아낼 수 있을 것이다.

3.6 k -ary Numeral System for Computation

3.5에서 특정 확률분포를 구할 수 있는 시행 규칙들 중 시행 횟수 기댓값이 가장 작은 시행 규칙을 찾기 위해 어떤 경우들을 조사하여야 하는지 다루었다. 그렇다면 이제 각각의 경우를 조사하는 방법 자체에 대해 살펴보자.

3.4에서 노드 합치기를 통해 어떤 시행 규칙의 시행 횟수 기댓값을 줄이는 과정에 대해서 설명하였다. 이 과정의 핵심은 대응값이 동일하고 깊이가 같은 leaf 노드 k 개가 존재한다면 동일한 대응값을 가지며 깊이가 하나 작은 노드 1개로 합쳐질 수 있다는 것이다. 구체적인 예를 하나 들어보자. Theorem 2의 과정을 통해 얻은 시행 규칙이 높이 h 인 포화 3진 트리이고, E_1 가 대응되는 leaf 노드의 개수가 14개라고 하자. 시행 횟수 기댓값을 줄이기 위해 노드 합치기를 수행하는 과정에서 E_1 대응 노드들의 배치는 다음과 같이 변화한다.

깊이 h 에 14개

\Downarrow 깊이 h 인 노드 합치기 4회 (4×3)

깊이 $h - 1$ 에 4개, 깊이 h 에 2개

\Downarrow 깊이 $h - 1$ 인 노드 합치기 1회 (1×3)

깊이 $h - 2$ 에 1개, 깊이 $h - 1$ 에 1개, 깊이 h 에 2개

특정 깊이에서 노드 3개가 합쳐지면 깊이가 하나 작은 노드 1개가 된다. 그런데 이는 어떤 숫자를 3진법으로 적는 과정에서 특정 자리에 3이 올 경우 다음 자리로 올려 1로 표기하는 것과 구조적으로 동일하다. 실제로 숫자 14를 3진법으로 표현하면 $112_{(3)}$ 인데, 이를 표현하는 과정은 위에서의 E_1 대응 노드 배치 변화 과정과 유사하며 최종 결과에서 숫자의 배열도 같은 것을 확인할 수 있다. 따라서 주어진 시행 규칙에서 시행 횟수 기댓값을 줄이는 과정을 최대한 수행한 뒤의 E_1 대응 노드의 배치는 $112_{(3)}$ 라는 표현으로 나타낼 수 있다.

이를 일반화하여 정리해보자. Theorem 2의 과정을 통해 만들어진 시행 규칙 T 가 k 진 트리이고 leaf 노드의 가능한 모든 대응값이 E_1, \dots, E_n, Re 로 총 $n+1$ 종류일 때, 시행 규칙 T 에서 각각의 개수를 k 진법으로 표현한 숫자 $n+1$ 개를 만들 수 있다. 이 k 진법 숫자들은 T 의 시행 횟수 기댓값을 줄이는 과정을 최대한 수행한 후 최종적으로 구해지는 시행 규칙 T' 에서 각각의 깊이에 어떤 종류의 leaf 노드가 몇 개나 있는지를 나타낸다. 따라서 해당 숫자들만 보면 T' 를 그리지 않아도 T' 의 시행 횟수 기댓값을 계산할 수 있다. 이는 T 에서 T' 까지 변화하는 과정을 일일히 그려서 따져보고 진행하는 대신 $n+1$ 개의 숫자들에 대해 k 진법 변환만 하고나면 T' 의 시행 횟수 기댓값을 계산할 수 있다는 것을 의미한다. 그리고 이 과정에서 얻은 k 진법 숫자들을 보고 T' 를 바로 그리는 것도 가능하다.

지금까지의 내용을 Table 5에서 $c = 5, h = 5$ 인 경우에 대해 적용하여 실제로 시행 횟수 기댓값을 계산해보고, 시행 횟수 기댓값을 줄이는 과정이 최대한 수행된 상태의 시행 규칙을 바로 그려보자. 해당 경우에 Theorem 2를 통해 만들어지는 시행 규칙을 T 라고 하고, T 의 시행 횟수 기댓값을 줄이는 과정을 최대한 수행한 후 최종적으로 구해지는 시행 규칙을 T' 라고 하자. T 는 높이가 5인 포화 이진 트리이며, leaf 노드 $2^5 = 32$ 개 중 E_1, E_2, E_3, Re 가 대응된 노드의 개수가 각각 5, 10, 15, 그리고 $32 - (5 + 10 + 15) = 2$ 개이다. 이 숫자들을 이진법으로 바꾸어 표현한 결과와 이를 통해 알 수 있는, T' 에서 깊이 별로 존재하는 E_1, E_2, E_3, Re 대응 노드 수를 정리하면 다음과 같다.

leaf 노드 종류	T 에서의 개수	T' 에서의 개수				
		깊이 1	깊이 2	깊이 3	깊이 4	깊이 5
E_1	$5 = 00101_{(2)}$	0	0	1	0	1
E_2	$10 = 01010_{(2)}$	0	1	0	1	0
E_3	$15 = 01111_{(2)}$	0	1	1	1	1
Re	$2 = 00010_{(2)}$	0	0	0	1	0
계	32	0	2	2	3	2

Table 6: leaf 노드 종류 별로 T 와 T' 에서의 개수를 정리한 표

T' 에서 어떤 leaf 노드의 깊이가 d 인 경우, root 노드에서 시작하여 맨 처음에 도달하는 leaf 노드가 해당 노드일 확률은 $(\frac{1}{2})^d$ 이다. 이를 이용하여 T' 에 대해 Theorem 4의 공식을 적용하면 T' 의 시행 횟수 기댓값 $\mathbf{E}[X']$ 는 다음과 같이 계산할 수 있다.

$$\begin{aligned}\mathbf{E}[X'] &= \frac{0 \cdot \frac{1}{2} \cdot 1 + 2 \cdot \frac{1}{4} \cdot 2 + 2 \cdot \frac{1}{8} \cdot 3 + 3 \cdot \frac{1}{16} \cdot 4 + 2 \cdot \frac{1}{32} \cdot 5}{\frac{30}{32}} \\ &= \frac{0 \cdot 16 \cdot 1 + 2 \cdot 8 \cdot 2 + 2 \cdot 4 \cdot 3 + 3 \cdot 2 \cdot 4 + 2 \cdot 1 \cdot 5}{30} \\ &= \frac{90}{30} = 3\end{aligned}$$

이제 T' 를 그려보자. T' 를 그리기 위해서는 다음 과정을 따르면 된다.

1. root 노드를 1개 추가한다. (이때 root 노드가 곧 leaf 노드가 된다.)
2. 현재 트리에서 대응된 값이 없는 leaf 노드들에 각각 2개의 자식 노드를 추가하고, 각 자식에게 가는 간선에 숫자 1, 2를 대응시킨다.
3. 현재 트리의 높이를 깊이로 하는 leaf 노드가 각 대응값 종류 별로 몇 개씩 존재하는지 확인 후 대응시킨다.
4. 더 이상 대응된 값이 없는 leaf 노드가 존재하지 않을 때까지 2.와 3.을 반복한다.

이 과정을 거쳐 실제로 시행 규칙을 그려보면 다음과 같다.

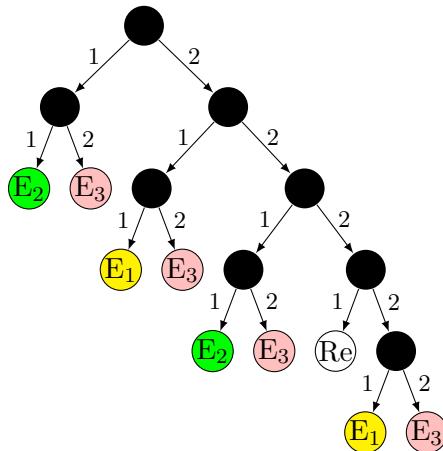


Figure 10: Table 6로부터 그린 시행 규칙

위에서의 과정은 이진 트리에 대해 수행한 것이지만, 일반적으로 k 진 트리의 경우에도 추가하는 자식 노드의 숫자를 k 로 하여 동일한 과정을 수행할 수 있으며 이를 통해 마찬가지로 시행 규칙을 그릴 수 있다.

4 Implementation

4.1 Development Environment

Analysis에서 특정 확률분포에 대해 해당 확률분포를 구할 수 있는 시행 규칙들 중 시행 횟수 기댓값이 가장 작은 시행 규칙을 찾고 출력하는 프로그램을 작성하기 위한 모든 이론적 분석을 완료하였다. 이제 해당 프로그램을 실제로 구현하기 위해 Microsoft Windows 10에서 Visual Studio 2019을 사용하여 C++로 코드를 작성하였다. 라이브러리는 C++ 표준 라이브러리만을 사용하였다. 완성된 프로그램은 Release 모드에서 빌드하여 Windows Terminal을 통해 실행하고 터미널 출력 및 파일 출력을 통해 결과를 확인하였다.

4.2 Calculator

우선 시행 규칙의 출력은 생략하고 시행 횟수 기댓값만 모든 경우에 대해 출력한 뒤 해당 값이 최소인 순간의 정보를 확인할 수 있는 프로그램인 Calculator를 제작하였다. 최종적으로 구현할 프로그램에서는 시행 횟수 기댓값을 모든 경우에 대해 출력할 필요가 없지만, 계산이 정상적으로 되는지 확인하고 또 특정 추측을 검증하기 위해 Calculator에서는 시행 횟수 기댓값을 모든 경우에 대해 출력하도록 하였다. 추측에 관해서는 5.1에서 자세히 다룬다.

사용할 시행 도구의 결괏값을 나타내는 이산확률변수가 $\text{unif}\{1, k\}$ 를 따른다고 할 때, 해당 k 값을 Calculator의 입력값으로 두었다. 또한 확률분포를 구성하는 확률들의 자연수비도 vector 형태의 입력값으로 두었으며, 해당 vector를 이루는 숫자들이 서로소가 아닌 경우 최대공약수로 나누어 서로소가 되도록 변환한 뒤 입력값으로 받아들이도록 하였다.

Calculator는 3.5에서 다룬 것과 같이 자연수비를 구성하는 숫자들에 곱할 c 값을 k 의 배수인 경우는 제외하고 차례대로 증가시켜가며 최종적으로 구해지는 정 k 진 트리 시행 규칙의 시행 횟수 기댓값을 계산한다. 시행 횟수 기댓값의 계산은 3.6에서 다루었던 내용을 기반으로 c 가 곱해진 확률들의 자연수비를 구성하는 숫자들을 각각 k 진수로 바꾸는 과정에서 바로 실시하며, 시행 규칙을 출력할 필요가 없기 때문에 해당 k 진수들을 따로 저장하지는 않는다.

c 값에 따라 최종적으로 구해지는 시행 규칙의 높이가 정해지므로 시행 규칙의 높이가 될 수 있는 각 값마다 해당 높이의 시행 규칙이 구해지도록 하는 c 값들 중 구해진 시행 규칙의 시행 횟수 기댓값이 최소인 경우의 c 값이 존재한다. 해당 c 값과 시행 규칙의 높이, 시행 횟수 기댓값을 기록 후 출력하여 확인할 수 있도록 하였다. 그리고 프로그램이 무한히 동작하지 않도록 extra depth limit을 입력받아 시행 규칙의 높이가 최솟값인 경우부터 최솟값보다 extra depth limit 만큼 큰 경우까지만 c 값을 증가시키며 찾도록 하였다.

실제 Calculator의 출력이 어떤 식으로 이루어지는지 살펴보자. 다음은 Table 5에서 다루었던 경우를 Calculator에 입력되도록 하여 나온 터미널 출력 결과이다. 입력되는 k 값은 2, vector는 { 1, 2, 3 }이며 extra depth limit은 5이다.

```
> .\Calculator.exe
k : 2 / vector : 1 2 3 / extra depth limit : 5
##### height = 3 #####
c = 1 ) E[X] = 3
##### height = 5 #####
c = 3 ) E[X] = 5
c = 5 ) E[X] = 3
##### height = 6 #####
c = 7 ) E[X] = 4.61905
c = 9 ) E[X] = 3.2963
##### height = 7 #####
c = 11 ) E[X] = 5.72727
c = 13 ) E[X] = 4.84615
c = 15 ) E[X] = 4.46667
c = 17 ) E[X] = 3.47059
c = 19 ) E[X] = 3.31579
c = 21 ) E[X] = 3
##### height = 8 #####
c = 23 ) E[X] = 5.57971
c = 25 ) E[X] = 5.02667
c = 27 ) E[X] = 4.90123
c = 29 ) E[X] = 4.56322
c = 31 ) E[X] = 4.39785
c = 33 ) E[X] = 3.56566
c = 35 ) E[X] = 3.47619
c = 37 ) E[X] = 3.28829
c = 39 ) E[X] = 3.2906
c = 41 ) E[X] = 3.06504

min E[X] = 3
min E[X] when c = 1
height = 3      | ~ c = 1
height = 5      | ~ c = 5
height = 6      | ~ c = 9
height = 7      | ~ c = 21
height = 8      | ~ c = 41
>
```

3.6에서의 계산 결과와 동일하게, $c = 5$, $height = 5$ 일 때 $E[X] = 3$ 이 출력되어 나오는 것을 확인할 수 있다.

다음은 Calculator에 입력되는 k 값을 3, vector를 { 5, 2 }, extra depth limit을 3으로 설정하여 나온 터미널 출력 결과이다.

```
> .\Calculator.exe
k : 3 / vector : 5 2 / extra depth limit : 3
##### height = 2 #####
c = 1 ) E[X] = 2.14286
##### height = 3 #####
c = 2 ) E[X] = 2.78571
##### height = 4 #####
c = 4 ) E[X] = 5.67857
c = 5 ) E[X] = 4.2
c = 7 ) E[X] = 2.69388
c = 8 ) E[X] = 2.78571
c = 10 ) E[X] = 2.14286
c = 11 ) E[X] = 1.55844
##### height = 5 #####
c = 13 ) E[X] = 5.30769
c = 14 ) E[X] = 4.53061
c = 16 ) E[X] = 4.3125
c = 17 ) E[X] = 3.12605
c = 19 ) E[X] = 2.75188
c = 20 ) E[X] = 2.78571
c = 22 ) E[X] = 2.43506
c = 23 ) E[X] = 2.25466
c = 25 ) E[X] = 2.76
c = 26 ) E[X] = 2.63736
c = 28 ) E[X] = 2.28061
c = 29 ) E[X] = 2.18719
c = 31 ) E[X] = 2.22581
c = 32 ) E[X] = 2.10268
c = 34 ) E[X] = 1.57563

min E[X] = 1.55844
min E[X] when c = 11
height = 2      | ~ c = 1
height = 3      | ~ c = 2
height = 4      | ~ c = 11
height = 5      | ~ c = 34
>
```

$c = 8$, $height = 4$ 일 때 Figure 5의 시행 규칙과 동일한 시행 규칙이 구해지며, 시행 횟수 기댓값의 계산 결과도 3.3에서 계산한 결과와 동일하다.

4.3 Printer

시행 규칙을 파일로 출력하는 실제 사용 가능한 프로그램 Printer를 완성하기 위해, Calculator를 수정하였다. 우선 사용자가 프로그램 실행 후 k 값, vector 구성 숫자, extra depth limit을 직접 입력할 수 있도록 하였고, 시행 횟수 기댓값을 모든 경우에 대해 출력하지 않도록 변경하였다. 각 height 값에서 시행 규칙의 시행 횟수 기댓값이 최소인 경우의 해당 값은 계속 출력하여 사용자가 확인할 수 있도록 하였고, 시행 규칙은 사전식 나열 형태로 출력하여 txt 파일 형태로 저장하도록 하였다.

txt 파일 내에서 출력되는 시행 규칙은 각 height 값에서 시행 횟수 기댓값이 최소인 시행 규칙이며, height가 더 작으면서 시행 횟수 기댓값이 작거나 같은 시행 규칙이 존재할 경우에는 출력을 생략하도록 하였다. 이는 사용자가 Printer를 사용하여 구한 시행 규칙을 실제 보드게임에서 사용가능한 형태로 인쇄할 경우, 높이가 큰 시행 규칙일 수록 복잡한 형태가 되므로 가능한 한 높이가 작은 시행 규칙을 사용하려고 할 것이라는 전제를 반영한 것이다.

시행 규칙 출력을 위해 시행 규칙을 나타내는 클래스를 정의하였다. 해당 클래스는 k 진 트리의 노드를 나타내는 구조체를 사용하며, c 가 곱해진 확률들의 자연수비를 구성하는 숫자들을 k 진수로 바꾼 결과를 2차원 vector로 저장하고 이를 이용하여 시행 규칙 내용을 완성하는 생성자를 가진다. 이 과정은 3.6에서 설명한 과정과 동일하다. 클래스 내에 시행 규칙을 사전식 나열 형태로 출력하기 위한 함수도 정의하였으며, 해당 함수는 전위 순회(preorder traversal)를 root 노드로부터의 경로를 기억하고 노드 값과 함께 출력하는 방식으로 구현하였다.

다음은 Printer 실행 후 k 값을 3, 자연수비 vector를 { 5, 2 }, extra depth limit을 6으로 입력하여 나온 터미널 출력 결과이다.

```
> .\Printer.exe
시행 도구에서 k가지 경우가 동일한 확률로 나오는 경우, 해당 k 값을 입력해주세요.
(Ex. 동전 2 / 주사위 6)
k 값 입력 : 3
각각의 사건이 일어날 확률들의 자연수비를 구성하는 숫자를 차례대로 입력해주세요.
(Ex. 당첨 30% 꽝 70% -> 3 7 / A 5%, B 20%, C 75% -> 1 4 15)
자연수비 구성 숫자 입력 : 5 2
extra depth limit을 입력해주세요.
(시행 규칙의 높이가 최솟값보다 extra depth limit 만큼 큰 경우까지 탐색합니다.)
extra depth limit 값 입력 : 6
# height = 2    // min E[X] = 2.14285714286 // c = 1
출력합니다.
# height = 3    // min E[X] = 2.78571428571 // c = 2
생략합니다.
# height = 4    // min E[X] = 1.55844155844 // c = 11
출력합니다.
```

```

# height = 5      // min E[X] = 1.57563025210 // c = 34
생략합니다.
# height = 6      // min E[X] = 1.50000000000 // c = 104
출력합니다.
# height = 7      // min E[X] = 1.50620119430 // c = 311
생략합니다.
# height = 8      // min E[X] = 1.50068608020 // c = 937
생략합니다.
>

```

이후 Printer에 의해 출력 파일이 생성되면 파일의 내용은 다음과 같다.

```

## input value ##
## k = 3 // vector = { 5 2 } // extra depth limit = 6 ##

## probabilities ##
E_1    : 71.429%
E_2    : 28.571%

=====
# height = 2 // min E[X] = 2.14285714286 #
-----
1 - E_1
2
2 1 - Re
2 2 - Re
2 3 - E_1
3
3 1 - E_1
3 2 - E_2
3 3 - E_2
=====

=====
# height = 3 // min E[X] = 2.78571428571 #
=====

=====
# height = 4 // min E[X] = 1.55844155844 #
-----
1 - E_1
2 - E_1
3
3 1 - E_2
3 2 - E_2
3 3
3 3 1 - Re
3 3 2 - E_2
3 3 3

```

```

3 3 3 1 - Re
3 3 3 2 - E_1
3 3 3 3 - E_2
=====
=====

# height = 5 // min E[X] = 1.57563025210 #
=====

=====

# height = 6 // min E[X] = 1.500000000000 #
-----

1 - E_1
2 - E_1
3
3 1 - E_2
3 2 - E_2
3 3
3 3 1 - E_1
3 3 2 - E_2
3 3 3
3 3 3 1 - E_2
3 3 3 2 - E_2
3 3 3 3
3 3 3 3 1 - E_1
3 3 3 3 2 - E_1
3 3 3 3 3
3 3 3 3 3 1 - Re
3 3 3 3 3 2 - E_1
3 3 3 3 3 3 - E_2
=====

=====

# height = 7 // min E[X] = 1.50620119430 #
=====

=====

# height = 8 // min E[X] = 1.50068608020 #
=====
```

이 경우 사용자는 height(시행 규칙의 높이)가 작은 대신 $\min E[X]$ (최소인 시행 횟수 기댓값)이 큰 경우를 선택할지, 아니면 height가 큰 대신 $E[X]$ 가 작은 경우를 선택할지 결정해야 한다. 이것은 일종의 공간적 비용과 시간적 비용 간 trade-off 관계가 형성되었다고 볼 수 있으며, 어떤 결정을 내릴 지는 구체적인 수치와 상황을 고려하여 판단하면 된다.

5 Discussion

5.1 Minimum $E[X]$ with Non-Maximum c

4.2에서 프로그램을 제작하는 과정에서, 적당한 extra depth limit을 입력했다면 각 height마다 c 값이 가장 큰 경우만 조사하면 전체 모든 경우 중에 시행 횟수 기댓값이 가장 작은 경우를 찾을 수 있다는 추측을 하였다. 3.3의 마지막 문단에서 밝혔듯이, 시행 규칙의 시행 횟수 기댓값을 계산하는 공식(Theorem 4)의 분모는 시행 규칙의 root 노드에서 시작하여 맨 처음 도달하는 leaf 노드가 Re가 대응되지 않은 노드일 확률이므로, 각 height마다 Re가 대응되지 않은 노드의 비중을 가능한 한 가장 크게 해서 분모가 가장 큰 경우만 조사하면 될 것이라 추측한 것이다.

그러나 여러 가지 입력값을 프로그램에 넣어가며 확인해본 결과 위 추측에 대한 반례를 발견하였다. 다음은 Printer에서 k 값을 6, vector를 { 1, 4, 15 }, extra depth limit를 6으로 입력하여 나온 터미널 출력 결과이다.

```
> .\Printer.exe
시행 도구에서 k가지 경우가 동일한 확률로 나오는 경우, 해당 k 값을 입력해주세요.
(Ex. 동전 2 / 주사위 6)
k 값 입력 : 6
각각의 사건이 일어날 확률들의 자연수비를 구성하는 숫자를 차례대로 입력해주세요.
(Ex. 당첨 30% 꽝 70% -> 3 7 / A 5%, B 20%, C 75% -> 1 4 15)
자연수비 구성 숫자 입력 : 1 4 15
extra depth limit을 입력해주세요.
(시행 규칙의 높이가 최솟값보다 extra depth limit 만큼 큰 경우까지 탐색합니다.)
extra depth limit 값 입력 : 6
# height = 2 // min E[X] = 2.4000000000 // c = 1
출력합니다.
# height = 3 // min E[X] = 1.3200000000 // c = 10
출력합니다.
# height = 4 // min E[X] = 1.2333333333 // c = 63
출력합니다.
# height = 5 // min E[X] = 1.2333333333 // c = 387
생략합니다.
# height = 6 // min E[X] = 1.2333333333 // c = 2331
생략합니다.
# height = 7 // min E[X] = 1.2333333333 // c = 13995
생략합니다.
# height = 8 // min E[X] = 1.2333333333 // c = 83979
생략합니다.
>
```

구해지는 시행 규칙이 최소의 높이, 최소의 시행 횟수 기댓값을 가지는 경우는 height가 4이고 $c = 63$ 일 때이다. 동일한 조건에 대해 extra depth limit만 2로 수정하여 Calculator의 입력값으로 설정하고 실행하여 나온 터미널 출력 결과는 다음과 같다.

```
> .\Calculator.exe
k : 6 / vector : 1 4 15 / extra depth limit : 2
##### height = 2 #####
c = 1 ) E[X] = 2.4
##### height = 3 #####
c = 2 ) E[X] = 7.35
c = 3 ) E[X] = 4.3
c = 4 ) E[X] = 3.75
c = 5 ) E[X] = 2.64
c = 7 ) E[X] = 2.14286
c = 8 ) E[X] = 1.8375
c = 9 ) E[X] = 1.43333
c = 10 ) E[X] = 1.32
##### height = 4 #####
c = 11 ) E[X] = 8.23636
c = 13 ) E[X] = 6.96923
c = 14 ) E[X] = 6.45
c = 15 ) E[X] = 5.3
c = 16 ) E[X] = 4.9875
c = 17 ) E[X] = 4.69412
c = 19 ) E[X] = 4.2
c = 20 ) E[X] = 3.975
c = 21 ) E[X] = 3.78571
c = 22 ) E[X] = 4.11818
c = 23 ) E[X] = 3.93913
c = 25 ) E[X] = 3.624
c = 26 ) E[X] = 3.54231
c = 27 ) E[X] = 3.27778
c = 28 ) E[X] = 3.23571
c = 29 ) E[X] = 2.75172
c = 31 ) E[X] = 2.63226
c = 32 ) E[X] = 2.48438
c = 33 ) E[X] = 2.79091
c = 34 ) E[X] = 2.66471
c = 35 ) E[X] = 2.64
c = 37 ) E[X] = 2.4
c = 38 ) E[X] = 2.37632
c = 39 ) E[X] = 2.26923
c = 40 ) E[X] = 2.265
c = 41 ) E[X] = 2.16585
c = 43 ) E[X] = 2.10698
c = 44 ) E[X] = 2.05227
c = 45 ) E[X] = 1.96667
c = 46 ) E[X] = 1.93043
```

```

c = 47 ) E[X] = 1.92766
c = 49 ) E[X] = 1.84898
c = 50 ) E[X] = 1.806
c = 51 ) E[X] = 1.77059
c = 52 ) E[X] = 1.74231
c = 53 ) E[X] = 1.70943
c = 55 ) E[X] = 1.64727
c = 56 ) E[X] = 1.6125
c = 57 ) E[X] = 1.58421
c = 58 ) E[X] = 1.37586
c = 59 ) E[X] = 1.35254
c = 61 ) E[X] = 1.3082
c = 62 ) E[X] = 1.31129
c = 63 ) E[X] = 1.23333
c = 64 ) E[X] = 1.24687

```

```

min E[X] = 1.23333
min E[X] when c = 63
height = 2      | ~ c = 1
height = 3      | ~ c = 10
height = 4      | ~ c = 64
>

```

height가 4일 때 c 값이 가장 큰 경우는 $c = 64$ 인데, 시행 횟수 기댓값이 가장 작은 경우는 $c = 63$ 이다. 이는 각 height마다 c 값이 가장 큰 경우만 조사한다면 시행 횟수 기댓값이 최소인 경우를 찾지 못할 수도 있다는 예시가 된다. 따라서 최종 완성된 프로그램에서는 각 height마다 c 값이 가장 큰 경우만 조사하는 것이 아니라 전부 조사하도록 구현하였다.

5.2 Converging Minimum $E[X]$

만약 모든 입력값에 대해 반드시 $\min E[X]$ 가 최솟값인 순간이 존재하고 이를 적당한 height 범위 내에서 찾는게 가능하다면, 프로그램이 extra depth limit을 입력받지 않고 그냥 최솟값 인 순간을 찾을 때까지 동작하도록 만들어도 될 것이다. 그러나 프로그램에 여러 가지 값을 입력해보는 과정에서 특정 입력값에 대해서는 $\min E[X]$ 이 최소인 순간이 존재하지 않고, 대신 height가 커짐에 따라 $\min E[X]$ 값이 특정 값에 수렴하면서 계속 감소하는 것을 발견하였다.

다음은 Printer에서 k 값을 2, vector를 { 3, 4, 5 }, extra depth limit를 28로 입력하여 나온 터미널 출력 결과이다.

```

> .\Printer.exe
시행 도구에서 k가지 경우가 동일한 확률로 나오는 경우, 해당 k 값을 입력해주세요.
(Ex. 동전 2 / 주사위 6)
k 값 입력 : 2
각각의 사건이 일어날 확률들의 자연수비를 구성하는 숫자를 차례대로 입력해주세요.

```

(Ex. 당첨 30% 꽝 70% -> 3 7 / A 5%, B 20%, C 75% -> 1 4 15)
 자연수비 구성 숫자 입력 : 3 4 5
 extra depth limit을 입력해주세요.
 (시행 규칙의 높이가 최솟값보다 extra depth limit 만큼 큰 경우까지 탐색합니다.)
 extra depth limit 값 입력 : 28

```
# height = 4    // min E[X] = 3.16666666667 // c = 1
출력합니다.
# height = 5    // no case
생략합니다. 해당 height인 case가 없습니다.
# height = 6    // min E[X] = 3.03333333333 // c = 5
출력합니다.
# height = 7    // min E[X] = 3.46296296296 // c = 9
생략합니다.
# height = 8    // min E[X] = 3.00793650794 // c = 21
출력합니다.
# height = 9    // min E[X] = 3.10162601626 // c = 41
생략합니다.
# height = 10   // min E[X] = 3.00196078431 // c = 85
출력합니다.
# height = 11   // min E[X] = 3.02465483235 // c = 169
생략합니다.
# height = 12   // min E[X] = 3.00048875855 // c = 341
출력합니다.
# height = 13   // min E[X] = 3.00611845326 // c = 681
생략합니다.
# height = 14   // min E[X] = 3.00012210012 // c = 1365
출력합니다.
# height = 15   // min E[X] = 3.00152681080 // c = 2729
생략합니다.
# height = 16   // min E[X] = 3.00003051944 // c = 5461
출력합니다.
# height = 17   // min E[X] = 3.00038152794 // c = 10921
생략합니다.
# height = 18   // min E[X] = 3.00000762951 // c = 21845
출력합니다.
# height = 19   // min E[X] = 3.00009537107 // c = 43689
생략합니다.
# height = 20   // min E[X] = 3.00000190736 // c = 87381
출력합니다.
# height = 21   // min E[X] = 3.00002384209 // c = 174761
생략합니다.
# height = 22   // min E[X] = 3.00000047684 // c = 349525
출력합니다.
# height = 23   // min E[X] = 3.00000596048 // c = 699049
생략합니다.
# height = 24   // min E[X] = 3.00000011921 // c = 1398101
출력합니다.
# height = 25   // min E[X] = 3.00000149012 // c = 2796201
생략합니다.
```

```

# height = 26 // min E[X] = 3.00000002980 // c = 5592405
출력합니다.
# height = 27 // min E[X] = 3.00000037253 // c = 11184809
생략합니다.
# height = 28 // min E[X] = 3.00000000745 // c = 22369621
출력합니다.
# height = 29 // min E[X] = 3.00000009313 // c = 44739241
생략합니다.
# height = 30 // min E[X] = 3.00000000186 // c = 89478485
출력합니다.
# height = 31 // min E[X] = 3.00000002328 // c = 178956969
생략합니다.
# height = 32 // min E[X] = 3.00000000047 // c = 357913941
출력합니다.
>

```

height가 증가함에 따라 $\min E[X]$ 이 계속 3에 가까워지며 더 작아지기 때문에, extra depth limit 없이 $\min E[X]$ 이 최소인 순간을 찾을 때까지 프로그램을 동작하도록 만든다면 프로그램이 무한히 동작하게 된다. 따라서 최종 완성된 프로그램에서 extra depth limit는 단지 실행 시간을 절약하기 위한 것 뿐만이 아니라 프로그램의 완성도를 위해 필수적으로 설정되어야 하는 입력값임을 알 수 있다.

6 Conclusion and Further Research

이 논문을 통해 보드게임 제작자가 보드게임을 만드는 과정에서 확률적 요소의 구현에 어려움을 겪던 문제를 수학과 컴퓨터 공학 관련 개념 및 이론을 바탕으로 해결하였고, 이를 기반으로 실제로 사용 가능한 프로그램도 제작하였다. 그리고 해당 프로그램을 통해 이론적인 분석 과정에서 스스로 제기한 추측들도 해결하였다.

5.2에서 다루었듯이 확률분포 구현을 위한 시행 규칙의 시행 횟수 기댓값이 최소인 경우를 찾는 것이 불가능한 경우가 존재하는데, 언제 이런 식으로 불가능한 경우가 되고 언제 찾는 것이 가능한지, 그 이유는 무엇인지 이론적으로 분석해서 경우를 나누어 일반화하는 것이 후속 연구로 가능할 것으로 생각된다. 다른 후속 연구로는 사전식 나열로 출력된 시행 규칙을 트리 형태의 그림으로 바꾸는 프로그램을 개발하는 것도 고려해볼 수 있으며, 시행 도구의 시행 결과값이 이산균등분포를 따르지 않는 경우를 연구하는 것도 고려해볼 수 있다.

논문에서 다룬 내용은 양자 난수 생성기(Quantum Random Number Generator, QRNG)를 내장한 전자기기가 해당 난수 생성기 내부의 큐비트(Qubit)가 따르는 이산균등분포를 원하는 특정 확률분포로 변환하는 알고리즘을 만드는 문제로 바꾸어 생각할 수도 있다. 큐비트에서

값을 측정하는 것을 시행 도구를 사용하는 것으로 치환하여 생각한다면, 이 논문에서 다룬 내용을 기반으로 효율적인 알고리즘을 만들 수 있을 것이다.

References

- [1] 한국콘텐츠진흥원, 『2021년 게임콘텐츠 제작지원 사업안내서』. 2021.
- [2] Roger Caillois, 『놀이와 인간』. 이상률 옮김, 문예출판사, 2018.