출품번호
융합과학교육원에서 기재함

제66회 서울과학전람회 예선대회 작품설명서

자연에서 발견되는 황금각의 원리를 이용한 공간 최적화 문제의 해결

2024. 9. 10.

구분	학생부
출품부문	산업 및 에너지

【 차례 】

I 탐구 동기 및 목적

1. 탐구 동기

과학책을 읽다가 자연에서 관찰되는 식물의 잎들 사이에 일정한 각도가 있다는 사실을 알게 되었다. 특히, 나뭇잎들이 약 137.5도씩 회전하며 자라나는데, 이는 위에서 볼 때 잎이 겹치지 않고 더 많은 햇빛을 받을 수 있도록 돕기 위한 것이라고 설명되어 있었다. 이 현상에 호기심이 생겨 자료를 더 찾아보니, 이 각도가 '황금각'으로 불리며 360도를 황금비에 따라 나눈 결과라는 것을 알게 되었다.

그러나 왜 하필 137.5도인지, 다른 각도와는 어떤 차이가 있는지에 대한 궁금증이 커졌다. 이 궁금증을 해결하기 위해 황금각의 원리를 파이썬 프로그래밍으로 확인해 보고자 했으며, 나아가 이 원리를 직선에도 적용할 수 있는 방법을 찾아낸다면 실생활에서 유용하게 활용할 수 있는 가능성도 있다고 생각했다. 이러한 이유로 이번 탐구를 시작하게 되었다.

2. 탐구 목적

본 탐구의 목적은 식물의 잎들에서 관찰되는 황금각의 원리를 분석하고, 각이 아닌 직선에 적용하여 그 효과를 분석하고, 이를 실생활에 응용할 수 있는 방법을 탐구 해 본다.

가. 황금각의 효율성 분석: 식물들이 잎을 배치할 때 사용하는 황금각이 다른 각도에 비해 더 고르게 분포하는지 검토해 본다.

나. 직선에 적용한 황금비: 각이 아닌 직선에 황금비를 반복적으로 적용했을 때 만들어지는 간격이 얼마나 고르게 분포하는지 조사하고, 다른 비율을 적용했을 때의 분포와 비교해 본다.

다. 실생활 적용 사례 탐구: 황금비의 원리를 활용하여 피보나치 해싱(Fibonacci hashing)을 만들고, 이를 실생활에 어떻게 적용할수 있을지 사례를 탐구해 본다.

II 탐구 개요

가. 탐구 이론적 배경

식물들은 잎을 배치할 때 서로 겹치지 않도록 황금각이라는 특별한 각도로 잎을 배열한다. 이 방법은 잎들이 최대한 많은 햇빛을 받을 수 있게 하여 식물의 광합성과 생존을 효율적으로 돕는다. 이러한 현상은 황금비, 황금각, 그리고 피보나치 수열을 통해 설명될 수 있다.

가. 황금비율: 황금비율(약 1.618:1)은 미적 아름다움의 기준으로 여겨지며, 자연, 예술, 건축 등 다양한 분야에서 발견된다. 이 비율은 한 선분을 두 부분으로 나누었을 때, 더 긴 부분과 전체 선분의 비율이 더 짧은 부분과 더 긴 부분의 비율과 같은 경우를 말한다. 이 수치는 약 1.6180339887로 계산된다.

나. 황금각: 황금각은 원을 황금비율에 따라 나눈 각도로, 약 137.5도에 해당한다. 식물들이 잎을 나선형으로 배열할 때 이 각도를 사용하며, 각 잎은 이전 잎에서 황금각만큼 회전하여 배치된다. 이 방법은 잎들이 겹치지 않고 최대한 효율적으로 햇빛을 흡수할 수 있도록 돕는다.

다. 피보나치 수열 :피보나치 수열은 이탈리아의 수학자 레오나르도 피보나치에 의해 소개되었으며, 다음과 같은 규칙을 따른다

첫 번째 항은 0이다.
두 번째 항은 1이다.
세 번째 항부터는 앞의 두 항의 합이 된다.

이 수열은 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, ...와 같이 전개되며, 각 숫자는 앞의 두 숫자의 합으로 이루어진다. 피보나치 수열의 항 간의 비율은 점점 황금비에 가까워지며, 무한히 진행될 때 연속하는 항 간의 비율은 정확히 황금비에 도달한다. 이러한 특성 때문에 피보나치 수열은 자연스럽게 황금비를 따르는 구조로 나타나며, 식물의 잎 배열이 황금비를 따르는 것도 이 수열과 직접적으로 연관되어 있다.

나. 탐구 기간

표 1: 탐구 기간

기간 / 연구내용		추진 시기 (월)				
		2024년				
		5	6	7	8	9
계획	탐구 주제 선정	●				
	이론적 배경 파악	●				
실행	선행 연구 조사		●			
	탐구 설계		●			
정리	탐구 수행		●	●		
	결과 정리 및 보고서 작성				●	●

III 선행 연구 조사

서울시 특별시교육청 융합과학교육원 홈페이지에서 과학경진 대회 입상작 - 과학전람회, 학생발명품경진대회, (구) 서울 학생 탐구발표대회 검색 결과, 유사 주제 탐구는 검색되지 않았다. 국립중앙과학관 홈페이지에서 전람회 전국 과학전람회 출품작 통합 검색 결과, 생물 분야에서 피보나치수열과 식물의 성장과 식생의 개선에 관한 탐구와, 산업 에너지 분야에서 피보나치수열을 이용한 조명 배치 등에 관한 연구들이 있었지만, 피보나치 해싱과 직접적인 관련이 있는 연구는 검색되지 않았다. 따라서 본 연구는 탐구해 볼 만한 가치가 있다고 판단하였다.

표 2 선행연구 조사 결과

1. 서울특별시교육청 융합과학교육원

대회	**유사 탐구**	**유사성**
과학전람회예선대회	없음	해당 없음
과학전람회본선대회	없음	
학생과학발명품경진대회	없음	
청소년과학페어(과학토론)	없음	

2. 국립중앙과학관- 전국과학전람회

연도	**분야**	**제목**	**내용**	**유사성**
2021	생물	생물 속의 피보나치수열 관찰 및 응용	동식물에서 나타나는 피보나치 패턴의 생물의 진화적 관점 탐구	해당 없음
2018	산업 및 에너지	피보나치수열을 이용한 실내조명 배치의 최적화에 관한 탐구	해바라기 씨앗에서 보이는 피보나치수열의 배치 원리를 이용한 실내조명 배치의 최적화	해당 없음
2017	환경	피보나치수열을 통한 화명수목원 식생 환경의 영향 분석 및 개선 연구	피보나치수열의 원리를 이용하여 수목원 내의 식물들의 자연적인 생장을 할 수 있는 환경 조성 탐구	해당 없음

IV 탐구 절차 및 방법

1. 조사 및 분석

자연에서 관찰되는 황금각의 원리 분석

2. 가설 설정

가설 1 - 황금각을 반복적으로 적용할 경우, 데이터는 다른 각도보다 더 고르게 분포할 것이다.
가설 2 - 이 원리를 선분에 적용해도 같은 결과를 얻을 수 있을 것이다.

3. 탐구 설계

각도에 따른 배열의 균일성 실험 - 파이썬 프로그래밍
다른 각들을 적용하는 비교 실험

4. 탐구 수행 및 분석

황금각과 다른 각도를 적용한 분포 비교 분석
황금비와 다른 비율을 적용한 선분 간격 분포 비교 분석
알고리즘 설계하여 다양한 데이터에 적용

5 탐구 결과

실험 결과를 바탕으로 가설 검증 및 평가
설계한 알고리즘 성능 평가
실생활 응용 방법 탐색

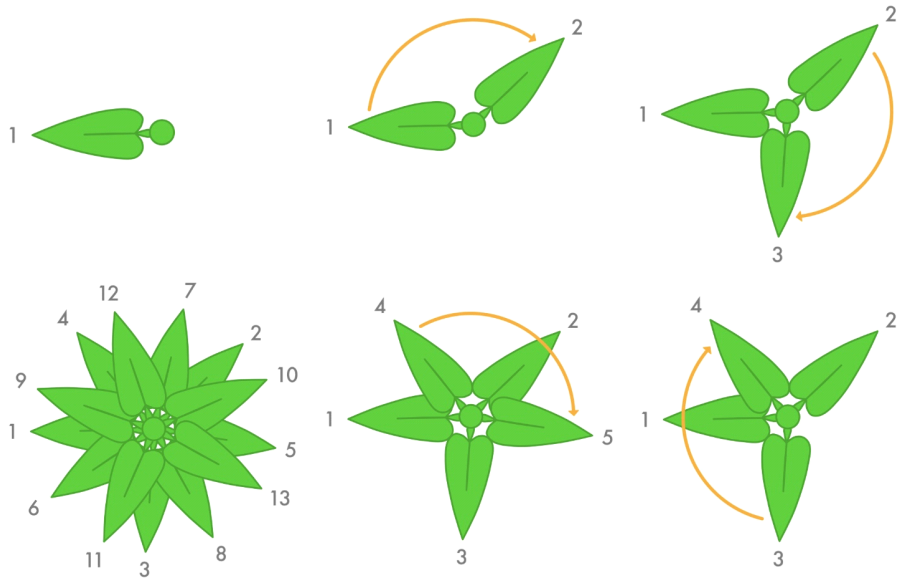
6. 결론 도출 및 활용성

연구 결과 종합
알고리즘의 효율성 확인
실생활 응용 사례 제안

V 탐구 내용

1. 자연에서 발견되는 황금각

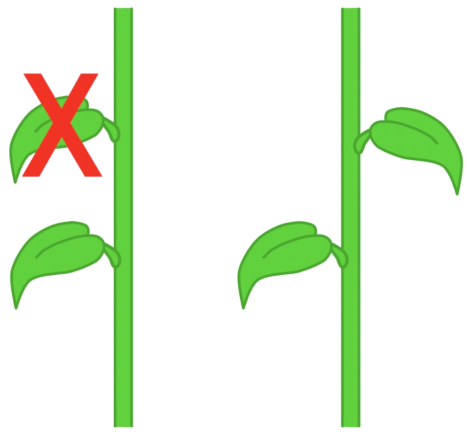
식물들은 자라면서 줄기나 잎을 하나씩 펼치게 되는데 위에서 보게 되면 일정한 각도를 이루는 것처럼 보인다. 많은 경우 이 각은 **137.5도**에 가까운 값을 가지는데 이 값을 **황금각**이라고 부른다.



(이미지 출처: https://gofiguremath.org/natures-favorite-math/the-golden-ratio/the-golden-angle/)

그림 1: 식물들이 황금각(137.5도)에 가까운 각을 이루면서 잎을 펼치는 원리

그림 1는 이 과정을 그림으로 나타낸 것이다.



(이미지 출처: https://gofiguremath.org/natures-favorite-math/the-golden-ratio/the-golden-angle/)

그림 2: 식물들의 잎이 이루는 각도에 따른 관계

그림 2는 위에서 보았을 때 잎들이 이루는 각도에 따라서 겹치는 경우와 겹치지 않는 경우를

그림으로 나타낸 것이다.

신기하게도 잎들이 서로 황금각(137.5도)에 가까운 각을 이루면서 나게 되면 위에서 보았을 때 서로 겹치는 면적이 최소가 된다고 한다. 잎들이 서로 겹치지 않게 되면 다음과 같은 여러 가지 장점을 가지게 된다.

- 햇빛을 받는 면적이 최대가 된다.
- 식물이 전체적으로 균형을 이루게 되어 잘 쓰러지지 않게 된다.
- 공간을 효율적으로 사용할 수 있게 되어 여러 가지(공기 순환, 자원 경쟁 등) 면에서 유리하다.

2. 각의 크기에 따라 겹치는 정도가 어느 정도 달라질까?

잎들이 이루는 각의 크기에 따라 서로 겹치는 정도를 어느 정도 차이가 나는지 궁금해졌다. 그래서 각을 입력하면 입력된 각도만큼 회전시키면서 분할각을 만들어서 그려주는 Python 프로그램을 작성해 보았다.

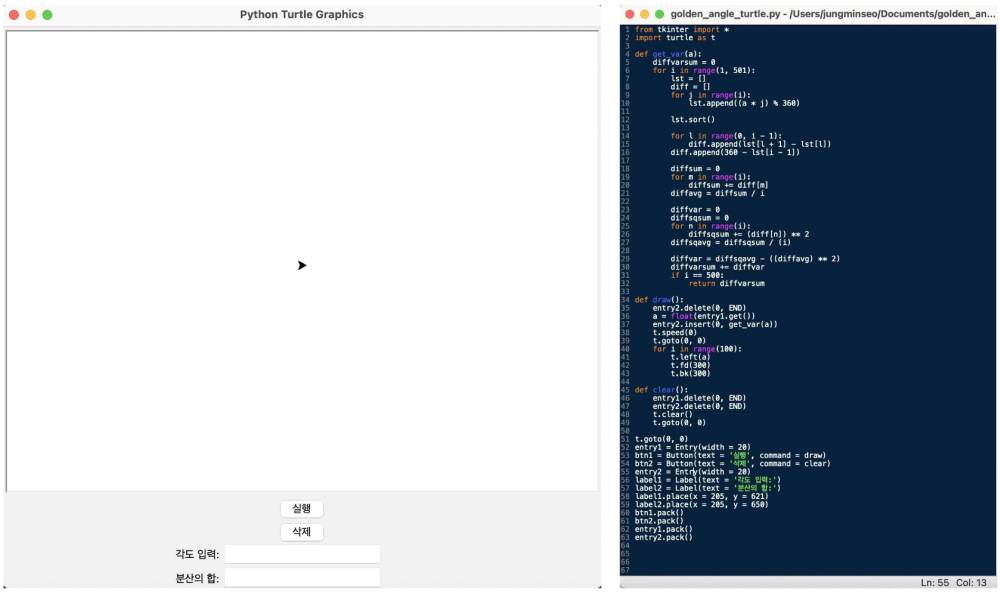


그림 3: 입력된 각만큼 반복해서 회전시키면서 분할각들을 만들어주는 Python 프로그램

그림 3는 Turtle Graphics 라이브러리를 이용하여 직접 작성한 Python 프로그램의 화면과 소스코드를 캡쳐한 것이다. 여러 가지 각도를 직접 입력해서 입력값에 따라서 만들어지는 각을 나누는 선들이 서로 겹쳐지는 정도가 차이가 나는지 실험해 보았다.

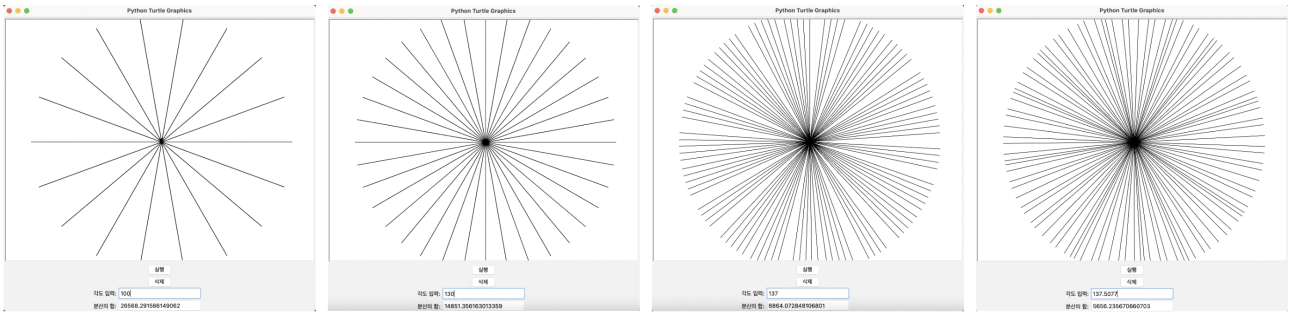


그림 4: 입력값을 다르게 했을 때 만들어지는 분할각들

그림 4는 여러 가지 각도를 입력했을 때 만들어지는 분할각들을 보여준다. 각각의 결과는 입력된 각을 회전시키면서 최대 500개까지 분할각을 만들어서 그림을 그린 결과이다. 그림의 실험에서 사용된 각들은 각각 100도, 130도, 137도, 137.5077도이다. 그림에서 보듯이 입력값이 황금각(실제로는 137.50776405...의 무리수 값)에 가까울수록 각을 나누는 선들이 겹치는 경우가 줄어드는 것을 관찰할 수 있었다.

각을 나누는 선들이 서로 겹치는 경우가 적다고 하는 것은 다른 말로 하면 분할각들이 360도 안에서 비슷한 간격을 두고 고르게 분포되어 있다는 뜻이다. 따라서 식물이 황금각을 이용해서 풀고 있는 문제를 다음과 같이 정의할 수 있다.

- **[각 나누기 문제]**: 360도를 가능한 한 비슷한 크기의 1개에서부터 N개의 분할각들으로 계속 나누어 가는 문제

분할각의 크기가 최대한 비슷하도록 나눈다는 것은 그 분산 값이 최소가 된다는 것과 같은 의미가 된다. 분산 값을 계산해서 좀 더 다양한 입력값들에 대해서 실험해 보기로 했다. 이것을 수행하는 Python 프로그램을 작성해서 실험을 진행하였다.

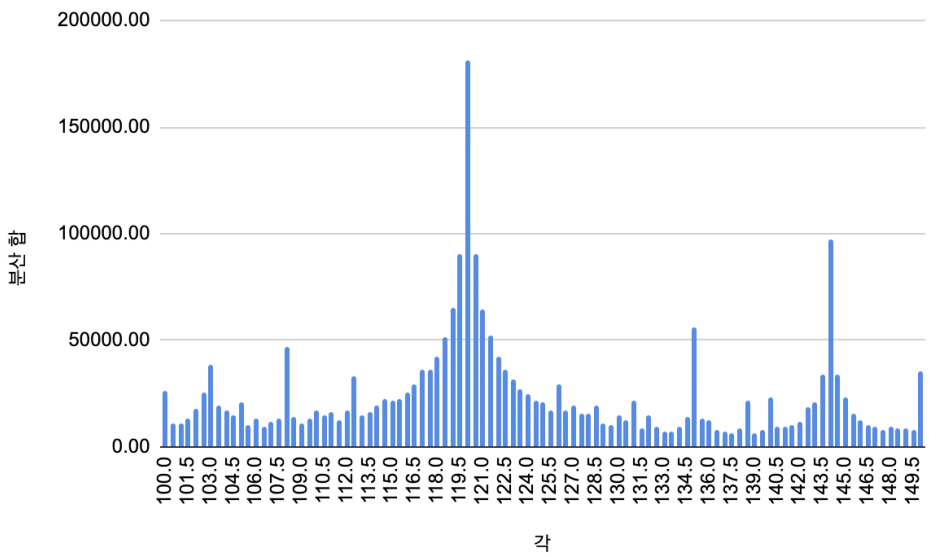


그림 5: 100도에서 150도 사이의 입력값에 대해서 만들어지는 각들의 분산 합

그림 5는 100도에서 150도 사이의 입력값들에 대해서 최대 500개의 분할각들이 만들어지는 과정에서 생기는 사이각들의 분산값들을 모두 더해서 얻어진 값(**분산 합**)들을 그래프로 표현한 것이다. 그림에서 보듯이 입력값이 황금각(137.5도)일 때 가장 작은 분산 합을 가지는 것을 확인할 수 있었다.

이렇게 입력값에 대해서 회전시키면서 분할각을 만들어내는 방식(**<황금각 반복>**)이 아닌 다른 방식들과 비교해 보기로 했다. 가장 쉽게 생각할 수 있는 방식이 랜덤 함수를 이용한 방식(**<랜덤 각>**)이다. 이것을 수행하는 Python 프로그램을 작성해서 실험을 진행하였다.

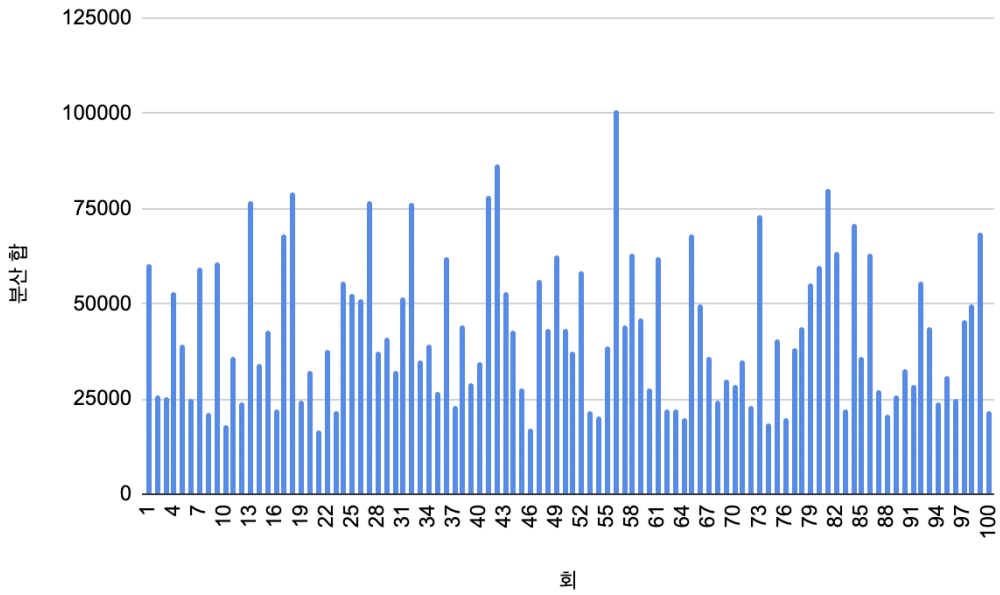


그림 6: 랜덤 함수를 이용해서 분할각을 만드는 과정을 100회 실시해서 얻어진 분산 합들

그림 6는 랜덤 함수를 이용해서 최대 500개의 분할각들을 만들어내면서 분산 합을 계산하는 과정을 100회 실시해서 얻어진 값들을 그래프로 표현한 것이다. 그림에서 보듯이 이 알고리즘을 사용해서 분할각을 만들 경우 만들어지는 분할각들은 16873.51에서 101090.88 사이의 다양한 분산 합을 가지는 것으로 나타났다.

생각할 수 있는 또 다른 방식은 지금까지 만들어진 분할각들 사이의 각들을 비교하면서 가장 값이 큰 사이각을 찾아내어 이등분하면서 분할각을 만들어내는 방식(**<사이각 이등분>**)이다. 이것을 수행하는 Python 프로그램을 작성해서 실험을 진행하였다. 작성한 프로그램을 이용해서 계산한 결과, 이 방식으로 만들어진 각들은 4235.76의 분산 합을 가지는 것으로 나타났다.

표 3: 각을 나누는 알고리즘들의 분산 합 비교

알고리즘	분산 합
황금각 반복	6093.73
랜덤 각	42593.66
사이각 이등분	4235.76

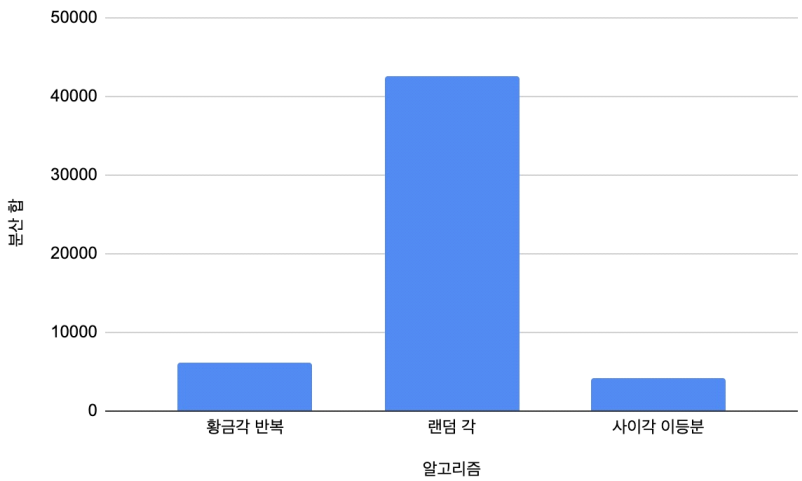


그림 7: 각을 나누는 알고리즘들의 분산 합 비교

표 3는 세 가지 알고리즘(<황금각 반복>, <랜덤 각>, <사이각 이등분>)을 사용했을 때의 분산 합들을 표로 정리한 것이고, 그림 7는 이것을 그래프로 나타낸 것이다. 그림에서 보듯이 <사이각 이등분>, <황금각 반복>, <랜덤 각> 순으로 좋은 결과를 나타내었다. <랜덤 각>의 경우와 다른 두 가지 경우는 차이가 컸지만 <황금각 반복>과 <사이각 이등분>는 차이가 크지 않았다.

이번 실험들에서 얻은 결과들을 요약하면 다음과 같다.

- 직접 작성한 Python프로그램을 사용해서, 입력값이 황금각(137.50776405...)에 가까울수록 만들어진 분할각들이 겹치는 경우가 줄어드는 것을 관찰할 수 있었다.
- 직접 작성한, 분할각들의 분산 합을 구해서 주는 Python 프로그램을 사용해서, 다양한 입력값들에 대해서 비교한 결과 황금각(137.5도)를 사용한 경우의 분산 합이 가장 작다는 것을 확인할 수 있었다.
- 황금각을 이용한 알고리즘(<황금각 반복>)을 다른 알고리즘들(<랜덤 각>, <사이각 이등분>)과 비교한 결과, <사이각 이등분>, <황금각 반복>, <랜덤 각> 순으로 좋은 결과를 나타내었다. <랜덤 각>의 경우와 다른 두 가지 경우는 차이가 컸지만 <황금각 반복>과 <사이각 이등분>는 차이가 크지 않았다.

3. 식물은 왜 황금각을 사용하도록 진화했을까?

그렇다면 식물은 왜 <황금각 반복>보다 더 좋은 결과를 내는 <사이각 이등분> 알고리즘을 사용하도록 진화하지 못하였을까 하는 궁금증이 생겼다. 이 궁금증을 해결하기 위해서 우선 <황금각 반복> 알고리즘과 <사이각 이등분> 알고리즘의 차이점에 대해서 분석해 보았다. 그 결과, <사이각 이등분> 알고리즘을 사용하기 위해서는 다음과 같은 조건을 만족해야 함을 알 수 있었다.

- 새로운 각을 만들기 위해서 지금까지 만들어진 분할각들의 값을 모두 알고 있어야 한다.
- 새로운 각을 만들기 위해서 가장 큰 사이각을 찾고 그 각을 이등분할 수 있어야 한다.
- 이러한 복잡한 작업을 수행하기 위해서는 컴퓨터의 CPU나 동물의 뇌와 같은 통제 기관이

필요하다.

그리고 <황금각 반복> 알고리즘은 다음과 같은 특징을 가짐을 알 수 있었다.

- 황금각(137.5도)만큼 회전하면서 분할각을 만드는 것은 아주 단순한 로직이다.

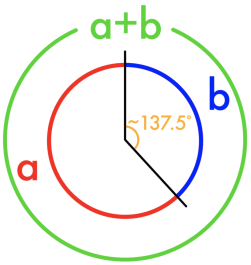
<사이각 이등분> 알고리즘을 사용하기 위해서는 아주 뛰어나지 않더라도 컴퓨터의 CPU나 동물의 뇌와 같이 복잡한 일을 수행할 수 있는 기관이 필요하다. 그런데, 이러한 복잡한 기관을 가지지 않도록 진화한 식물의 입장에서는 단순하면서도 <사이각 이등분>와 비슷한 결과를 낼 수 있는 <황금각 반복> 알고리즘이 차선의 선택이었을 것으로 생각된다. 식물의 성장 과정에서 이러한 일정한 크기의 각을 회전하면서 잎을 만들어 낼 수 있는 이유는 아마도 세포가 분열하면서 일정한 분열마다 잎을 만들게끔 유전자 속에 프로그래밍됨으로 해서 가능한 것으로 추측된다.

앞에서 고찰한 내용을 요약하면 다음과 같다.

- 식물이 성능이 더 좋은 <사이각 이등분> 알고리즘 대신 <황금각 반복> 알고리즘을 사용하도록 진화하게 된 이유는 동물의 뇌와 같은 기관이 없는 식물의 경우 복잡한 일을 수행하기 힘들기 때문에 단순하고 효율적인 <황금각 반복> 알고리즘을 이용하도록 진화한 것이 아닌가 생각된다.
- 식물이 세포 분열 과정에서 일정한 분열마다 잎을 만들어지도록 유전자 속 프로그래밍을 통해서 <황금각 반복> 알고리즘이 동작하도록 만드는 것이 아닌가 생각된다.

4. 황금각의 원리를 선분 나누기에 적용

<황금각 반복> 알고리즘과 비슷한 방식을 각이 아닌 직선에 적용할 수도 있지 않을까 하는 생각이 들었다. 이를 위해서 황금각이 137.5도에 가까운 값을 가지게 되는 과정을 좀 더 살펴보았다.
황금각은 황금비에서 나온 값이다. 그런데 황금비는 피보나치 수열과 깊은 관계가 있다. 즉,
피보나치 수열이 $F_{n+2} = F_{n+1} + F_n$으로 나타내어질 때 $\frac{F_{n+1}}{F_n}$은 n이 점점 커짐에 따라 1.61803398875...라고 하는 무리수에 점점 가까워 지는데, 이 비율 1:1.618...을 황금비라고 부른다.
황금각은 360도를 황금비로 나누어서 나온 두 각 중 작은 값이 된다. 즉,
$360 \times \frac{1}{1+1.618...} = 137.50776405...$이 된다.



(이미지 출처: https://gofiguremath.org/natures-favorite-math/the-golden-ratio/the-golden-angle/)
그림 8: 황금비로부터 황금각이 만들어지는 과정 (b:a = 1:1.618..)

그림 8는 황금비로부터 황금각이 유도되는 과정을 그림으로 표현한 것이다.

360도를 1:1.618...의 황금비로 나누어서 황금각을 만든 것과 같은 원리로 길이가 1인 선분을 황금비로 나눌 수 있는데, 이 중 작은 값은 $1 \times \frac{1}{1+1.618...} = 0.38196601125...$이 되고 큰 값은 $1 - 0.381... = 0.61803398875...$이 되는데, 이 중 큰 값은 단순히 1/황금비, 즉 $\frac{1}{1.618...} = 0.618...$과 같은 값이 된다.

식물이 황금각을 반복해서 회전시킴으로써 각을 만들어 내는 것과 같은 방식으로, 주어진 비율을 계속 더함으로써 길이가 1인 선분을 나누는 방식을 고안해 보기로 했다. 이를 위해서 1/황금비 = 0.618...의 값을 사용할 수 있다. 이때 더해진 결과가 1을 넘게 되면 정수 부분을 버리고 소수 부분만 사용하면 되는데, 이를 위해서 Python 프로그램에서는 1로 나눈 나머지를 반환하는 % 1 연산을 사용하기로 했다.

이러한 방식이 선분을 얼마나 고르게 나누는지 알아보기 위해서, 각 나누기 문제에서와 비슷하게 0에서 1사이의 비율을 입력하면 입력된 비율만큼 반복해서 더하면서 분할선들을 500개까지 만들어 내는 과정을 보여주는 Python 프로그램을 작성해 보았다.

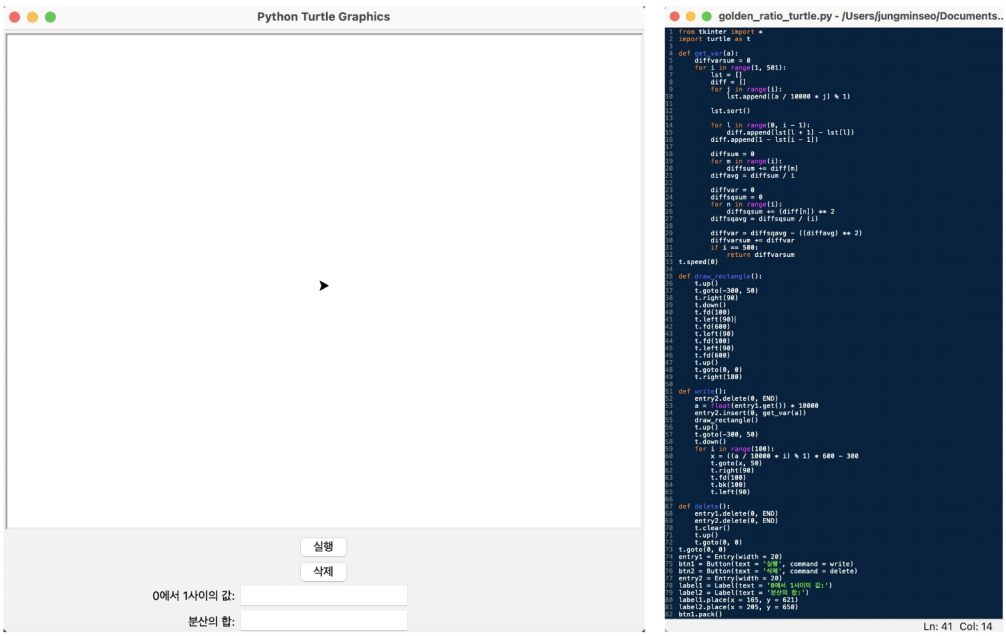


그림 9: 입력된 비율만큼 반복해서 더하면서 선분을 나누어주는 Python 프로그램

그림 9는 Turtle Graphics 라이브러리를 이용하여 직접 작성한 Python 프로그램의 화면과 소스코드를 캡쳐한 것이다. 여러 가지 비율을 직접 입력해서 입력값에 따라서 분할선들이 겹치는 정도가 차이가 나는지 실험해 보았다.

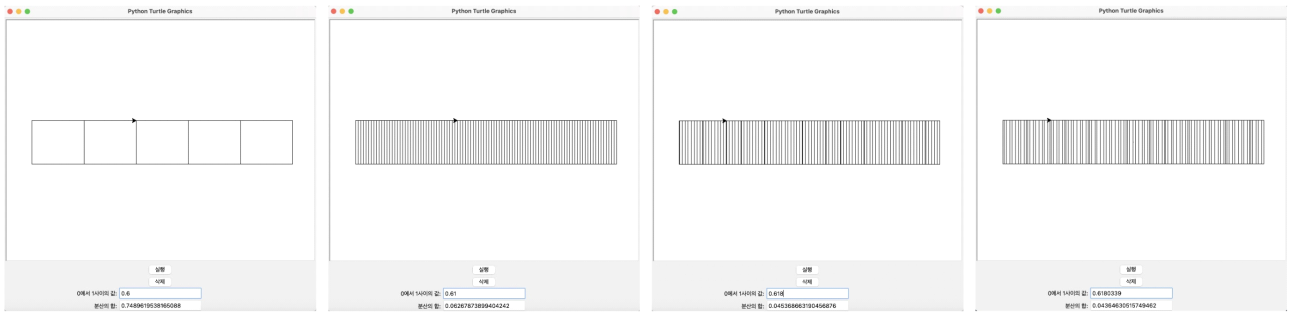


그림 10: 입력값을 다르게 했을 때 만들어지는 분할선들

그림 10는 여러 가지 비율을 입력했을 때 만들어지는 분할선들을 보여준다. 각각의 결과는 입력된 비율을 더하면서 최대 500개까지 분할선을 만들어서 그림을 그린 결과이다. 그림의 실험에서 사용된 비율들은 각각 0.6, 0.61, 0.618, 0.6180339도이다. 그림에서 보듯이 입력값이 1/황금비(실제로는 0.61803398875...의 무리수 값)에 가까울수록 분할선들이 겹치는 경우가 줄어드는 것을 관찰할 수 있었다.

분할선들이 서로 겹치는 경우가 적다고 하는 것은 다른 말로 하면 분할선들이 길이가 1인 선분 안에서 비슷한 간격을 두고 고르게 분포되어 있다는 뜻이다. 따라서 풀고자 하는 문제를 다음과 같이 정의할 수 있다.

- **[선분 나누기 문제]**: 길이가 1인 선분을 가능한 한 비슷한 크기의 1개에서부터 N개의 선분들로 계속 나누어 가는 문제

분할선들의 간격이 최대한 비슷하도록 나눈다는 것은 그 분산 값이 최소가 된다는 것과 같은 의미가 된다. 분산 값을 이용해서 좀 더 다양한 입력값들에 대해서 실험해 보기로 했다. 이것을 수행하는 Python 프로그램을 작성해서 실험을 진행하였다.

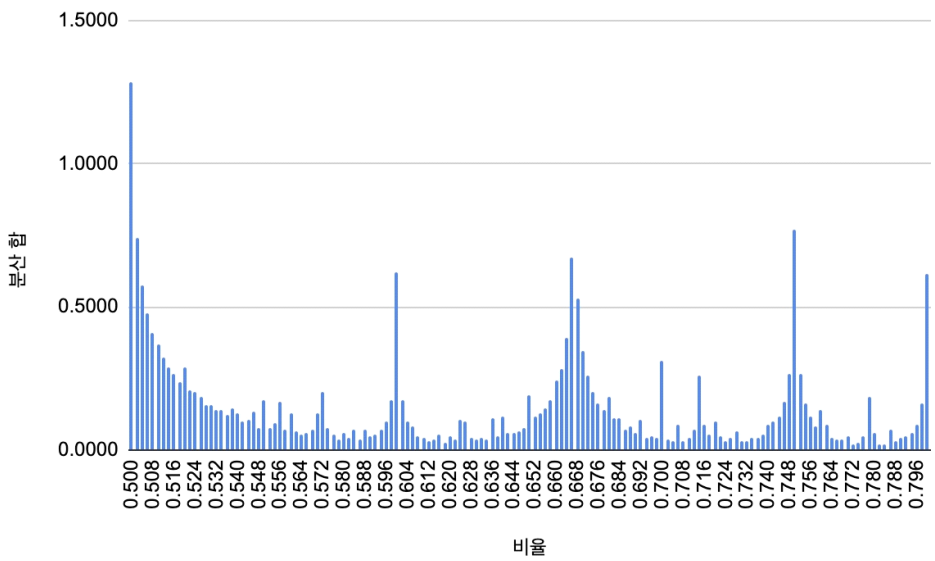


그림 11: 0.5에서 0.8 사이의 입력값에 대해서 만들어지는 분할선들의 분산 합

그림 11는 0.5와 0.8 사이의 입력값들에 대해서 최대 500개의 분할선들이 만들어지는 과정에서 생기는 간격들의 분산값들을 모두 더해서 얻어진 값(**분산 합**)들을 그래프로 표현한 것이다. 그림에서 보듯이 입력값이 1/황금비(0.618)일 때 아주 작은 분산 합을 가지는 것을 확인할 수 있었다. 이때의 분산 합은 0.0236이었는데 이보다 더 작은 분산 합을 가지는 경우는 0.772, 0.782, 0.784의 경우 밖에 없었는데 이때 분산 합은 각각 0.0218, 0.0179, 0.0221이었다.

이렇게 입력값에 대해서 더하면서 분할선을 만들어내는 방식(**<황금비 반복>**)이 아닌 다른 방식들과 비교해 보기로 했다. 분할각을 만드는 경우에서와 마찬가지로 랜덤 함수를 이용한 방식(**<랜덤 비율>**)을 생각할 수 있다. 이것을 수행하는 Python 프로그램을 작성해서 실험을 진행하였다.

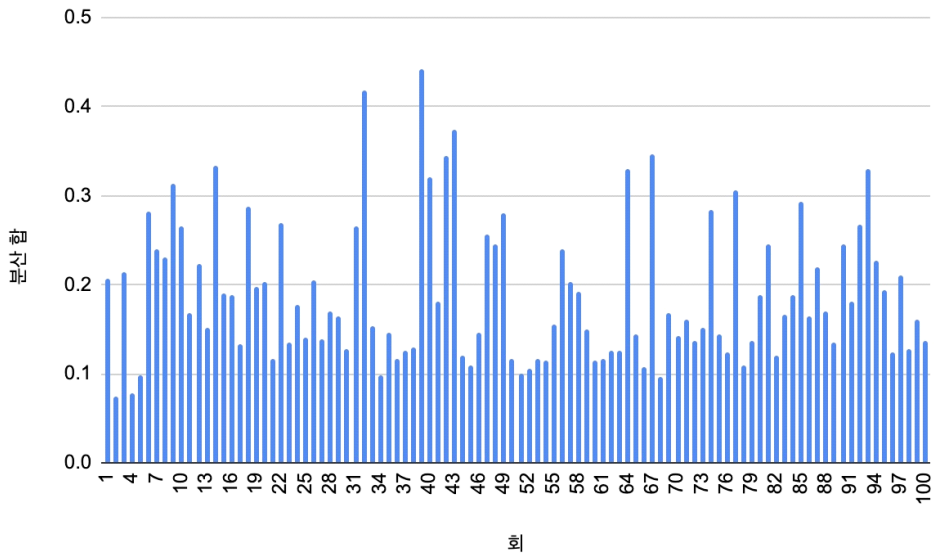


그림 12: 랜덤 함수를 이용해서 분할선을 만드는 과정을 100회 실시해서 얻어진 분산 합들

그림 12는 랜덤 함수를 이용해서 최대 500개의 분할선들을 만들어내면서 분산 합을 계산하는 과정을 100회 실시해서 얻어진 값들을 그래프로 표현한 것이다. 그림에서 보듯이 이 알고리즘을 사용해서 분할선을 만들 경우 만들어지는 분할선들은 0.0743에서 0.4430 사이의 다양한 분산 합을 가지는 것으로 나타났다.

또한, 분할각을 만드는 경우에서와 마찬가지로 지금까지 만들어진 분할선들 사이의 간격을 비교하면서 가장 값이 큰 간격을 찾아내어 이등분하면서 분할선을 만들어내는 방식(**<선분 이등분>**)을 생각할 수 있다. 이것을 수행하는 Python 프로그램을 작성해서 실험을 진행하였다. 작성한 프로그램을 이용해서 계산한 결과, 이 방식으로 만들어진 분할선들은 0.0150의 분산 합을 가지는 것으로 나타났다.

표 4: 선분을 나누는 알고리즘들의 분산 합 비교

알고리즘	분산 합
황금비 반복	0.0236
랜덤 비율	0.1895
간격 이등분	0.0150

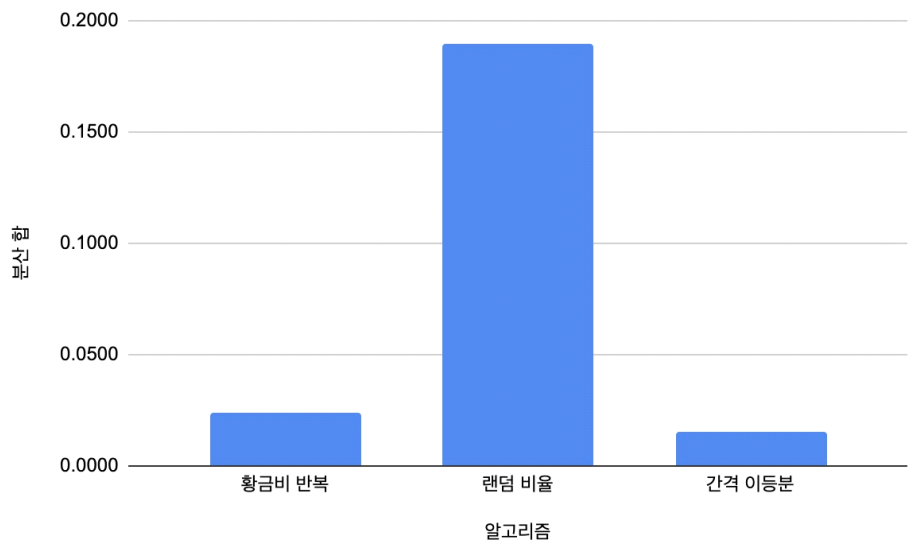


그림 13: 선분을 나누는 알고리즘들의 분산 합 비교

표 4는 세 가지 알고리즘(<황금비 반복>, <랜덤 비율>, <간격 이등분>)을 사용했을 때의 분산 합들을 표로 정리한 것이고, 그림 13는 이것을 그래프로 나타낸 것이다. 그림에서 보듯이 <간격 이등분>, <황금비 반복>, <랜덤 비율> 순으로 좋은 결과를 나타내었다. <랜덤 비율>의 경우와 다른 두 가지 경우는 차이가 컸지만 <황금비 반복>과 <간격 이등분>는 차이가 크지 않았다.

이번 실험들에서 얻은 결과들을 요약하면 다음과 같다.

- 직접 작성한 Python프로그램을 사용해서, 입력값이 1/황금비(0.61803398875...)에 가까울수록 만들어진 분할선들이 겹치는 경우가 줄어드는 것을 관찰할 수 있었다.
- 직접 작성한, 분할선들의 분산 합을 구해서 주는 Python 프로그램을 사용해서, 다양한 입력값들에 대해서 비교한 결과 1/황금비(0.618)를 사용한 경우의 분산 합이 아주 작다는 것을 확인할 수 있었다.
- 황금비를 이용한 알고리즘(<황금비 반복>)을 다른 알고리즘들(<랜덤 비율>, <간격 이등분>)과 비교한 결과, <간격 이등분>, <황금비 반복>, <랜덤 비율> 순으로 좋은 결과를 나타내었다. <랜덤 비율>의 경우와 다른 두 가지 경우는 차이가 컸지만 <황금비 반복>과 <간격 이등분>는 차이가 크지 않았다.

5. 황금비의 원리를 이용한 해시 함수: 피보나치 해싱

앞에서 황금비를 이용해서 0에서 1사이의 선분을 고르게 나눌 수 있다는 것을 알 수 있었다. 이러한 황금비의 성질을 이용해서 해시 함수를 만들 수 있지 않을까 하는 생각이 들었다.

해시 함수(hash function)는 입력값에 따라 고정된 길이의 값들 중 하나를 반환하는 함수이다. 예를 들어 입력값을 100으로 나눈 나머지를 반환하는 나머지 함수는 입력값을 [0, 1, 2, ..., 99] 중의 하나를 반환하는 해시 함수라고 할 수 있다. 이것을 Python 코드로 나타내면 그림 14와 같다.

```
def modulo_hash(i, n):
    return round(i) % n
```

그림 14: 나머지 연산을 이용하여 0에서 n사이의 정수값을 반환하는 해시 함수의 예

앞에서 1/황금비(0.618...)를 계속 더해주는 것을 반복해서 길이가 1인 선분을 나누는 분할선들을 만들어 내는 방식(<황금비 반복>)을 사용하였다. 이 방식의 원리를 이용하되 황금비를 반복해서 더하는 대신 입력된 수만큼 곱하면 총 연산 회수를 줄일 수 있다. 여기에 버림 연산(round())과 나머지 연산(%)을 적절히 사용하게 되면 0에서 n사이의 정수값을 반환하는 함수를 얻을 수 있게 된다. 이것을 Python 코드로 나타내면 그림 15와 같다.

```
golden_ratio = 0.61803398875

def fibonacci_hash(i, n):
    return round(golden_ratio * i * n) % n
```

그림 15: <황금비 반복> 알고리즘을 이용하여 0에서 n사이의 정수값을 반환하는 피보나치 해싱 함수

이렇게 만들어진 해시 함수를 **<피보나치 해싱>** 함수라고 부르기로 했다.

이번 연구에서 얻은 결과들을 요약하면 다음과 같다.

- 황금비 반복 알고리즘을 응용해서 해시 함수(<피보나치 해싱>)를 만들 수 있었다.

6. 피보나치 해싱을 실생활에 적용해 보기: 수영장 락커룸

앞에서 만든 피보나치 해싱 함수를 실생활에 적용해 보기 위해서 수영장 락커 배정 프로그램을 만들어 보기로 했다. 평소에 살고 있는 동네의 수영장을 가게 되면 탈의실 락커를 배정해 주는 키오스크가 있는데 갈 때마다 몇몇 곳들만 붐비고 다른 곳은 한가한 경우를 많이 보게 되었다. 해시 함수를 이용해서 사람들을 고르게 배정하게 되면 이런 문제를 해결할 수 있지 않을까 하는 생각이 들었다.

그림 16: 수영장 락커 배정 과정

그림 16은 수영장의 락커 배정 과정을 그림으로 표현한 것이다. 그림에서 보듯이 락커룸에는 모두 300개의 락커가 일렬로 늘어서 있는 것으로 가정했고 모두 100명의 사람들이 차례로 입장한다고 가정했다.

우선 앞에서 만든 <피보나치 해싱> 함수를 이용해서 락커를 배정하는 알고리즘을 만들어 보았다. 새로운 사람이 들어오게 되면 우선 1번부터 입장번호를 배정하고 이 입장번호와 락커의 총 개수인 300을 <피보나치 해싱> 함수의 입력으로 넘겨서 0에서 299 사이의 정수값을 반환받을 수 있는데 이 값을 새롭게 배정하는 락커의 번호로 사용하게 된다. 만일 이미 다른 사람이 사용하고 있는 번호로 배정될 경우가 생기면 입장번호 * 2한 값으로 다시 해시 함수를 호출해서 새 락커 번호를 받는 것을 반복한다.

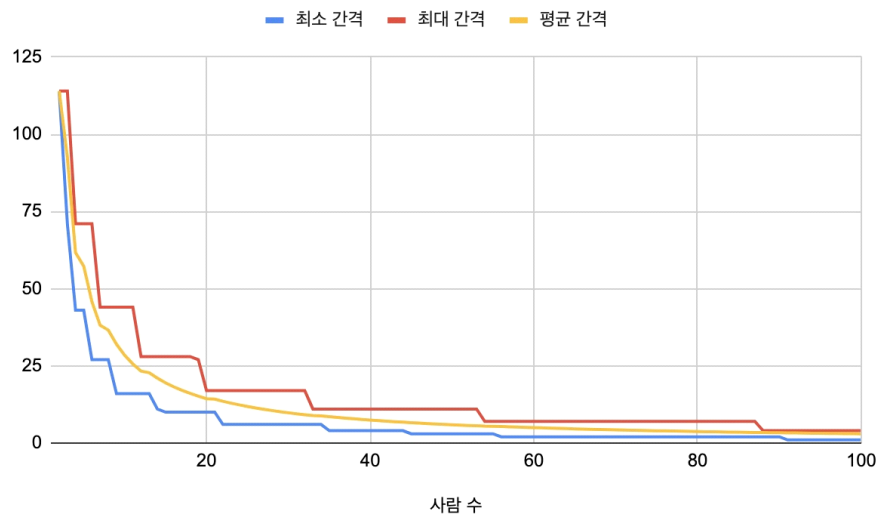


그림 17: 피보나치 해싱을 이용한 락커 배정 알고리즘의 최소, 최대 및 평균 간격 변화

그림 17은 피보나치 해싱 알고리즘을 사용해서 락커를 배정했을 경우의 배정받은 락커들의 간격의 최소, 최대 및 평균값의 변화를 그래프로 표현한 것이다. 입장하는 사람들이 늘어남에 따라 간격이

점차 줄어드는 것을 확인할 수 있다. 따라서 입장하는 사람의 수가 늘어남에 따라서 배정된 락커가 적절히 분산되는 것을 알 수 있다.

피보나치 해싱을 이용한 락커 배정 알고리즘의 성능이 어느 정도인지 알아보기 위해서 앞에서 사용했던 랜덤 함수를 이용한 방식과 간격 이등분을 이용한 방식과 비교해 보기로 했다.

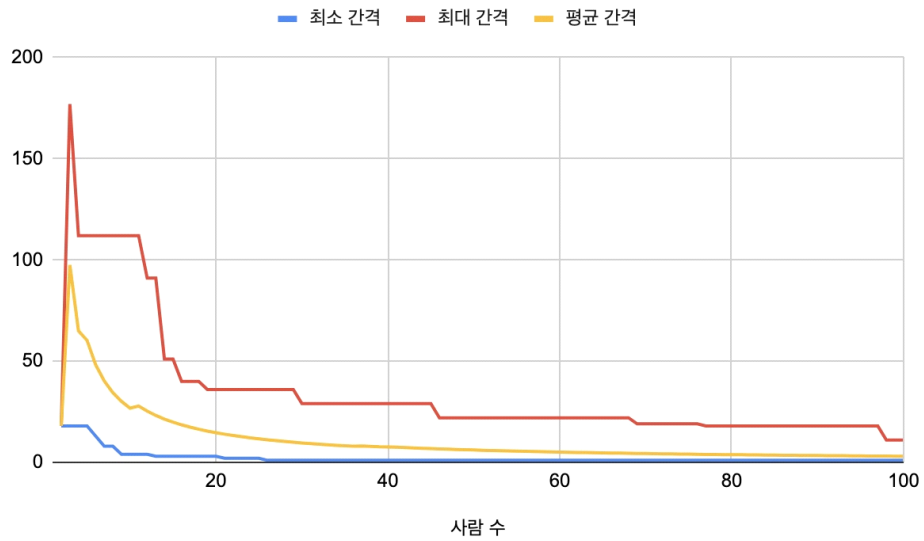


그림 18: 랜덤 함수를 이용한 락커 배정 알고리즘의 최소, 최대 및 평균 간격 변화

그림 18은 랜덤 함수 알고리즘을 사용해서 락커를 배정했을 경우의 배정받은 락커들의 간격의 최소, 최대 및 평균 간격의 변화를 그래프로 표현한 것이다. 피보나치 해싱 알고리즘의 경우보다 최소 간격과 최대 간격의 차이가 큰 것을 확인할 수 있었다.

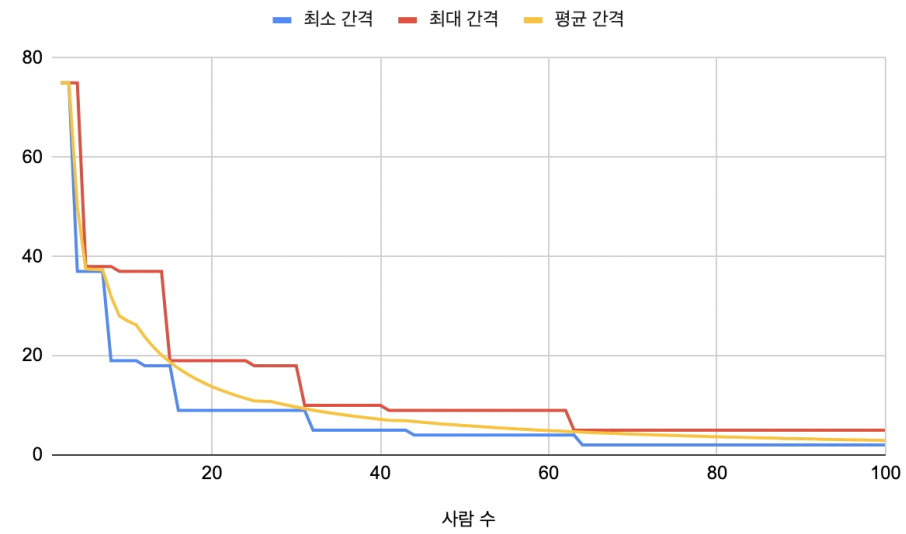


그림 19: 간격 이등분을 이용한 락커 배정 알고리즘의 최소, 최대 및 평균 간격 변화

그림 19는 간격 이등분 알고리즘을 사용해서 락커를 배정했을 경우의 배정받은 락커들의 간격의

최소, 최대 및 평균 간격의 변화를 그래프로 표현한 것이다. 랜덤 함수 알고리즘의 경우보다 최소 간격과 최대 간격의 차이가 작은 것을 확인할 수 있었다.

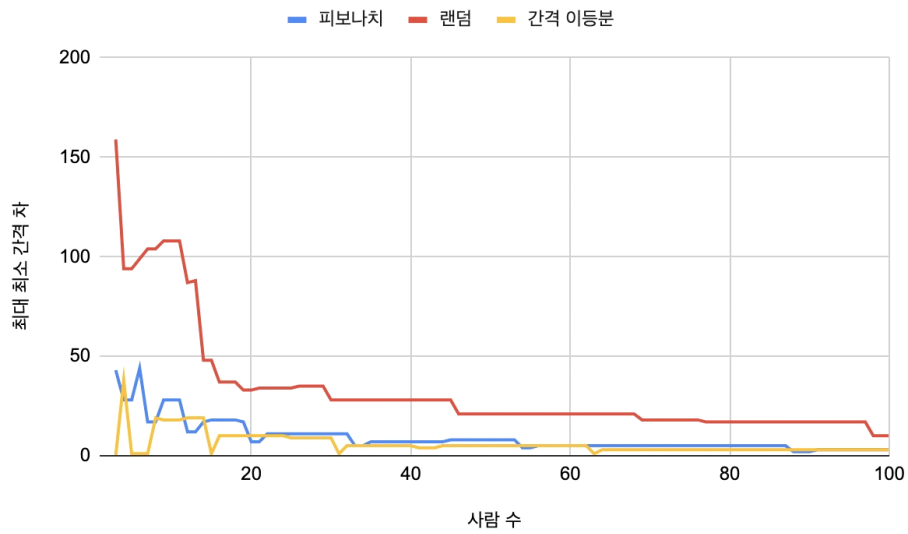


그림 20: 락커 배정 알고리즘들 간의 최대 최소 간격 차이 비교

그림 20은 세 가지 알고리즘들(<피보나치 해싱>, <랜덤 함수>, <간격 이등분>) 성능을 직접적으로 비교해 보기 위해서 최대 간격과 최소 간격의 차이를 나타내는 최대 최소 간격 차를 그래프로 그린 것이다. 그림에서 보듯이 <피보나치 해싱> 알고리즘과 <간격 이등분> 알고리즘이 <랜덤 함수> 알고리즘보다 좋은 결과를 내는 것을 볼 수 있다. <간격 이등분> 알고리즘이 <피보나치 해싱> 알고리즘과 비교해서 비슷하거나 조금 더 좋은 결과를 내는 것을 알 수 있었다.

이번 연구에서 얻은 결과들을 요약하면 다음과 같다.

- 수영장 락커룸의 효율적인 락커 배정 문제를 풀기 위해서 <피보나치 해싱> 알고리즘을 사용하였다.
- <피보나치 해싱> 알고리즘을 사용하게 되면 입장하는 사람의 수가 늘어남에 따라서 배정된 락커가 적절히 분산되는 것을 확인할 수 있었다.
- <랜덤 함수> 알고리즘과 <간격 이등분> 알고리즘과 비교했을 때, <랜덤 함수> 알고리즘보다는 훨씬 좋은 결과를, <간격 이등분> 알고리즘과 비교했을 때는 비슷하거나 약간 못한 결과를 보여준다는 것을 확인할 수 있었다.

VI 결론

식물의 잎이 나는 경우에서와 같이 황금각 만큼 반복해서 회전하면서 각을 만들게 되면 겹치는 경우가 적은지 직접 확인해보기 위해서 Turtle Graphics 라이브러리를 사용하는 Python 프로그램을 직접 만들어서 확인하는 실험을 진행하였다. 실험 결과, 황금각에 가까운 값을 넣을수록 만들어진

각들이 겹쳐지는 경우가 줄어드는 것을 직접 확인할 수 있었다.

보다 다양한 각에 대해서 비교해보기 위해서 만들어진 각들이 이루는 사이각들의 분산을 모두 더해서 비교하는 실험을 진행하였다. 실험 결과, 황금각을 사용했을 때 만들어지는 사이각들의 분산 합이 가장 작은 값을 가지는 것을 확인할 수 있었다. 이 문제를 [각 나누기 문제]라고 정의하고 이 문제를 풀 수 있는 <랜덤 각> 알고리즘과 <사이각 이등분> 알고리즘과 비교했을 때에도 <황금각 반복> 알고리즘이 좋은 결과를 보임을 확인할 수 있었다.

식물이 성능이 더 좋은 <사이각 이등분> 알고리즘과 비슷한 방식을 사용하도록 진화한 대신 <황금각 반복> 알고리즘을 사용하는 식으로 진화한 이유에 대해서 고찰해본 결과 다음과 같은 결론을 얻을 수 있었다. 첫째, 동물의 뇌와 같은 기관이 없는 식물의 경우 <사이각 이등분> 알고리즘과 같은 복잡한 작업을 수행하기 힘들고 대신 단순하면서 효율적인 <황금각 반복> 알고리즘을 이용하는 쪽으로 진화하였을 것이다. 둘째, 식물이 세포 분열 과정에서 일정한 수의 분열마다 잎이 만들어지도록 유전자 프로그래밍을 통해서 <황금각 반복> 알고리즘이 작동하도록 만드는 것이 아닌가 생각된다.

황금각과 비슷한 원리를 적용하면 선분을 나누는 선들도 서로 겹치지 않게 만들 수 있을지 않을까 하는 생각에 1/황금비 만큼 반복해서 더하면서 선분을 나누는 방식을 적용해보았다. Turtle Graphics 라이브러리를 사용한 Python 프로그램을 직접 만들어서 확인한 결과, 1/황금비에 가까운 값을 넣을수록 만들어진 선들이 서로 겹치는 경우가 줄어드는 것을 직접 확인할 수 있었다.

황금각에서와 마찬가지로 분산 합을 비교하는 실험을 진행한 결과, 1/황금비의 값을 적용했을 때 분산 합이 아주 작은 값을 가지는 것을 확인할 수 있었다. 이 문제를 [선분 나누기 문제]라고 정의하고 이 문제를 풀 수 있는 <랜덤 비율> 알고리즘과 <간격 이등분> 알고리즘과 비교했을 때에도 <황금비 반복> 알고리즘이 좋은 결과를 보임을 확인할 수 있었다.

황금각과 황금비의 원리를 응용해서 해시 함수(입력값에 따라 고정된 길이의 값들 중 하나를 반환하는 함수)를 만들 수 있었는데, 이 함수를 <피보나치 해싱> 함수라고 부르기로 했다.

<피보나치 해싱> 함수를 수영장 락커 배정 문제에 적용해 보았다. 평소에 살고 있는 동네의 수영장을 가게 되면 키오스크를 통해서 배정된 락커들이 한쪽으로 쏠려서 몇몇 곳들만 붐벼서 비슷한 시간대에 입장한 사람들이 서로 불편한 경우를 많이 경험하였기 때문에, 이 문제를 해시 함수를 사용해서 해결할 수 있지 않을까 하는 생각이 들었기 때문이다. <피보나치 해싱>을 이용한 락커 배정 알고리즘을 가지고 실험한 결과, 입장하는 사람이 늘어나더라도 배정된 락커들 사이의 간격이 고르게 유지된다는 것을 확인할 수 있었다.

<랜덤 함수> 알고리즘과 <간격 이등분> 알고리즘과 비교하기 위해서 간격이 얼마나 고른지를 나타내는 최대 간격과 최소 간격의 차이를 비교한 결과, <랜덤 함수> 알고리즘 보다는 아주 좋은 결과를, <간격 이등분> 알고리즘과 비교했을 때에는 비슷하거나 조금 못한 결과를 내는 것을 확인할 수 있었다.

VII 전망과 활용성

식물이 황금각을 이용해서 잎을 만드는 원리를 이용해서 <피보나치 해싱> 함수를 만들 수 있었고 이 함수를 이용해서 수영장의 락커룸의 락커를 고르게 배정하는 문제를 해결할 수 있었다. 이러한 원리는 다른 여러 문제들을 해결하는데에도 활용될 수 있을 것으로 예상된다. 다음은 앞으로의 발전시킬 방향과 어디에 활용하면 좋을지 아이디어들을 요약한 것이다.

- 주어진 입력 값을 고르게 분포하도록 만들어주는 해시 함수의 성질을 이용할 수 있는 다양한 문제들에 적용할 수 있을 것 같다. (예를 들어 주차장의 주차공간 배정 문제, 자동차 영화관 좌석 자동 배정 문제 등..)
- 식물과 같이 중앙에서 명령을 내리는 기관이 없는 경우에 해당하는 문제들에 적용할 수 있을 것 같다. 예를 들어 재난이 발생한다든지 해서 인터넷이나 모바일 네트워크가 동작하는 않는 상황에서 대피 통로를 배정해야 하는 상황을 생각해 볼 수 있을 것 같다.

VIII 참고 문헌

1. 참고문헌

- 황금각: https://en.wikipedia.org/wiki/Golden_angle
- 황금각: https://gofiguremath.org/natures-favorite-math/the-golden-ratio/the-golden-angle/
- 황금비: https://ko.wikipedia.org/wiki/%ED%99%A9%EA%B8%88%EB%B9%84
- 해시 함수: https://ko.wikipedia.org/wiki/%ED%95%B4%EC%8B%9C_%ED%95%A8%EC%88%98

2. 데이터 분석 스프레드시트

- 황금각, 황금비, 락커룸: https://docs.google.com/spreadsheets/d/1TqtMHlamgjhWCuLaqAgLKv4zxsiJdxzPteb3AVw_tuw/edit?gid=0#gid=0

3. 소스 코드

- 황금각 실험: https://github.com/seo-family/korea-science-fair-2024-2025/tree/main/golden-angle
- 황금 비 실험: https://github.com/seo-family/korea-science-fair-2024-2025/tree/main/golden-ratio
- 락커룸 실험: https://github.com/seo-family/korea-science-fair-2024-2025/tree/main/locker-room