chapter1. 알고리즘의 소개

- Hello Coding 그림으로 개념을 이해하는 알고리즘 -

TABLE OF CONTENTS

1. 이진 탐색

전화번호부에서 누군가의 번호를 찾고 있는 중이라고 가정해봅시다.
찾을 사람의 이름은 K로 시작합니다.
처음부터 페이지를 한 장씩 넘기면서 찾는방법이 있지만 너무 비효율적입니다.
이 방법보다는 일단 책 한가운데를 펼치는 방법을 사용하는게 더 효율적일것입니다.

페이스북에 로그인한다고 가정해봅시다.
계정이 실제로 존재하는지 데이터베이스에서 아이디를 찾아야 합니다. 아이디가 k로 시작한다면 페이스북은 알파벳 a부터 시작해서 차례대로 아이디를 찾을 수도 있겠지만, 중간에 어디쯤에서 찾기 시작하는것이 더 나을 수도 있습니다.

이런 문제를 **탐색(search)문제**라고 합니다. 그리고 위에서 예를 들었던 모든 경우에 **이진 탐색(binary search)**이라고 하는 알고리즘을 사용할 수 있습니다.

```python
def binary_search(list, item):
  low = 0
  high = len(list) - 1

  while low <= high:
    mid = (low + high) // 2
    guess = list[mid]
    if guess == item:
      return mid
    if guess > item:
      high = mid - 1
    else:
      low = mid + 1
  return None


list_a = [1,3,5,7,8,9,15,25,46]

print(binary_search(list_a, 25))
print(binary_search(list_a, -1))
```

```
7
None
```

low와 high는 전체 리스트 중에서 어떤 부분을 탐색해야 하는지 알려줍니다.

while low <= high: → 탐색 범위를 하나로 줄이지 못하면 계속 실행합니다.

mid = (low + high) // 2 → 가운데 숫자를 확인

if guess == item: → 아이템을 찾았습니다.

if guess > item: → 추측한 숫자가 너무 큽니다.

else: → 추측한 숫자가 너무 작습니다.

return None → 아이템이 리스트에 없습니다.

2. 실행 시간

리스트에 100개의 원소가 있는 경우 처음부터 일일이 다 세는 단순탐색은 100번 추측해야 합니다. 원소가 40억개라면 40억번 추측해야 합니다. 그러니까 추측해야 할 최대 횟수는 리스트의 길이와 같습니다. 이런 것을 선형시간이라고 합니다.

이진 탐색은 원소 개수가 100개라면 7번만 추측해도 됩니다. 40억개라면 32번만 추측하면 됩니다. 이진 탐색의 경우에는 로그 시간으로 실행됩니다.

〈단순 탐색〉 : 100개 → 100번, 40억 개 → 40억번, 선형 시간(On)
〈이진 탐색〉 : 100개 → 7번, 40억 개 → 32번, 로그 시간(Log n)

3. 빅오 표기법

빅오 표기법은 알고리즘이 얼마나 빠른지 표시하는 특별한 방법입니다.

원소의 개수가 증가해도 이진 탐색에 걸리는 시간은 얼마 늘어나지 않습니다.

하지만 단순 탐색의 시간은 엄청나게 증가하죠.

그러니까 원소의 개수가 커질수록 이진 탐색은 단순 탐색보다 훨씬 빨라지는 겁니다.

빅오 표기법은 알고리즘이 얼마나 빠른지를 말해줍니다. 예를 들어, 리스트의 크기가 n이라고 가정해봅시다.

단순 탐색은 n번을 연산해야 합니다. 그래서 빅오 표기법에 따른 실행 시간은 O(n)입니다.

빅오 표기법은 속도를 시간 단위로 세지 않습니다.

“초”와 같은 시간 단위는 어디로 갔을까요? **빅오 표기법**은 *속도를 시간 단위로 세지 않습니다.*

빅오 표기법은 연산 횟수를 비교하기 위한 것입니다. **빅오 표기법**을 사용하면 수행해야 할 일이 많아질 때 알고리즘에 걸리는 시간이 어떤 식으로 증가하는지를 알 수 있습니다.

O(log n), 로그시간: 예) 이진 탐색
O(n), 선형시간: 예) 단순 탐색
O(n * log n): 예) 퀵 정렬(4장에 나옵니다)과 같이 빠른 정렬 알고리즘
O(n의제곱): 예) 선택 정렬(2장에 나옵니다)과 같이 느린 정렬 알고리즘
O(n!): 예) 외판원 문제와 같이 정말 느린 알고리즘

위에서부터 아래로 갈수록 느려집니다.

〈요약〉

1. 이진 탐색은 단순 탐색보다 아주 빠릅니다.
2. O(log n)은 O(n)보다 빠릅니다. 리스트의 원소의 개수가 증가하면 상대적으로 더 빨라집니다.
3. 알고리즘의 속도는 시간으로 측정하지 않습니다.
4. 알고리즘의 시간은 어떻게 증가하는가로 측정합니다.
5. 알고리즘의 시간은 빅오 표기법으로 나타냅니다.