알고리즘과 자료구조 워크북

교과목명 : 알고리즘과 자료구조
차시명: 2차시 Brute-force 전략1

◈ 담당교수: 신일훈 **(서울과학기술대학교)**

◉ 세부목차
- brute-force 전략의 개념
- brute-force 전략의 적용1: 합계를 구하는 알고리즘
- brute-force 전략의 적용2: 최대값, 최소값을 구하는 알고리즘
- brute-force 전략의 적용3: 약수의 개수를 구하는 알고리즘

학습에 앞서

■ 학습개요

Brute-force(억지) 전략은 문제의 해답을 찾기 위해 가능한 솔루션을 전부 시도하는 단순한 방법이다. 가령 N개의 좌표가 주어지고, 두 좌표 중 가장 짧은 거리를 찾는 문제라고 하면, 가능한 N*(N-1)/2의 경우에 대해 모두 거리를 계산하고 이들 중 최소값을 찾는 방식으로 문제를 해결한다. 본 장에서는 먼저 brute-force 전략의 개념을 이해하고, 이를 적용하여 합계 구하기, 최대값 찾기, 약수의 개수 구하는 문제를 해결한다.

■ 학습목표

1	brute-force 전략의 개념을 이해한다.
2	합계 구하는 알고리즘을 설계하고 표현할 수 있다.
3	최대값 구하는 알고리즘을 설계하고 표현할 수 있다.
4	약수의 개수를 구하는 알고리즘을 설계하고 표현할 수 있다.

■ 주요용어

용어	해설
brute-force	어떤 문제의 해답을 구하기 위해 가능한 모든 경우를 모두 시도하는 방법
for 반복문	원소의개수만큼 반복하는 반복문
파이썬 리스트	여러 개의 값을 저장할 수 있는 파이썬 자료형. []로 표시한다.
문자열	여러 개의 문자로 구성된다. 파이썬에서는 “” 또는 ‘’로 표시한다.

학습하기

1. brute-force (억지) 전략 개념

가. 개념

- 답을 찾기 위해, 모든 가능한 경우를 전부 확인하거나 수행하는 단순한 방법
- 주로 반복 기법 활용
- 컴퓨터의 빠른 성능을 활용

나. brute-force (억지) 전략 예시

- 1-100까지의 합계를 구하는 방법
  - sum = 1 + 2 + … + 100
  - 합계를 구할 때, 단순하게 대상 숫자들을 모두 더하여 합계를 구한다.
  - 반복문으로 구현한다.
- 9^n을 구하는 방법
  - result = 9 * 9 * … * 9
  - n제곱을 구할 때, 단순하게 대상 숫자를 n번 곱하여 결과값을 구한다.
  - 반복문으로 구현한다.
- 오름차순으로 정렬된 숫자들의 리스트에서 최대값 찾는 방법
  - 첫번째 값을 max로 초기화
  - 리스트의 두번째부터 마지막 원소를 현재의 max와 비교하여, max가 작으면, max를 리스트의 원소값으로 변경.
  - 반복문으로 구현한다.
- 서울에서 출발하여, 대전, 부산, 광주를 모두 방문하고 다시 서울로 돌아오는 최단경로는? (방문 순서는 상관없음)
  - 경로1: 서울 - 대전 - 부산 - 광주 - 서울
  - 경로2: 서울 - 대전 - 광주 - 부산 - 서울
  - …
  - 가능한 모든 경로에 대해 각각 비용을 계산하고, 이 중 최소값을 가진 경로를 선택한다.

다. brute-force (억지) 전략 특성

- 상대적으로 알고리즘을 설계하기 쉽다.
- 광범위한 문제에 적용 가능하다.
- 입력(데이터)의 크기(개수)가 작은 경우에 효과적이다.
- 입력(데이터)의 크기(개수)가 큰 경우, 시간복잡도가 지나치게 높을 수 있고 결과적으로 성능이 낮을 수 있다.
- 더 효율적인 알고리즘을 통해 성능을 개선하는 것이 가능할 수 있다.

2. brute-force 전략의 적용1: 합계 구하기

가. 문제

- 1부터 자연수 N까지의 합계를 구하시오.

나. 아이디어

- 합계를 저장할 변수 sum을 0으로 초기화한다.
- 1 ~ N까지의 수를 차례로 더하여 sum에 저장한다.
- 즉, 처음에는 sum에 1을 더하여 그 결과를 다시 sum에 저장한다.
- 다음으로는 sum에 2를 더하여 그 결과를 다시 sum에 저장한다.

- 이러한 작업을 N을 더할 때까지 반복한다.

다. 알고리즘(의사코드로 표현)

```
sum = 0
for num in 1-N :
        sum = sum + num
print(sum) # 생략 가능
```

- for num in 1-N의 의미는 num을 1 ~ N까지 증가시키면서 for 문의 내용을 반복하여 실행하라는 의미이다.
- 즉 for문은 N번 반복되며 num의 값은 1 ~ N으로 변화된다.
- 따라서 처음 반복 시에는 sum = sum + 1이 실행되고, 두 번째 반복될 때는 sum = sum + 2가 실행된다.
- 최종적으로는 sum = sum + N이 실행된다.
- 결과적으로 sum 에 1 ~ N 까지의 합계가 저장된다.

라. 알고리즘 최악 시간복잡도

- for 문이 N번 반복되므로 for문의 반복 회수가 입력 데이터의 크기인 N에 비례한다.
- 따라서 알고리즘의 최악 시간복잡도는 O(N)이다.
- 이 경우에는 최선, 평균 시간복잡도도 모두 O(N)이다.

마. 파이썬 구현 및 실행1

```
N = 10
sum = 0
for num in range(1, N+1) :
        sum = sum + num
print(sum)
```

- range(1, N+1)은 [1, 2, 3, ..., N]을 반환한다.
- 따라서 위 for 문은 for num in [1, 2, 3, ..., N] : 과 동일하다.
- 따라서 for 문은 N번 반복되며, num의 값은 1 ~ N으로 변화된다.
- 결과적으로 위 파이썬 코드는 1 ~ 10까지의 합계를 구하여 출력한다.
- 다음은 이 파이썬 코드를 스파이더에서 실행한 결과 화면이다.

```
In [14]: runfile('D:/data/lecture/
파이썬으로배우는자료구조와알고리즘/code/알고리즘/
untitled1.py', wdir='D:/data/lecture/
파이썬으로배우는자료구조와알고리즘/code/알고리즘')
55
```

바. 파이썬 구현 및 실행2

```
def cal_sum(N) :
        sum = 0
        for num in range(1, N +1) :
                sum = sum + num
        return sum

print(cal_sum(10))
```

- 위 코드는 합계를 구하는 cal_sum() 함수를 먼저 정의하고 이 함수를 호출함으로써 1 ~ 10까지의 합

계를 구한다.
- 함수를 정의하지 않았던 첫 번째 방법보다 코드의 재활용성이 높은 장점이 있다

3. brute-force 전략의 적용2: 최대값 구하기

가. 문제
- N개의 숫자를 저장한 파이썬 리스트에서 가장 큰 값을 출력하시오.
- 가령 대상 리스트가 [5, 3, 2, 1, 7, 9, 6] 라면, 최대값 9를 출력해야 한다.

나. 아이디어
- 최대값을 저장할 변수 max를 사용한다.
- max를 어떤 값으로 초기화할 지가 매우 중요하다.
- 만약 0으로 초기화하면, 대상 리스트의 값들이 모두 음수인 경우 최대값이 0으로 출력되는 오류가 발생한다.
- 따라서 최대값은 나올 수 있는 가장 작은 값으로 초기화하거나 대상 리스트의 값들 중 하나로 초기화해야 한다.
- 이후에는 max와 리스트에 저장된 각각의 값들을 차례로 비교하며, 만약 기존의 max 보다 리스트의 값이 크다면 max를 해당 값으로 수정한다.
- 이러한 비교 및 업데이트 작업을 리스트의 모든 값에 대해 반복하면, max가 최대값을 저장하게 된다.

다. 알고리즘(의사코드로 표현)

```
max = list[0]
for num in list[1] ~ list[N-1] :
        if (max < num) :
                max = num
print(max)
```

- list[0]은 list의 첫 번째 값을 의미한다.
- 즉, max는 list의 첫 번째 값으로 초기화된다.
- for문은 N-1 번 반복되며 num의 값은 list[1] ~ list[N-1] 로 변화된다.
- 따라서 처음 반복 시에는 max와 list[1]을 비교하며, 두 번째 반복될 때는 max와 list[2]를 비교한다.
- 최종적으로는 max와 list[N-1]을 비교하며, 결과적으로 max에 최대값이 저장된다.

라. 알고리즘 최악 시간복잡도
- for 문이 N-1번 반복되므로 for문의 반복 회수가 입력 데이터의 크기인 N에 비례한다.
- 따라서 알고리즘의 최악 시간복잡도는 O(N)이다.
- 이 경우에는 최선, 평균 시간복잡도도 모두 O(N)이다. (N개의 값들을 모두 비교해야만 최대값을 찾을 수 있기 때문)

마. 파이썬 구현 및 실행

```
def find_max(list) :
        max = list[0]
        for num in list:
                if (max < num) :
                        max = num
        return max

print(find_max([5, 3, 2, 1, 7, 9, 6]))
```

- for num in list: 구문은 list에 저장된 값의 개수만큼 반복 실행하라는 의미이다. list의 값 값이 num에 저장되며 반복문이 실행된다.
- 위의 예에서 list의 값의 개수는 7개이므로 for 문은 7번 반복된다,
- 첫 번째 반복 시에는 5가 num에 저장되고, 두 번째 반복 시에는 num에 3이 대입된다.
- 엄밀하게 따진다면 list의 첫 번째 값으로 max가 초기화되었으므로 list의 두 번째 값부터 비교해야 하며, 첫 번째 값과 비교하는 것은 불필요하다. 하지만 코드의 단순함을 위해 위와 같이 구현하였으며 연산을 한 번만 더 실행하므로 성능 저하가 거의 없고 결과는 동일하다.
- 다음은 이 파이썬 코드를 스파이더에서 실행한 결과 화면이다.

```
In [16]: runfile('D:/data/lecture/
파이썬으로배우는자료구조와알고리즘/code/알고리즘/
untitled1.py', wdir='D:/data/lecture/
파이썬으로배우는자료구조와알고리즘/code/알고리즘')
9
```

4. brute-force 전략의 적용2: 최대값 구하기

가. 문제

- 자연수 N의 약수의 개수를 출력하시오.
- 가령 N이 10 이라면, 약수가 1, 2, 5, 10 이므로 개수인 4를 출력해야 한다.

나. 아이디어

- N의 약수는 N을 해당 수로 나누었을 때 나머지가 0인 수이다.
- N의 약수는 N 이하의 숫자이다.
- 따라서 1 ~ N 까지의 모든 숫자에 대해 해당 숫자가 약수인지를 검사하면 약수의 개수를 구할 수 있다.
- 약수의 개수를 저장할 변수 count를 사용하여 0으로 초기화하고, 1 ~ N의 숫자 중 약수라고 판명될 때마다 count를 1 만큼 증가하면 된다.

다. 알고리즘(의사코드로 표현)

```
count = 0
for num in 1 ~ N :
        if (N % num == 0) :
                count = count + 1
print count
```

- % 연산자는 나머지를 구하는 연산자이다. N을 num으로 나눈 나머지를 반환한다.
- 따라서 if 문은 num이 N의 약수일 때 참이 된다.
- count = count + 1은 count를 1 증가시키라는 의미이며, count += 1로도 쓸 수 있다.

라. 알고리즘 최악 시간복잡도

- for 문이 N번 반복되므로 for문의 반복 회수가 입력 데이터의 크기인 N에 비례한다.
- 따라서 알고리즘의 최악 시간복잡도는 O(N)이다.
- 이 경우에는 최선, 평균 시간복잡도도 모두 O(N)이다. (1 ~ N의 숫자들을 모두 검사해야 하기 때문)

마. 파이썬 구현 및 실행

```
def count_divisors(N) :
        count = 0
        for num in range(1, N+1) :
                if (N % num == 0) :
                        count = count + 1
        return count

print(count_divisors(10))
```

- 위의 파이썬 코드는 의사 코드와 거의 동일하다.
- 다음은 이 파이썬 코드를 스파이더에서 실행한 결과 화면이다.

```
In [18]: runfile('D:/data/lecture/
파이썬으로배우는자료구조와알고리즘/code/알고리즘/
untitled1.py', wdir='D:/data/lecture/
파이썬으로배우는자료구조와알고리즘/code/알고리즘')
4
```

연습문제

1. brute-force 전략에 대해 설명하시오.

정답 : 문제의 해답을 찾기 위해 가능한 솔루션을 전부 시도하는 단순한 방법이다.
해설 : brute-force는 억지 전략이라고도 하며, 문제의 해답을 찾기 위해 가능한 솔루션을 전부 시도하는 단순한 방법이다.

2. brute-force 전략의 단점에 대해 설명하시오.

정답 : 시간복잡도가 높기 때문에 입력의 크기가 클 때 알고리즘의 효율성이 낮을 수 있다.
해설 : brute-force 전략은 가능한 모든 경우를 시도하므로 입력의 크기가 크면 알고리즘의 효율성이 크게 낮아질 수 있다.

3. 1-N까지의 합계를 구하는 알고리즘을 의사코드로 표현하시오.

정답 :

```
sum = 0
for num in 1 ~ N :
        sum += num
print(sum)
```

해설 : 결과를 저장할 변수 sum을 0으로 초기화하고 1 ~ N까지의 숫자에 대해 반복하며 이를 sum에 더해준다.
최종적으로 sum을 출력한다.

정리하기

1. brute-force 전략은 문제의 해답을 찾기 위해 가능한 솔루션을 전부 시도하는 단순한 방법

이다.

2. 합계 구하는 알고리즘을 brute-force 전략을 활용하여 설계할 수 있다.
3. 최대값을 구하는 알고리즘을 brute-force 전략을 활용하여 설계할 수 있다.
4. 약수의 개수를 구하는 알고리즘을 brute-force 전략을 활용하여 설계할 수 있다.

참고자료

- 파이썬 알고리즘, 최영규, 2021, 생능출판

다음 차시 예고

- 억지기법(brute-force) 전략을 활용한 문제들에 대해 학습한다.