Greedy Algorithm #1

Dosung(Dave) Kim, Ph.D.

Table of Contents

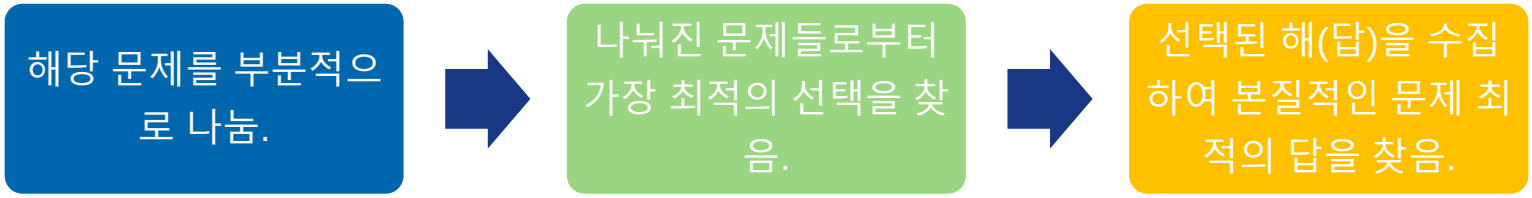
Greedy Algorithm이란?

- **Greedy Algorithm**
  - 개요
    - 최적의 해를 구하는 알고리즘으로 매 선택마다 최적의 답을 선택해서 가장 적합한 결과를 도출하는 알고리즘
    - 100% 정확한 답을 도출하는 것이 아님. 어떤 문제에 대해서 가장 적합한 답을 도출 및 참고하여 효율적인 방안을 찾는데 유용한 알고리즘임.
    - 흔히들 Greedy 알고리즘을 욕심쟁이 알고리즘 또는 탐욕적 알고리즈이라고 함.
  - 특징
    - 현재순간의 선택이 다음 단계 또는 미래에 영향을 주지 않음(탐욕적 선택).
    - 가장 큰 순서대로 아니면 가장 작은 순서대로 그 기준을 제시를 해줌. 따라서 정렬(Sort)알고리즘과 같이 적용시키는 경향이 있음.
    - 최적의 부분적인 구조를 가지는 알고리즘임. 즉 부분적인 문제의 최적해를 구해 가장 적합한 해를 구하는 구조임.

장점	단점
처리속도가 다른 알고리즘과 비교했을 때 빠름.	최적의 답을 선택해서 산출한다 하더라도 항상 보장되는 것은 아님.

Greedy Algorithm의 원리(1/3)

- **Greedy Algorithm**
  - 절차
    - 다음과 같은 절차로 진행을 할 수 있음.



    - 예시
      - Greedy 알고리즘의 대표적인 예시로 “거스름돈 문제 " 가 있음. 예를 들어 500원, 100원, 50원 및 10원 등 여러 개의 동전 종류가 있다고 가정함. 고객에게 지불에 대한 거스름돈 지불을 위해 N원을 지불해야 하는 상황이 발생 함. N값은 5,300원이라고 가정함. 이 경우, 최소한의 동전은 몇 개가 필요한가?
      - 부분적인 부분에서 가장 큰 단위 인 500원을 지불한다면 10개가 필요함. 그 다음 큰 단위인 100원으로 3개가 필요함. 따라서 최소한의 동전개수는 13개가 필요함.

Greedy Algorithm의 원리(2/3)

- **Greedy Algorithm**
  - 예시(거스름 돈 파이썬 예시코드)
    - 아래와 같이 N값을 찾기 위한 코드를 간단하게 작성할 수 있음.
    - For문을 보면 반복적으로 최소값이 되도록 계속 카운팅을 함. 결과값은 13

```
4   #N값 설정
5   n = 5300
6   #개수변수 설정
7   lnCount = 0
8   #몫설정
9   lnQuotient = 0
10
11  #동전종류 리스트 설정
12  lsCoin_type = [500, 100, 50, 10]
13
14  #동전종류별로 N값이 됨
15  for coin in lsCoin_type:
16      lnQuotient = n // coin
17      lnCount += lnQuotient
18      n %= coin
19
20  print(lnCount)
```

Greedy Algorithm의 원리(3/3)

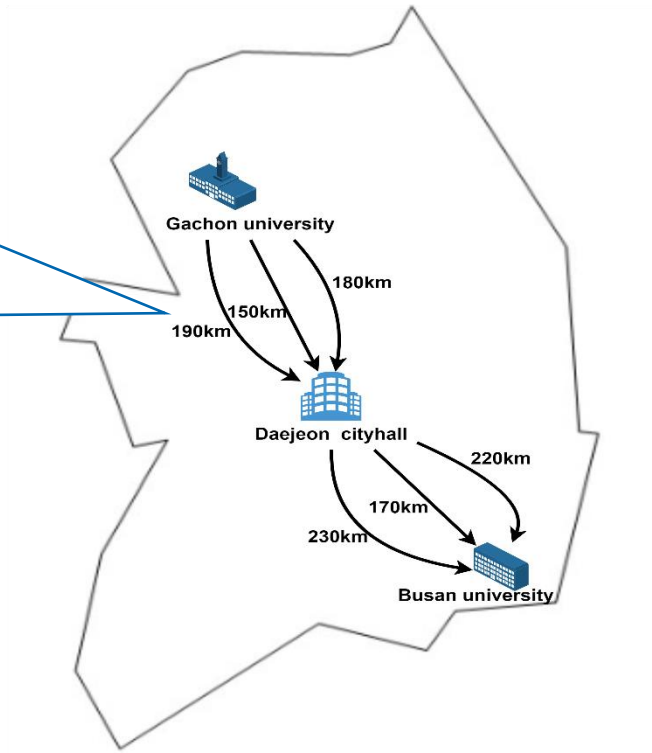
- **Greedy Algorithm**
  - 예시(추를 이용한 최소 무게 측정 파이썬 예시코드)
    - 초기 입력 추의 리스트는 6개로써 (2, 1, 6, 7, 29, 4) 임.
    - 아래 로직을 보면 추를 이용해서 가장 최소의 무게를 측정하는 알고리즘을 구현한 것임. 해당 리스트를 오름차순으로 정렬 하고 리스트를 반복적으로 돌면서 최소의 무게합의 값을 찾아내는 로직 임. 결과: 21

```
4       lnValue = 1
5
6       lsWeights = [2, 1, 6, 7, 29, 4]
7       lsWeights.sort()
8
9       for weight in lsWeights:
10          if lnValue < weight:
11              break
12          else:
13              lnValue += weight
14
15      print(lnValue)
```

테스트 예제(1/2)

- **Greedy Algorithm**
  - 테스트 예제 1
    - Greedy 알고리즘을 적용하여 출발지로부터 목적지까지 최적의 동선을 파악하고 총 최소거리를 산출하는 알고리즘을 산출해야 함.
    - 아래 그림과 같이 경기도 성남 가천대학교 정문에서 출발하여 부산까지 가려고 할 때 아래와 같은 경로가 있다고 가정 함. 가장 최적의 경로를 기반으로 최소거리를 산출하는 알고리즘을 작성하기 바람. 경로는 가천대학교 → 대전시청 → 부산대학교로 가야 함.

- 입력조건
  - 첫번째는 Gachon university ←→ Daejeon cityhall간의 여러 경로에 해당하는 리스트 임. Ex) [190, 150, 180]
  - 두번쨰는 Daejeon cityhall ←→ Busan cityhall간의 여러 경로에 해당하는 리스트임. Ex) [230, 170, 220]
  - Min함수를 사용하면 안됨.
- 출력조건
  - 첫번째의 최단거리를 출력
  - 두번째의 최단거리를 출력
- 출력 예시
  - 첫번째 경로와 두번째 경로의 최단거리를 합한 수가 출력되어야 함.
    Ex) 330km



테스트 예제(2/2)

- **Greedy Algorithm**
  - 테스트 예제 2
    - Greedy 알고리즘을 적용하여 최적의 수강신청 스케줄을 알아내는 알고리즘을 작성해야 함.
    - 가천대학교 홍길동 학생이 KEA 강의를 수강하기 위해 강의 스케줄을 체크 및 조율하고 있음. 10개의 과목들 중 최대한 많이 들으려고 조율하고 있음. 따라서 강의시간이 서로 겹치면 안되기 때문에 최적의 강의시간을 산출해야 함. 입력조건은 튜플형태로 입력됨.-
    - 해당 학기의 수업 리스트는 아래와 같음.

      [(4, 8), (3, 5), (2, 4), (1, 4), (8, 10), (6, 8), (5, 7), (4, 5), (5, 8), (9, 11)]

    - 상위 수업리스트의 형태에서 각 튜플의 0번째는 강의시작교시를 의미하고 1번째는 종료교시를 의미함. 예를 들어 첫번째 튜플 인덱스 인 (4, 8)의 강의를 보면 4교시에 시작하고 종료는 8교시에 함.
    - 또한 (8, 10)강의와 (6, 8)강의를 보면 시작교시와 종료교시가 겹침. 이럴 경우에는 두 강의수강은 불가능한 것으로 간주함.

    - 입력조건: 튜플형식의 [(4, 8), (3, 5), (2, 4), (1, 4), (8, 10), (6, 8), (5, 7), (4, 5), (5, 8), (9, 11)]
    - 출력조건: 겹치지 않고 최적의 강의시간을 산출해야 함.
    - 출력 예시: [(2, 4), (5, 7), (8, 10)]