2021 KCA Algorithm Study

Day 2: Greedy Algorithm

KU201711424

KCA Mentor **Pt J**

목차

Greedy Algorithm이란?

- ✔ 매 순간 Solution 후보 중 최선의 것을 찾아 최적화
- ✔ 한 번 선택한 것은 그 선택은 번복하지 않음
- ✔ 최선의 Solution을 확신할 수 없지만 좋은 Solution 제공

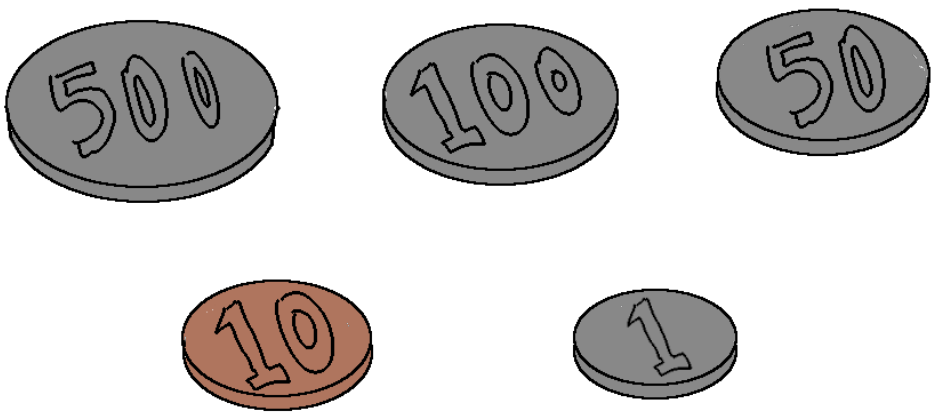
- ✔ Greedy [탐욕적인]
  data 간의 관계를 고려하지 않고 그 순간의 최선을
  "욕심내어" 선택하는 근시안적인 선택

예시

첫번째 예시: 동전 거스름돈

Q. 가장 적은 수의 동전을 사용하여 거스름돈 주는 방법 찾기

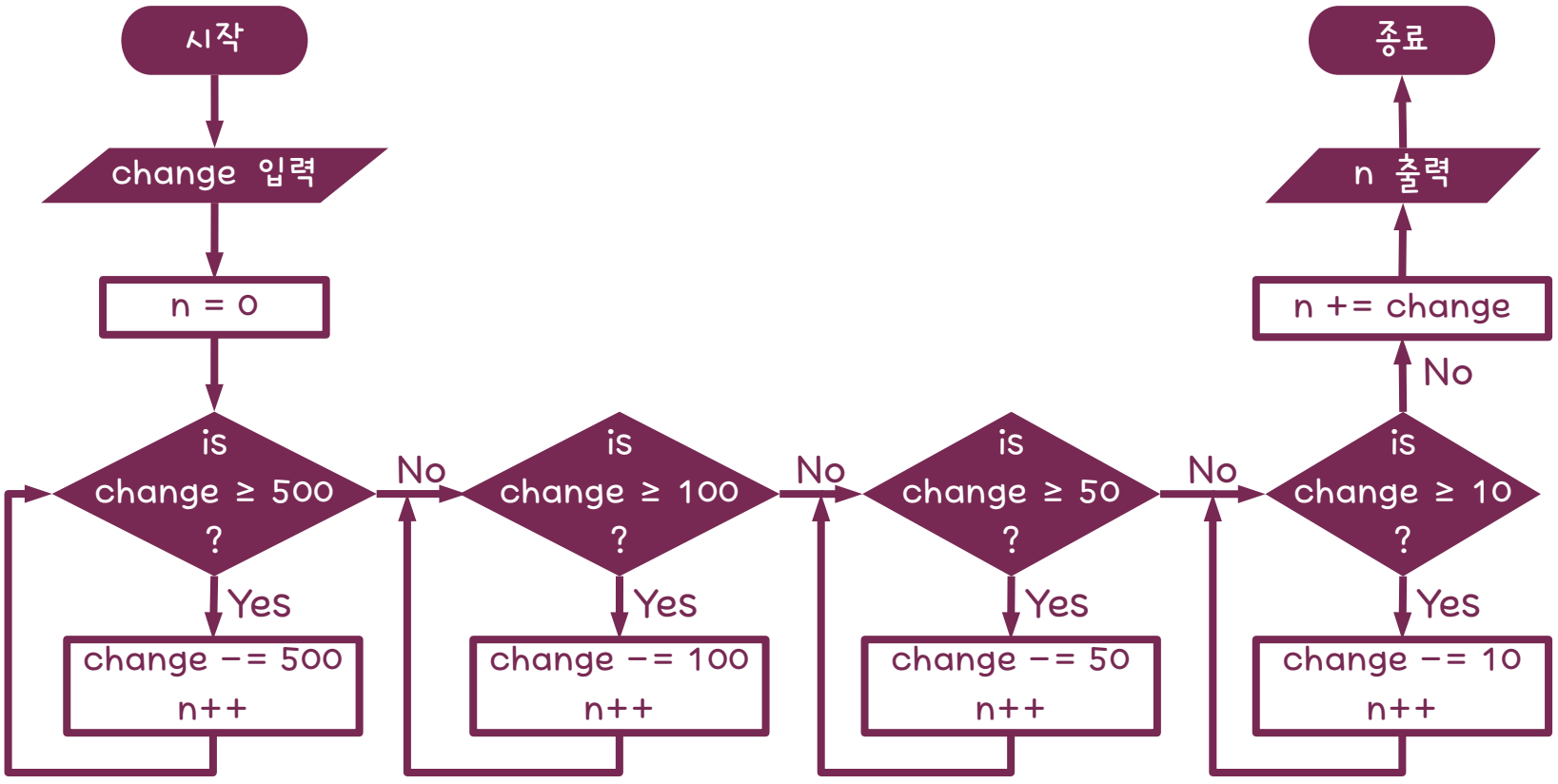
- ✔ 가장 큰 동전부터 선택한다.
- ✔ 해당 동전을 더 이상 선택할 수 없게 되면
  다음 단위의 동전을 선택한다.



예시

첫번째 예시: 동전 거스름돈

Q. 가장 적은 수의 동전을 사용하여 거스름돈 주는 방법 찾기



예시

첫번째 예시: 동전 거스름돈

Q. 가장 적은 수의 동전을 사용하여 거스름돈 주는 방법 찾기

✔ Greedy Algorithm으로 최선의 답이 보장되는 문제

✔ 만약 가상의 160원 동전이 존재한다면?

최선의 답이 보장되지 않는 경우?

✔ 각 선택이 이후 선택의 quality에 영향을 주는 경우

"선택한 unit은 최종 해의 quality와 관련하여
미결정 unit에 종속적"인 경우

✔ 두 가지 결정이 종속적이면 그 조합을 평가한 후 선택

⇒ Greedy하지 않은 방식

예시

두번째 예시: 부분 배낭 문제

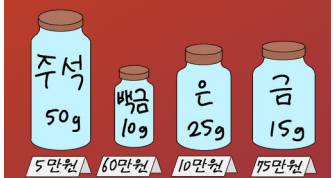
Q. 가져갈 수 있는 물건들 찾기

무게와 가치를 가진 N 개의 물건이 있다.
배낭에는 무게 제한이 있다.

가치의 총합이 최대가 되도록 배낭에
물건을 넣는다고 할 때
가져갈 수 있는 물건의 목록을 구하라!

✔ "배낭 문제"는 물건을 부분적으로 가져갈 수 없지만
"부분 배낭 문제"는 물건을 쪼개 부분적으로 가져갈 수 있다

예시



두번째 예시: 부분 배낭 문제

Q. 가져갈 수 있는 물건들 찾기

✔ "단위 무게 당 가치"를 구하여 내림차순 정렬

물건	단위 그램 당 가치
백금	60000원
금	50000원
은	4000원
주석	1000원

✔ 가방에 40g까지 넣을 수 있다면?

예시

세번째 예시: 허프만 코딩

- ✔ 텍스트 압축에 사용되는 방식 중 하나
- ✔ “일련의 문자들”로 이루어진 텍스트를 “일련의 bit들”로 변환

- ✔ 압축하지 않은 N개의 문자는 ASCII코드로 작성되었을 때
  ASCII 문자 하나가 8bit 이므로 8*N bit 공간 차지

예시

세번째 예시: 허프만 코딩

- ✔ 다른 문자들에 비해 높은 빈도로 사용되는 문자에는 짧은 bit code 할당
- ✔ 다른 문자들에 비해 낮은 빈도로 사용되는 문자에는 긴 bit code 할당
- ✔ 접두어 속성 (Prefix Property)
  문자와 문자 사이에 구분을 위한 문자나 기호를 넣어주지 않아도 어떤 bit sequence가 발견된다면 그것은 곧바로 해당 문자로 변환할 수 있다는 특성

예시

세번째 예시: 허프만 코딩

Q. 접두어 속성을 가진 bit code 생성하기

✔ binary tree를 사용한다.

이진 트리 (Binary Tree)
차수가 최대 2인 트리

트리 (Tree)
계층적인 구조를 나타내는 1:N 관계의 자료구조
상위 원소에서 하위 원소로 내려가며 확장

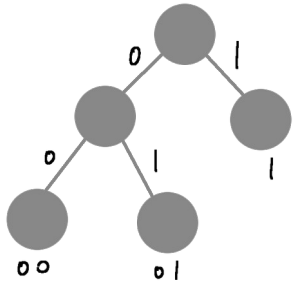
노드 (node): 트리의 구성 요소
간선 (edge): 부모 노드와 자식 노드를 연결하는 선
루트 (root): 트리의 맨 위에 있는 , 부모가 없는 노드
단말 노드 (leaf node): 트리의 맨 아래 있는 , 자식이 없는 노드
차수 (degree): 노드가 가지고 있는 자식 노드의 개수

예시

세번째 예시: 허프만 코딩

Q. 접두어 속성을 가진 bit code 생성하기

- ✔ binary tree를 사용한다.
- ✔ 각각의 edge에 0 또는 1 할당
- ✔ 문자는 leaf node에 존재
- ✔ root node부터 문자까지 도달하는 경로 상의 bit sequence



예시

세번째 예시: 허프만 코딩

Q. 접두어 속성을 가진 bit code 생성하기

✔ 예를 들어... 다음과 같은 빈도의 4개의 문자가 사용된다고 가정

✔ A: 450
✔ T: 90
✔ G: 120
✔ C: 270

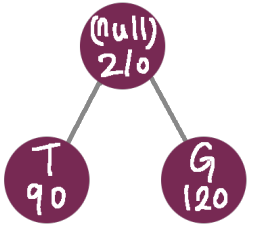
✔ 집합 Q에 {문자, 빈도수}로 이루어진 노드들을 넣는다.

예시

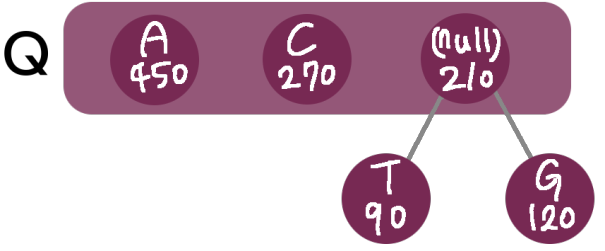
세번째 예시: 허프만 코딩

Q. 접두어 속성을 가진 bit code 생성하기

✔ 빈도수가 가장 적은 두 node를 꺼내
새 부모 node의 자식으로 묶고
부모 node의 빈도수는 두 자식의 빈도수 합으로 한다



✔ 부모 node를 집합 Q에 집어 넣는다

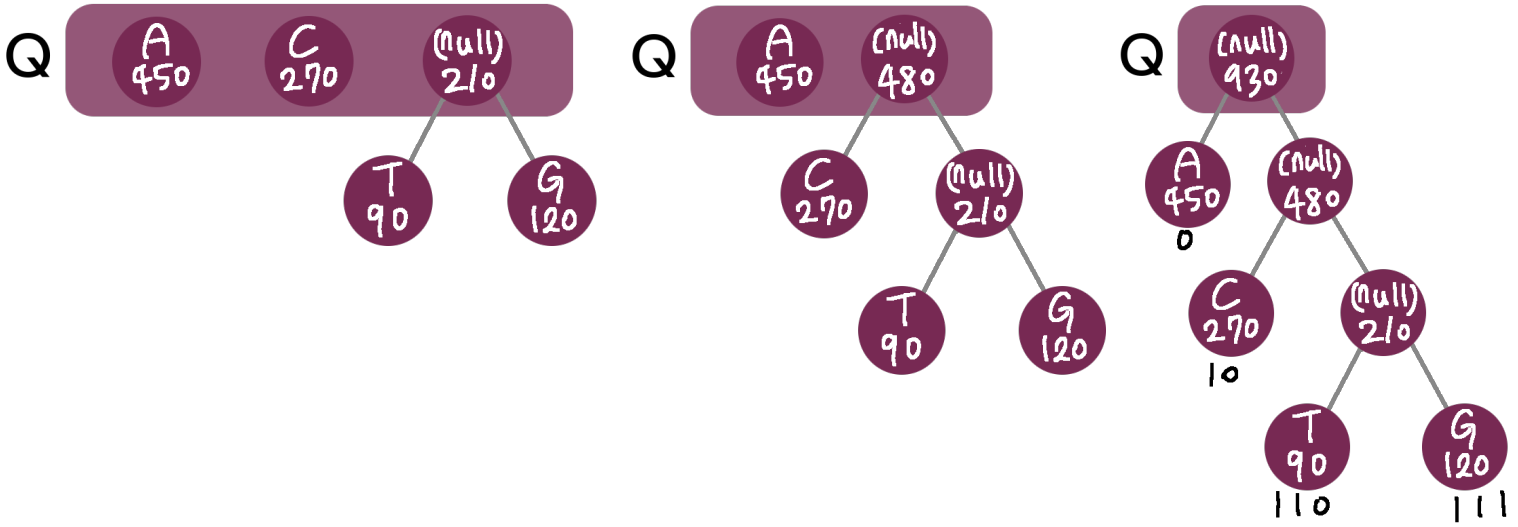


예시

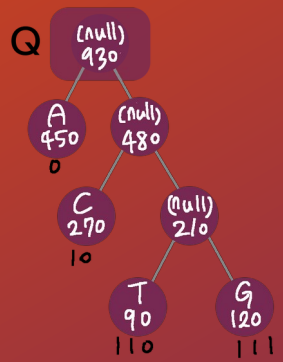
세번째 예시: 허프만 코딩

Q. 접두어 속성을 가진 bit code 생성하기

✔ 집합 Q의 원소 수가 1이 될 때까지 반복



예시



세번째 예시: 허프만 코딩

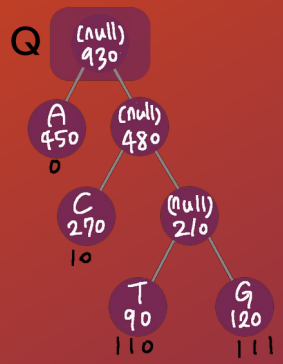
Q. 접두어 속성을 가진 bit code 생성하기

✔ 압축할 텍스트 : G T T A C G A G A T

⇒

✔ 압축을 풀 때도 (접두사 속성에 의해) Greedy하게!

예시



세번째 예시: 허프만 코딩

Q. 접두어 속성을 가진 bit code 생성하기

✔ ASCII 코드로 작성했을 때 bit 수

(450 + 90 + 120 + 270) * 8 = 7440

✔ 허프만 코딩으로 압축했을 때 bit 수

450 * 1 + 270 * 2 + 90 * 3 + 120 * 3 = 1620

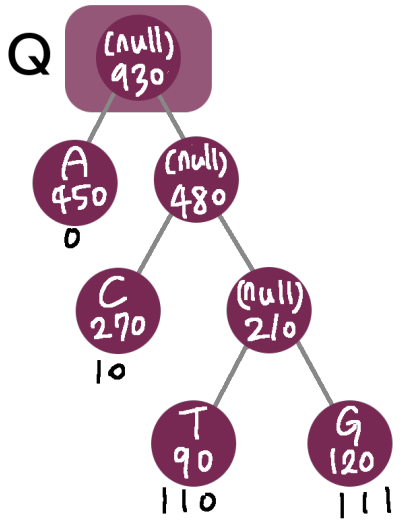
✔ 압축률 = 1620 / 7440 * 100 ≒ 21.774%

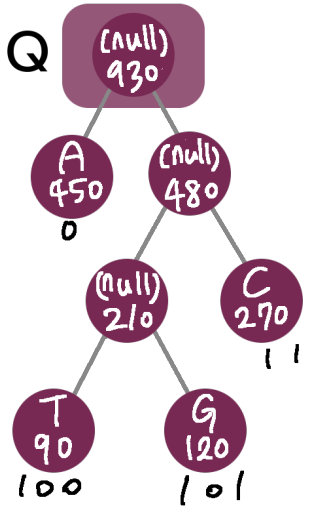
예시

세번째 예시: 허프만 코딩

Q. 접두어 속성을 가진 bit code 생성하기

- ✔ 새 node를 집합 Q의 어디에 넣냐에 따라 서로 다른 bit code가 나올 수 있다
- ✔ BUT! 빈도수에 따른 bit수는 다르지 않다





정리

- ✔ 각 Step에서 그 순간의 최선의 선택을 하는 방식
- ✔ 쉬운 방법으로 좋은 근사값을 얻을 수 있지만
  정답을 찾지 못하는 경우도 있을 수 있다

다음 시간에는

다양한 알고리즘 (2) Divide and Conquer

에 대해 배워보도록 하겠습니다★