탐욕법 (그리디 알고리즘)

190507(화) - 전현승

경북대학교 알고리즘 문제해결 동아리 GORI

목차

- 탐욕법 (그리디 알고리즘)
- 그리디의 조건
- 그리디의 응용
- 5585 - 거스름돈
- 11399 - ATM
- 1931 - 회의실배정

탐욕법 (그리디 알고리즘)

- "매 선택에서 **지금 이 순간 당장 최적인 답을 선택**하여 적합한 결과를 도출하자"
- 각 단계마다 최선의 선택을 한다 → 전체 문제에 대한 최적해가 된다
- 모든 선택을 고려하지 않는다. 그때그때 가장 좋아 보이는 선택을 하면 답이다.
- 지금 선택이 다음 선택에 영향을 미치면 안 된다.

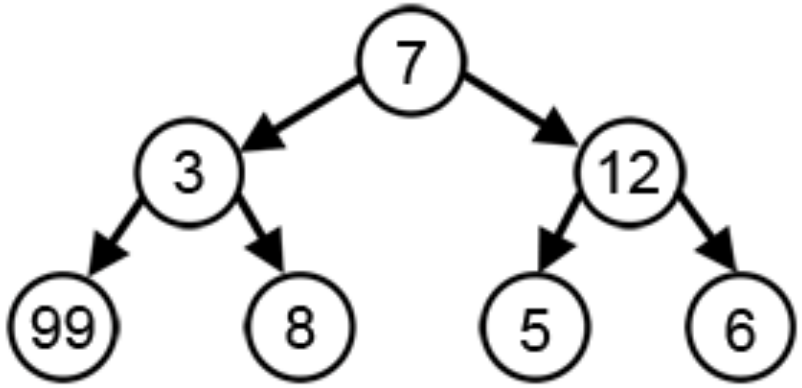
- 다 같은 말

그리디의 조건

- 모든 문제에서 탐욕법을 쓰면 당연히 안 됨
- == 단계별 최선의 선택이 전체 문제의 최적해가 되지 않는 경우

- 예) 길 위에 돈이 떨어져 있다. 길을 가면서 돈을 가장 많이 줍는 방법은?
  - 아무런 조건이 없다면, 그냥 있는 대로 주우면 된다
  - 각 단계마다 최선의 선택을 하면 전체 문제의 최적해가 된다 → 그리디
- 만약 "연속해서 3번 이상 못 줍는다"라면?
  - 무턱대고 계속 주우면 안 된다 (큰 걸 못 주울 수도 있다)
  - 현재 선택이 다음 선택에 영향을 미친다 → 그리디 X

그리디의 조건



- 예) 루트에서 시작해서 합이 가장 큰 경로는?
  - 그때그때 둘 중 더 큰 자식 쪽으로 골라서 가면?
  - 7 → 12 → 6
  - 7 → 3 → 99 쪽으로 가는 게 합이 더 크다.
  - 각 단계마다 최선의 선택을 해도, 그게 전체 문제의 최적해가 아닐 수 있다.
  - → 그리디 X

그리디의 조건

- Formal:
- 1. Greedy choice property : 현재 선택이 미래 선택에 영향 X
- 2. Optimal substructure : 부분문제 최적해 → 전체문제 최적해

- 2번은 DP에서 한 번 봤다
- 대부분 그리디 문제는 DP로도 풀 수 있다

그리디의 응용

- Kruskal 알고리즘
- 다익스트라 알고리즘
- Fractional Knapsack 문제
- 등등
- 다양한 분야에서 응용

5585 - 거스름돈

- (1000엔 이하로) 물건을 사고 1000엔 한 장을 낸다
- 잔돈으로 500엔, 100엔, 50엔, 10엔, 5엔, 1엔이 충분히 있다
- 동전 개수 가장 적게 거슬러 주는 방법?

- 그냥 제일 비싼 거부터 거슬러 주면?
- 757엔을 거슬러 주려면
  - → 500엔 1개, 100엔 2개, 50엔 1개, 5엔 1개, 1엔 2개
- 된다

5585 - 거스름돈

ALGORITHM 6 Greedy Change-Making Algorithm.

procedure *change*($c_1, c_2, \ldots, c_r$: values of denominations of coins, where
$c_1 > c_2 > \cdots > c_r$; n: a positive integer)
for $i := 1$ **to** r
    $d_i := 0$ {d_i counts the coins of denomination c_i used}
    while $n \geq c_i$
        $d_i := d_i + 1$ {add a coin of denomination c_i}
        $n := n - c_i$
{d_i is the number of coins of denomination c_i in the change for $i = 1, 2, \ldots, r$}

- 동전 액수끼리 배수 관계가 성립하는 경우 증명이 되어 있다

5585 – 거스름돈

```
int n; cin >> n;
n = 1000 – n;  // 거슬러줄 돈

int coin[6] = {500, 100, 50, 10, 5, 1};  // 큰 액수부터 거슬러주자
int total = 0;  // 사용한 동전 개수
for (int i = 0; i < 6; i++) {
        total += n / coin[i];
        n %= coin[i];
}

cout << total << ‘\n’;
```

11399 – ATM

- i번 사람이 돈을 인출하는데 걸리는 시간 : P(i)
- N명 줄세웠을 때, 각 사람이 돈을 인출하는데 필요한 시간의 합 최소화
- ATM은 1대밖에 없다 → 앞사람 기다리는 시간 포함

- 예) 총 5명, P(1) = 3, P(2) = 1, P(3) = 4, P(4) = 3, P(5) = 2
  - 줄을 [1, 2, 3, 4, 5] 순서대로 서면
  - 1번 사람 뽑는 시간 : P(1) = 3분
  - 2번 사람 뽑는 시간 : P(1) + P(2) = 3 + 1 = 4분
  - 3번 사람 뽑는 시간 : P(1) + P(2) + P(3) = 3 + 1 + 4 = 8분
  - …

11399 – ATM

- 앞에 있는 사람이 걸리는 시간만큼 누적된다
- 인출하는데 걸리는 시간이 제일 적은 사람부터 인출하면 된다

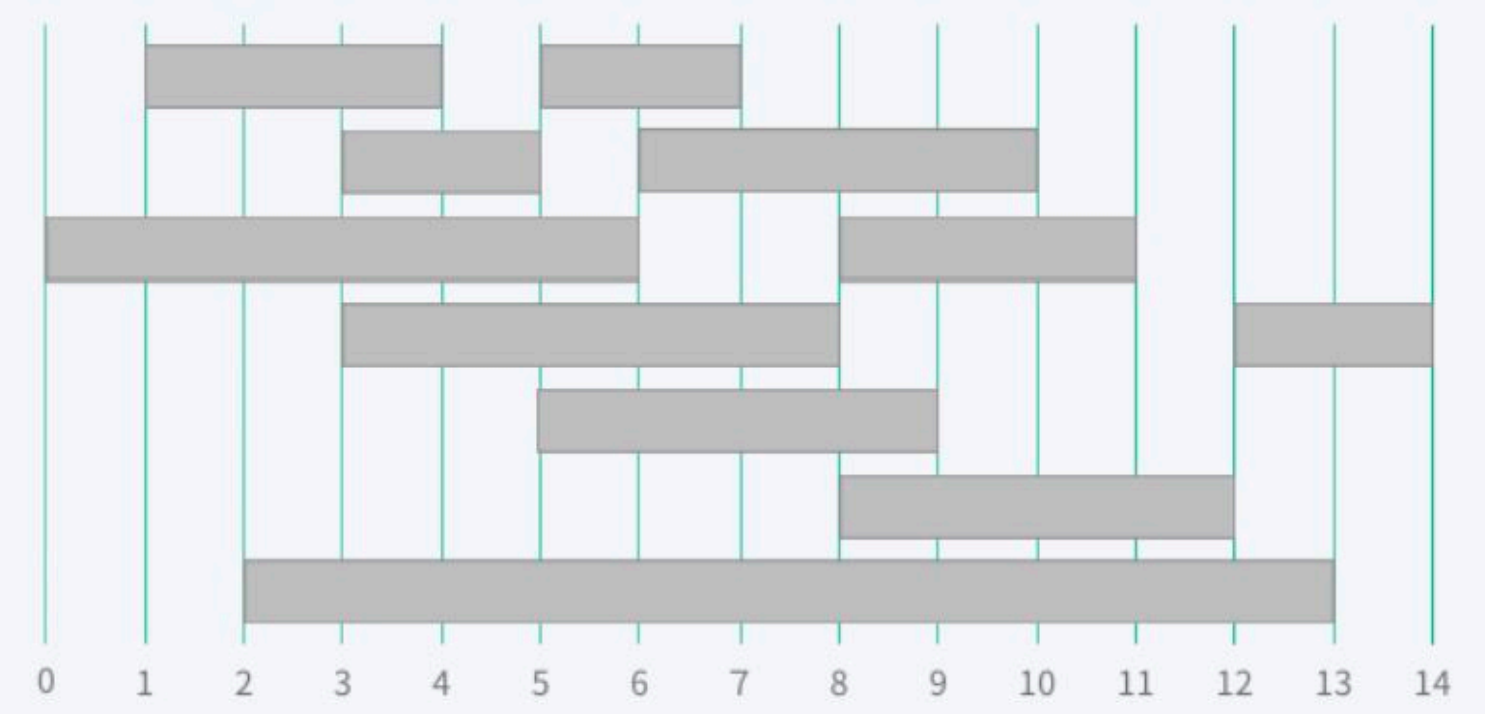
- N명 줄을 [a, b, c, d, …] 순서로 세웠다고 하자
  - a 뽑는 시간 : P(a)
  - b 뽑는 시간 : P(a) + P(b)
  - c 뽑는 시간 : P(a) + P(b) + P(c)
  - …
- 총 합
  - = P(a) + (P(a) + P(b)) + (P(a) + P(b) + P(c)) + …
  - = N * P(a) + (N – 1) * P(b) + (N – 2) * P(c) + …

11399 – ATM

```
int time[1001], total = 0, sub = 0;
int n; cin >> n;
for (int i = 0; i < n; i++) {
        cin >> time[i];
}

sort(time, time + n);  // 인출 시간이 짧은 순으로 정렬

for (int i = 0; i < n; i++) {
        sub += time[i];
        total += sub;  // 누적합 구하기
}

cout << total << '\n';
```

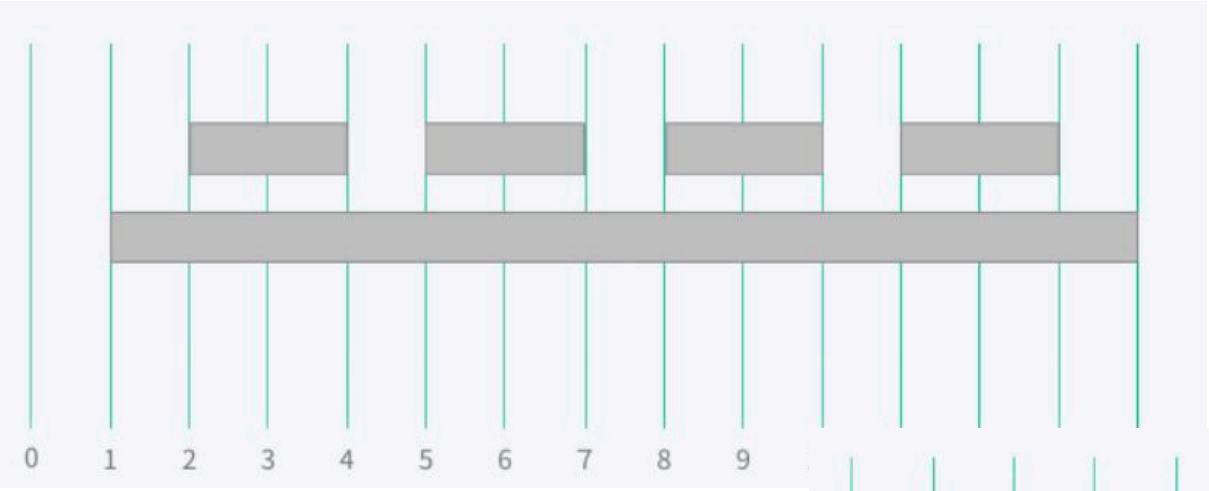
1931 – 회의실배정

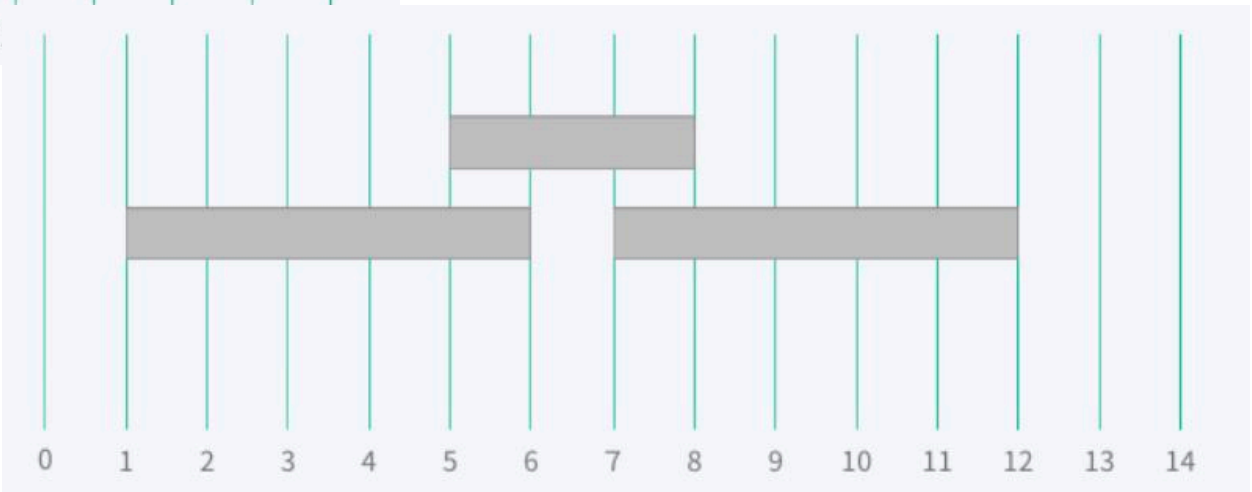
- 회의실 1개, 회의 N개
- 회의가 겹치지 않게 하면서 최대한 많은 회의 배정



1931 – 회의실배정

- 일찍 시작하는 것부터? 짧은 것부터?





1931 – 회의실배정

ALGORITHM 7 Greedy Algorithm for Scheduling Talks.

procedure *schedule*($s_1 \leq s_2 \leq \cdots \leq s_n$: start times of talks,
$e_1 \leq e_2 \leq \cdots \leq e_n$: ending times of talks)
sort talks by finish time and reorder so that $e_1 \leq e_2 \leq \ldots \leq e_n$
$S := \emptyset$
for $j := 1$ **to** n
    if talk j is compatible with S **then**
        $S := S \cup \{\text{talk } j\}$
return S {S is the set of talks scheduled}

- https://en.wikipedia.org/wiki/Interval_scheduling#Greedy_polynomial_solution

1931 – 회의실배정

- http://boj.kr/008e88e2d16148198469e992d9e62be6

참고

- https://en.wikipedia.org/wiki/Greedy_algorithm
- https://kim6394.tistory.com/67 (회의실 배정 그림)

끝