#09-2. 그리디

나정휘
https://justiceHui.github.io/

그리디

그리디

- Greedy: [형용사] 탐욕스러운
- 현재 상태에서 가장 좋은 선택을 하는 방법
  - 현재 상태에서 가장 좋은 선택이 전체 범위에서도 가장 좋은 선택이라는 것은 보장 못함
  - 만약 이 전략이 전체 범위에서도 최적이라면, 그리디 기법으로 문제를 해결할 수 있음

- 동적 계획법 vs 그리디 기법
  - 동적 계획법: D[i]를 계산할 때 $1 \le j < i$인 모든 D[j]를 확인
  - 그리디 기법: D[i]를 계산할 때 적당한 D[j] 하나만 확인

그리디

그리디

- 많이 보이는 형태
  - n개의 원소가 주어짐
  - 특정 조건을 만족하도록 원소 몇 개 선택해서
  - 원소의 가중치 합을 최대/최소화하는 문제

```
A[N] = 원소들
Res[] = 정답 집합

sort(A, A+N); // A를 특정 기준으로 정렬
for(int i=0; i<N; i++){
    if(Feasible(Res + A[i])){ // 정답에 A[i]를 추가해도 조건을 만족하면
        Res += A[i];
    }
}
```

그리디 예시 - 1

BOJ 11047 동전 0

- N가지의 동전을 사용해서 K원을 만들어야 함
- 사용하는 동전의 개수를 최소화
- 동전은 서로 약수/배수 관계, 1원 짜리 동전 항상 존재

- 가격이 큰 동전부터 최대한 많이 사용하는 전략
  - 증명?

그리디 예시 - 1

BOJ 11047 동전 0

- 그리디 기법으로 구한 답보다 더 좋은 해가 없다는 것을 증명
  - 가격 내림차순으로 동전을 정렬하자 : C[i] = i번째로 비싼 동전의 금액
  - 그리디에서 각 동전을 사용한 개수를 저장한 리스트 A
  - 실제 최적해에서 각 동전을 사용한 개수를 저장한 리스트 B

  - (귀류법) A보다 더 적은 동전을 사용한 최적해 B가 존재한다고 가정
    - A와 B가 처음으로 다른 지점 i가 존재
    - 비싼 동전부터 최대한 많이 가져가는 전략에 의해 A[i] > B[i]
    - C[i] * (A[i] - B[i])원 만큼의 차이를 채우기 위해 C[i]보다 싼 동전을 더 사용할 텐데
    - C[i]의 약수면서 싼 동전으로 C[i] * (A[i] - B[i])원을 만드는데 A[i]-B[i]보다 많은 동전이 필요하므로
    - B는 A보다 동전을 더 적게 사용할 수 없다.

질문?

그리디 예시 - 2

Fractional Knapsack Problem

- N개의 물건이 있고, i번째 물건의 무게는 W_i, 가격은 P_i원
- 무게의 합이 Xkg 이하가 되도록 물건을 적당히 선택할 때
- 가능한 가격의 합의 최댓값을 구하는 문제
- 단, 물건을 원하는 대로 분할할 수 있음
  - (W_i, P_i)를 무게가 x, W_i-x인 물건으로 분할하면
  - $(x, P_i/W_i*x)$, $(W_i-x, P_i/W_i*(W_i-x))$로 분할됨

- 무게 대비 가격(효율, P_i/W_i)이 높은 물건부터 가져가는 전략

그리디 예시 - 2

Fractional Knapsack Problem

- 그리디 기법으로 구한 답보다 더 좋은 해가 없다는 것을 증명
  - 가격이 0 이하인 물건은 신경 쓰지 않아도 됨
  - 무게의 합이 X 이하인 경우, 모두 가져가는 것이 최적
    - 그리디 알고리즘은 이 경우를 잘 처리함
  - 그렇지 않은 경우, 무게의 합이 X가 되도록 가져가는 것이 최적
    - 그리디 알고리즘은 이 경우 무게의 합이 X가 됨
    - 그리디 알고리즘이 가격을 최대화하는지 증명하자.

그리디 예시 – 2

Fraction Knapsack Problem

- 증명 (cont.)
  - 물건을 효율에 대한 내림차순으로 정렬하자. 편의상 효율이 모두 다르다고 가정한다.
  - 그리디에서 각 물건을 가져간 무게를 저장한 리스트 A, 실제 최적해의 리스트 B

  - (귀류법) A보다 가격이 더 비싼 **최적해** B가 존재한다고 가정
    - A와 B가 처음으로 달라지는 지점 i가 존재
    - 효율이 높은 것부터 최대한 많이 가져가는 그리디 알고리즘에 의해 A[i] > B[i]
    - B는 최적해이기 때문에 A[j] < B[j]인 j(> i)가 존재

    - 새로운 해 B'를 구성한다. B'[i] = B[i] + eps, B'[j] = B[j] – eps이고, 다른 모든 B'[k] = B[k]이다.
    - B와 B'의 무게는 동일하지만 가격은 B'가 더 비싸다.
    - B가 최적해라는 가정에 모순이 생겼으므로 A보다 더 비싼 최적해는 존재하지 않는다.

질문?

그리디 예시 – 3

BOJ 11399 ATM

- i번째 프로세스는 CPU를 P_i 시간 동안 점유함
- 대기 시간의 합을 최소로 하는 스케줄링을 찾는 문제

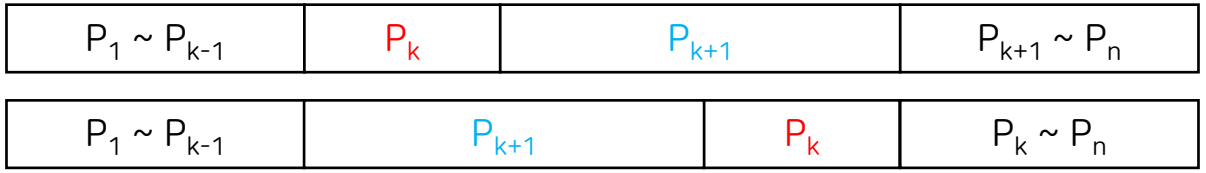
- SJF(Shortest Job First) Scheduling
  - 평균 대기 시간을 최소화한다고 알려져 있음
  - 평균 대기 시간 최소 ⇒ 대기 시간 합 최소

- 소요 시간이 짧은 프로세스부터 처리하면 됨

그리디 예시 – 3

BOJ 11399 ATM

- P_i가 작은 프로세스부터 처리하는 것이 최적임을 증명
  - 두 프로세스 P_k, P_{k+1} 중 먼저 처리해야 하는 것을 결정하자.
  - 인접한 원소의 순서만 잘 결정하면, 버블 정렬 느낌으로 전체 원소를 정렬할 수 있음



  - P_k와 P_{k+1}의 대기 시간만 보면 됨
  - 위 : $\text{sum}(P_1 \sim P_{k-1}) + \text{sum}(P_1 \sim P_{k-1}) + P_k$
  - 아래 : $\text{sum}(P_1 \sim P_{k-1}) + \text{sum}(P_1 \sim P_{k-1}) + P_{k+1}$
  - 동류항 없애면 각각 P_k, P_{k+1}만 남음
  - 그러므로 $P_k < P_{k+1}$이면 P_k가 먼저 오도록 정렬하는 것이 이득

질문?

그리디 예시 - 4

BOJ 1931 회의실 배정

- 한 개의 회의실과 N개의 회의가 있음
- 각 회의는 시작 시간과 종료 시간이 있어서, 해당 기간 동안 회의실을 점유함
- 몇 개의 회의를 선택해서 서로 겹치지 않도록 회의를 진행할 때
- 진행 가능한 회의의 최대 개수를 구하는 문제

- 끝나는 시간이 빠른 회의부터 선택하는 그리디가 성립함

그리디 예시 - 4

BOJ 1951 회의실 배정

- 종료 시간이 가장 빠른 회의를 포함하는 최적해가 있다는 것을 증명
  - 종료 시간이 가장 빠른 회의 m을 포함하지 않는 최적해 O가 존재한다고 하자.
  - 당연히 O에는 겹치는 회의가 존재하지 않는다.
  - O에서 종료 시간이 가장 빠른 회의 x를 제거하고 m을 추가한 O'를 생각해보자.
  - O와 O'의 크기는 동일하고, O'에는 겹치는 회의가 존재하지 않는다.
  - 그러므로 모든 최적해는 m을 포함하도록 바꿀 수 있다.

  - 종료 시간이 가장 빠른 회의를 선택한 뒤
  - 그 회의와 겹치는 회의를 모두 제거하고
  - 다시 종료 시간이 가장 빠른 회의를 선택하는 방식

질문?

정리

지금까지 본 증명 방법

- A보다 좋은 최적해 B가 없음을 보임
  - ex. 동전 0
- A보다 좋은 최적해 B가 있다고 가정한 뒤, B가 최적해가 아님을 보임
  - ex. Fractional Knapsack Problem
- 원소의 우선순위를 정한 뒤, 우선순위가 높은 것부터 선택하는 것이 최적임을 보임
  - ex. ATM (SJF Scheduling)
- 어떤 원소를 포함하는 최적해가 항상 존재함을 보임
  - ex. 회의실 배정

Exchange Argument

Exchange Argument

- BOJ 11399 ATM의 증명 방법을 일반화한 것

- 인접한 두 원소의 순서를 결정할 수 있으면
- 버블 정렬을 이용해 전체 원소의 순서를 결정할 수 있음

- 버블 정렬과 다른 일반적인 비교 기반 정렬의 결과물을 동일하므로
  - ex. quick sort, heap sort, merge sort, ...
- 비교 함수를 잘 작성한 뒤 std::sort를 사용하면 됨

그리디 예시 - 5

BOJ 14908 구두 수선공

- i번째 작업을 수행하는데 T_i일 걸림
- i번째 작업의 완료가 하루 지연될 때마다 S_i원 벌금 내야 함
- 벌금을 최소로 하는 작업 순서를 정하는 문제

그리디 예시 - 5

BOJ 14908 구두 수선공

- $\text{sum}(T_{1..k-1}) * S_k + (\text{sum}(T_{1..k-1}) + T_k) * S_{k+1}$
- $\text{sum}(T_{1..k-1}) * S_{k+1} + (\text{sum}(T_{1..k-1}) + T_{k+1}) * S_k$

- $T_k * S_{k+1} \le T_{k+1} * S_k$가 되도록 정렬해야 함
- $T_k / S_k \le T_{k+1} / S_{k+1}$이 되도록 정렬해야 함
- T_i / S_i 오름차순 정렬

질문?

그리디 예시 - 6

BOJ 2180 소방서의 고민

- 일차함수 $f_i(x) = a_i x + b_i$가 여러 개 주어짐 ($a_i, b_i > 0$)
- $x = 0$에서 시작해서 $x = x + f_i(x)$를 한 번씩 적용할 때
- 최종 결과의 최솟값을 구하는 문제

그리디 예시 - 6

BOJ 2180 소방서의 고민

- $(a_{k+1}+1)\ ((a_k+1)x + b_k) + b_{k+1}$
- $(a_k+1)\ ((a_{k+1}+1)x + b_{k+1}) + b_k$

x	$a_kx + b_k$	$a_{k+1}x + b_{k+1}$	f

x	$a_{k+1}x + b_{k+1}$	$a_kx + b_k$	f

- $a_{k+1}x + a_ka_{k+1}x + a_{k+1}b_k + x + a_kx + b_k + b_{k+1}$
- $a_kx + a_ka_{k+1}x + a_kb_{k+1} + x + a_{k+1}x + b_{k+1} + b_k$

- $a_{k+1}b_k \le a_kb_{k+1}$이 되도록 정렬
- $b_k / a_k \le b_{k+1} / a_{k+1}$이 되도록 정렬
- b_i / a_i 오름차순 정렬

질문?