알고리즘

6. Dynamic Programming

상명대학교 컴퓨터과학과

민 경 하

Contents

Contents

Motivation to DP

- 삼시세끼 메뉴 정하기
  - SM대학 근처에 3개의 식당이 있다.
    - 분식집 (양가네), 양식집 (Dogs & Cats), 그리고 중식집 (황하강)
  - 이 식당들의 메뉴는 다음과 같다
  - 삼시세끼를 풍성하게 즐겨보자
  - 반드시 세끼를 먹어야 한다. (한끼 생략하면 안됨)
  - 예산은 1만원을 넘기면 안 된다.
  - 아침은 반드시 양가네에서만 먹어야 한다.

	메뉴	가격
양가네 (AM 6:00~)	라면	2,000
	제육덮밥	3,000
	감자탕	4,000
	돈까스	5,000
Dogs&Cats (PM12:00~)	샐러드	3,000
	스파게티	4,000
	피자	5,000
황하강 (PM12:00~)	군만두	3,000
	짜장면	5,000
	짬뽕	5,000
	볶음밥	5,000

Motivation to DP

- 삼시세끼 메뉴 정하기
  - 제약 조건
    - 세끼를 다 먹을 것
    - 예산은 1만원 이내
    - 아침은 양가네에서만 먹을 것
  - 목표
    - 최대한 싸게 먹자
  - 전략
    - 메뉴를 가격순으로 정렬해서 앞에서부터 선택함
    - 라면 (2,000, 양), 제육 (3,000, 양), 샐러드 (3,000, D&C), 군만두 (3,000, 황)...
  - 답
    - 아침: 라면 (2,000), 점심: 라면 (2,000), 저녁: 라면 (2,000)
  - 결론
    - 최저의 예산 (6,000)으로 제약 조건을 모두 만족시키는 답을 구함

	메뉴	가격
양가네 (AM 6:00~)	라면	2,000
	제육덮밥	3,000
	감자탕	4,000
	돈까스	5,000
Dogs&Cats (PM12:00~)	샐러드	3,000
	스파게티	4,000
	피자	5,000
황하강 (PM12:00~)	군만두	3,000
	짜장면	5,000
	짬뽕	5,000
	볶음밥	5,000

Motivation to DP

- 삼시세끼 메뉴 정하기
  - 제약 조건
    - 세끼를 다 먹을 것
    - 예산은 1만원 이내
    - 아침은 양가네에서만 먹을 것
    - **전부 다른 집에서 먹자 (같은 식당 이용하면 안됨)**
  - 목표
    - 최대한 싸게 먹자
  - 전략
    - 메뉴를 가격순으로 정렬해서 앞에서부터 선택함
    - 단, 같은 식당에서 고른 메뉴가 있으면 패스
    - 라면 (2,000, 양), 제육 (3,000, 양), 샐러드 (3,000, D&C), 군만두 (3,000, 황)...
  - 답
    - 아침: 라면 (2,000, 양), 점심: 샐러드 (3,000, D&C), 저녁: 군만두 (3,000, 황)
  - 결론
    - 최저의 예산 (8,000)으로 제약 조건을 모두 만족시키는 답을 구함

	메뉴	가격
양가네 (AM 6:00~)	라면	2,000
	제육덮밥	3,000
	감자탕	4,000
	돈까스	5,000
Dogs&Cats (PM12:00~)	샐러드	3,000
	스파게티	4,000
	피자	5,000
황하강 (PM12:00~)	군만두	3,000
	짜장면	5,000
	짬뽕	5,000
	볶음밥	5,000

Motivation to DP

- 삼시세끼 메뉴 정하기
  - 제약 조건
    - 세끼를 다 먹을 것
    - 예산은 1만원 이내
    - 아침은 양가네에서만 먹을 것
    - 전부 다른 집에서 먹자 (같은 식당 이용하면 안됨)
    - 한끼는 반드시 밥으로 먹을 것
  - 목표
    - 최대한 싸게 먹자
  - 전략
    - 메뉴를 가격순으로 정렬해서 앞에서부터 선택함
    - 단, 같은 식당에서 고른 메뉴가 있으면 패스
    - 라면 (2,000, 양), 제육 (3,000, 양), 샐러드 (3,000, D&C), 군만두 (3,000, 황)...
  - 답
    - 아침: 라면 (2,000, 양), 점심: 샐러드 (3,000, D&C), 저녁: 볶음밥 (5,000, 황) → 10,000
  - 하지만,
    - 아침: 제육덮밥 (3,000, 양), 점심: 샐러드 (3,000, D&C), 저녁: 군만두 (3,000, 황) → 9,000

	메뉴	가격
양가네 (AM 6:00~)	라면	2,000
	제육덮밥	3,000
	감자탕	4,000
	돈까스	5,000
Dogs&Cats (PM12:00~)	샐러드	3,000
	스파게티	4,000
	피자	5,000
황하강 (PM12:00~)	군만두	3,000
	짜장면	5,000
	짬뽕	5,000
	볶음밥	5,000

Motivation to DP

- 삼시세끼 메뉴 정하기
  - 제약 조건
    - 세끼를 다 먹을 것
    - 예산은 1만원 이내
    - 아침은 양가네에서만 먹을 것
    - 전부 다른 집에서 먹자 (같은 식당 이용하면 안됨)
    - 한끼는 반드시 밥으로 먹을 것
  - 목표
    - 최대한 싸게 먹자
  - 전략
    - 메뉴를 가격순으로 정렬해서 앞에서부터 선택함
    - 단, 같은 식당에서 고른 메뉴가 있으면 패스
    - 라면 (2,000, 양), 제육 (3,000, 양), 샐러드 (3,000, D&C), 군만두 (3,000, 황)...
  - 결론
    - 그리디 알고리즘으로는 최적의 해를 구할 수 없다!!
    - 양가네에서 메뉴를 고를 때, 앞으로의 메뉴를 고려해서 결정해야 한다.

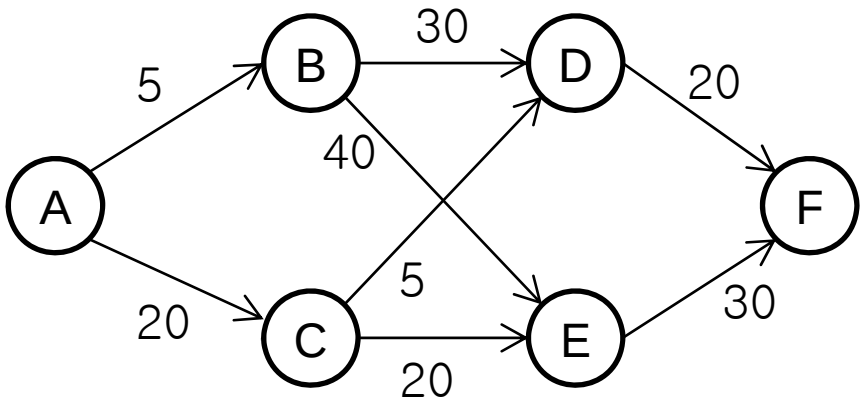
	메뉴	가격
양가네 (AM 6:00~)	라면	2,000
	제육덮밥	3,000
	감자탕	4,000
	돈까스	5,000
Dogs&Cats (PM12:00~)	샐러드	3,000
	스파게티	4,000
	피자	5,000
황하강 (PM12:00~)	군만두	3,000
	짜장면	5,000
	짬뽕	5,000
	볶음밥	5,000

6.0 Introduction

- Dynamic programming (동적 계획법)
  - 1940년대부터 Richard Bellman이 사용하기 시작함
  - Multistage decision making process에 대한 해법
  - 2차 세계 대전의 영향
    - 최적의 조합으로 자원을 배분하는 문제에 대해서 많은 연구가 진행됨
    - 최적의 판단을 내리기 위한 다양한 방법이 고려되었으며, 많은 계산량 때문에 컴퓨터를 이용한 방법이 연구됨.

6.0 Introduction

- Key idea of dynamic programming (1)
  - Comes from greedy algorithm
    - A sequence of selections
    - The counterexample of greedy algorithm
      - Find a shortest path from A to F: Cost(A → F)?

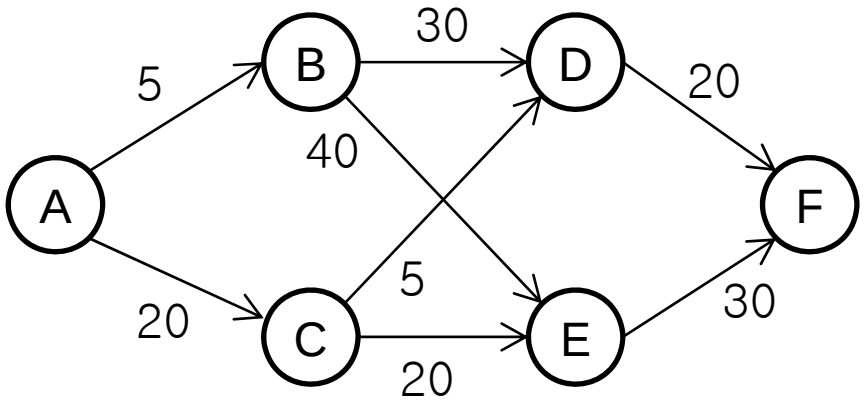


Can a greedy algorithm solve this problem?

6.0 Introduction

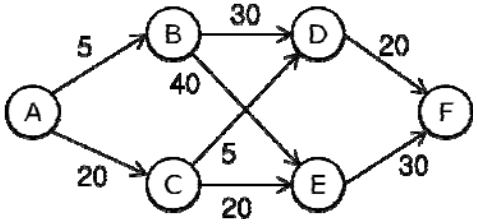
- Key idea of dynamic programming (2)
  - Comes from divide & conquer
    - Decomposing a problem into a set of sub-problems
    - Recursive structure of solution
      - Cost (A → F)?

Cost (A→F) = min(w(AB) + Cost (B→F), w(AC) + Cost(C→F))



6.0 Introduction

- Key idea of dynamic programming (3)
  - Recursive decomposition of a problem



Cost (A→F)

min(w(AB) + Cost (B→F), w(AC) + Cost(C→F))

Cost (B→F)

min(w(BD) + Cost (D→F), w(BE) + Cost(E→F))

Cost (C→F)

min(w(CD) + Cost (D→F), w(CE) + Cost(E→F))

Cost (D→F)

w(DF)

Cost (E→F)

w(EF)

6.0 Introduction

- Principle of optimality

> **Whatever the initial state and decision are, the remaining decisions must constitute an optimal decision sequence with regard to the state resulting from the first decision.**

- Leads to a recurrence relation
- Two approaches
  - There are two ways to formulate the recurrence relations
  - Forward approach
    - The formulation for decision x_i is made in terms of optimal decision sequences x_{i+1}, …, x_n.
  - Backward approach
    - The formulation for decision x_i is made in terms of optimal decision sequences x_1, …, x_{i-1}.

6.0 Introduction

다음 설명 중 옳지 않은 것을 모두 고르시오.

(a) dynamic programming은 연속적인 선택을 통해서 문제의 해를 구하는 방법이다.

(b) dynamic programming은 greedy algorithm보다 더 효율적인 해를 구할 수 있다.

(c) dynamic programming에서 문제를 recurrence relation을 이용해서 설정할 수 있다.

(d) dynamic programming의 recurrence relation은 초항부터 풀어야 한다.