제5장 동적 계획 알고리즘 (1)

과 목 명 정 보 처 리 알 고 리 즘

담당교수 김 성 훈

경북대학교 과학기술대학 소프트웨어학과

이 장에서 배울 내용

1. 동적 계획 알고리즘의 기본개념
2. 모든 쌍 최단 경로(All Pairs Shortest Path)
3. 연속 행렬 곱셈(Chained Matrix Multiplication)
4. 동전 거스름돈 문제(Coin Change)
5. 편집 거리 문제(Edit Distance Problem)
6. 배낭문제(Knapsack Problem)

사고의 전환:
동적계획 유사 전략(Analogy)

- 동물들의 헌팅 전략들:

  - 낚시 전략: 먹이사슬을 활용하여 미끼작전, 생존본능 활용
  - 물소 사냥: 사자의 물소 떼 사냥 전략, 이동 및 무리본능 활용
  - 연어낚시: 연어 알을 계속에서 산란시켜 방사, 회귀본능 활용
  - 헌팅전략: 새끼를 이용한 유인 함정만들기, 포유류의 모성본능 활용

- 마케팅 전략: 미끼상품, 쿠폰, 사은품, 상품권 당첨 등, 고객의 공짜심리 활용
- 도미노 게임: 말 하나로 전체의 말을 쓰러뜨리도록 치밀하게 계획
- 핵 폭탄의 연쇄 반응: 작은 핵 융합 반응의 연쇄작용으로 큰 폭발을 야기
- 인류의 농업 발명: 씨앗을 먹지 않고 땅에 뿌려두면,
  이것이 자라서 더 큰 수확을 거둔다. 식물의 성장원리를 활용

→ Bottom-UP Approach

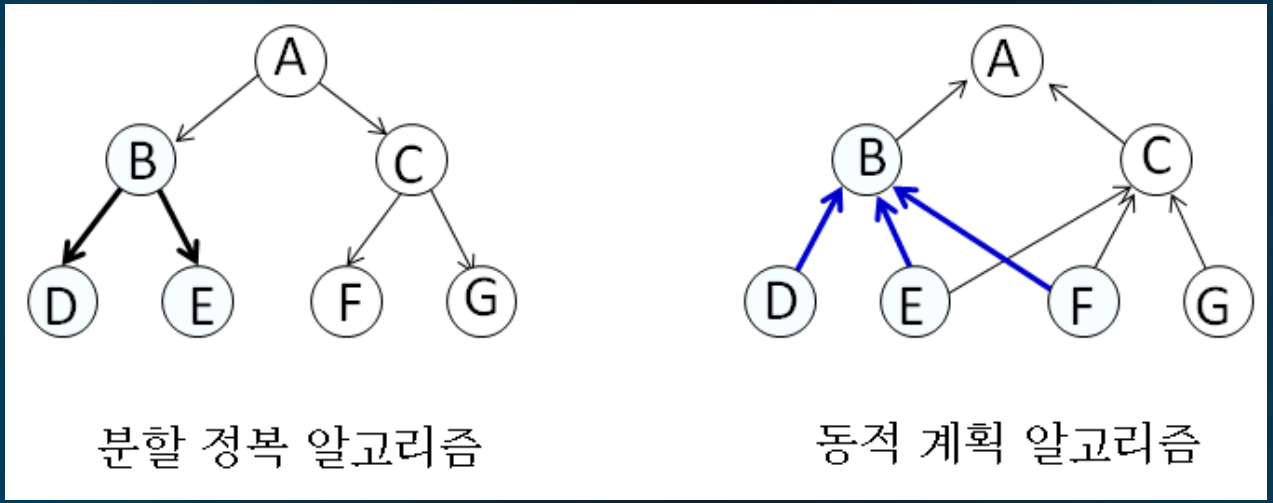
동적 계획 알고리즘 기본 개념

- 동적 계획 (Dynamic Programming) 알고리즘은
  그리디 알고리즘과 같이 최적화 문제를 해결하는 알고리즘이다.

- 동적 계획 알고리즘은,
  - 먼저 입력 크기가 작은 부분 문제들을 모두 해결한 후에
  - 그 해들을 이용하여 보다 큰 크기의 부분 문제들을 해결하여,
  - 최종적으로 원래 주어진 입력의 문제를 해결하는 알고리즘이다.

동적 계획 알고리즘 기본 개념(2)

- 분할 정복 알고리즘과 동적 계획 알고리즘의 전형적인 부분문제들 사이의 관계



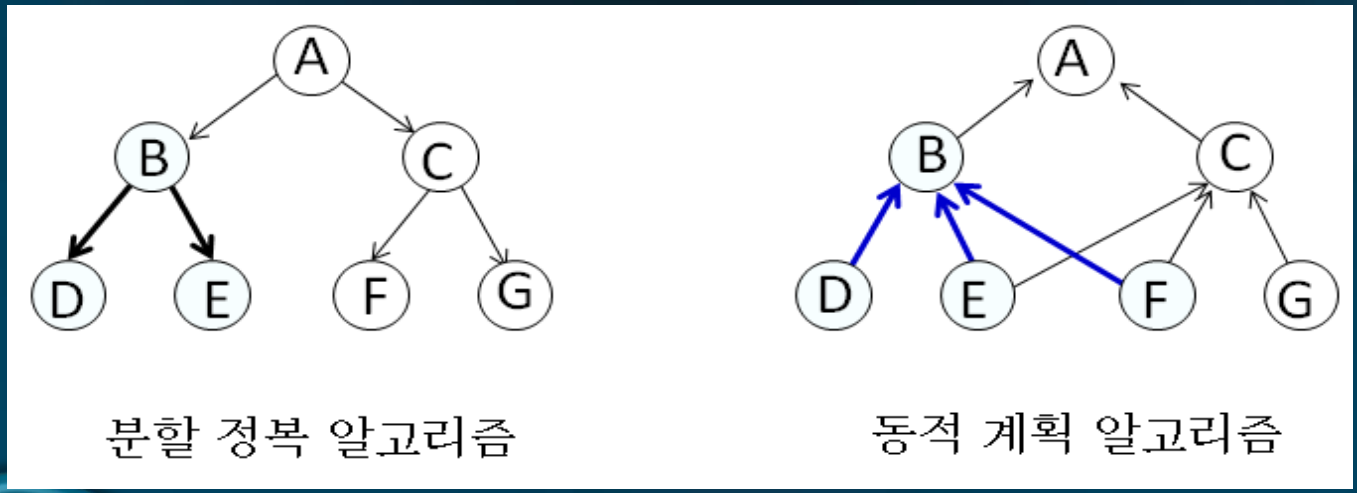
- **분할 정복 알고리즘**의 부분문제들 사이의 관계:
  - A는 B와 C로 분할되고, B는 D와 E로 분할되는데, D와 E의 해를 취합하여 B의 해를 구한다. 단, D, E, F, G는 각각 더 이상 분할할 수 없는 (또는 가장 작은 크기의) 부분문제들이다.
  - 마찬가지로 F와 G의 해를 취합하여 C의 해를 구하고, 마지막으로 B와 C의 해를 취합하여 A의 해를 구한다.

동적 계획 알고리즘 기본 개념(3)

- **동적 계획 알고리즘**은

  먼저 최소 단위의 부분 문제 D, E, F, G의 해를 각각 구한다. 그 다음에,

  - D, E, F의 해를 이용하여 B의 해를 구한다.
  - E, F, G의 해를 이용하여 C의 해를 구한다.
  - B와 C의 해를 구하는데 E와 F의 해 모두를 이용한다.
  - 분할 정복은 부분문제의 해를 중복 사용하지 않지만, 동적 계획은 중복 사용한다.



동적 계획 알고리즘 기본 개념(4)

- 동적 계획 알고리즘에는 부분문제들 사이에 의존적 관계가 존재한다.
  - 예를 들면, D, E, F의 해가 B를 해결하는데 사용되는 관계가 있다.

- 이러한 관계는 문제 또는 입력에 따라 다르고,
  대부분의 경우 뚜렷이 보이지 않아서 '함축적인 순서' (implicit order)라고 한다.

5.1 모든 쌍 최단 경로

- 모든 쌍 최단 경로 (All Pairs Shortest Paths) 문제는
  각 쌍의 점 사이의 최단 경로를 찾는 문제이다.

	서울 Seoul	인천 Incheon	수원 Suwon	대전 Daejeon	전주 Jeonju	광주 Gwangju	대구 Daegu	울산 Ulsan	부산 Busan ›
서울 Seoul		40.2	41.3	154	232.1	320.4	297	407.5	432
인천 Incheon			54.5	174	253.3	351.6	317.6	447	453
수원 Suwon				132.6	189.4	299.6	268.1	356	390.7
대전 Daejeon					96.9	185,2	148.7	259.1	283.4
전주 Jeonju						105.9	219.7	331.1	322.9
광주 Gwangju							219.3	329.9	268
대구 Daegu								111.1	135.5
울산 Ulsan									52.9
부산 Busan									

플로이드-워샬 알고리즘 소개

- 이 문제를 해결하려면, 각 점을 시작점으로 정하여
  다익스트라(Dijkstra)의 최단 경로 알고리즘을 수행하면 된다.
  - 이때의 시간복잡도는 배열을 사용하면 $(n-1)xO(n^2) = O(n^3)$이다.
    단, n은 점의 수이다.
- Warshall은 그래프에서 모든 쌍의 경로 존재 여부(transitive closure)를
  찾아내는 동적 계획 알고리즘을 제안했고,
  Floyd는 이를 변형하여 모든 쌍 최단 경로를 찾는 알고리즘을 고안하였다.
- 따라서, 모든 쌍 최단 경로를 찾는 동적 계획 알고리즘을
  플로이드-워샬 알고리즘이라 한다. (간략히 플로이드 알고리즘이라 함.)

플로이드-워샬 알고리즘 소개(2)

- 플로이드 알고리즘이 다익스트라 알고리즘보다 나은가?

- 결론적으로 말하면,

  플로이드 알고리즘의 시간복잡도는 $O(n^3)$으로

  다익스트라 알고리즘을 사용하는 것과 시간복잡도는 동일하다.

- 그러나 플로이드 알고리즘은 매우 간단하여

  다익스트라 알고리즘을 사용하는 것보다 좀 더 효율적이다.

- 왜 일까?

부분 문제의 정의

- 동적 계획 알고리즘으로 모든 쌍 최단 경로 문제를 해결하려면
  먼저 부분문제들을 찾아야 한다.
- 이를 위해 일단 그래프의 점의 수가 적을 때를 생각해보자.
  - 그래프에 3개의 점이 있는 경우,
    점 i에서 점 j까지의 최단 경로를 찾으려면 2가지 경로,
    즉, 점 i에서 점 j로 직접 가는 경로와 점 1을 경유하는 경로 중에서 짧은 것을 선택하면 된다.

부분 문제의 정의(2)

- 또 하나의 중요한 아이디어는 경유 가능한 점들을
  - 점 1로부터 시작하여,
    점 1과 2,
    그 다음엔 점 1, 2, 3으로 하나씩 추가하여,
    마지막에는 점 1~n까지의 모든 점을 경유 가능한 점들로 고려하면서,
    모든 쌍의 최단 경로의 거리를 계산한다.
- 부분문제 정의:

  입력 그래프의 점을 각각 1, 2, 3, …, n이라 하자. 이때,

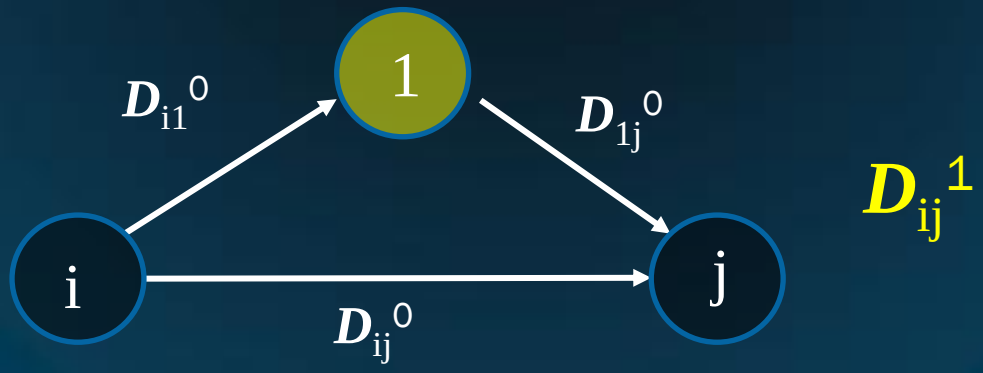
  D_{ij}^{k} = 점 {1, 2, …, k}만을 경유 가능한 점들로 고려하여,
  점 i로부터 점 j까지의 모든 경로 중에서
  가장 짧은 경로의 거리

부분 문제의 초기화

- 여기서 주의할 것은, 점 {1, 2, …, k}만을 고려한다는 것은
  점 1에서 점 k까지의 모든 점들을 반드시 경유하는 경로를 의미하는 것이 아니다.
- 심지어는 D_{ij}^{k}는 이 점들을 하나도 경유하지 않으면서 점 i에서 점 j에 도달하는 경로, 즉 선분 (i,j)가 최단 경로가 될 수도 있다.
- 여기서 k≠i, k≠j이고, k=0인 경우, 점 0은 그래프에 없으므로 어떤 점도 경유하지 않는다는 것을 의미한다.
  따라서 D_{ij}^{0}은 입력으로 주어지는 선분 (i,j)의 가중치이다.

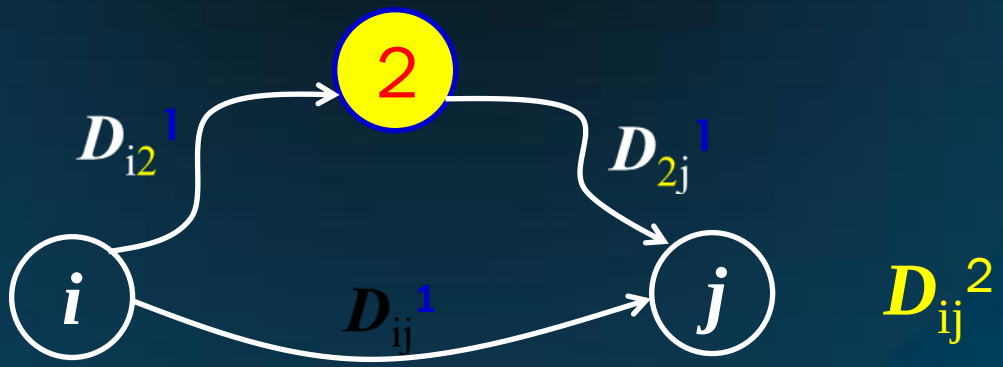
부분문제들 사이의 의존적 관계

- D_{ij}^{1}은 i에서 점1을 경유하여 j로 가는 경로와,
  i에서 j로 직접 가는 경로, 즉 선분 (i, j)중에서 짧은 거리이다.
- 따라서, 모든 쌍 i와 j에 대하여,
  D_{ij}^{1}를 계산하는 것이 가장 작은 부분문제들이다.
  단, i≠1, j≠1 이다.



부분문제들 사이의 의존적 관계(2)

- 그 다음엔 i에서 점 2를 경유하여 j로 가는 경로의 거리와
  D_{ij}^{1} 중에서 짧은 거리를 D_{ij}^{2}로 정한다.
  단, 점 2를 경유하는 경로의 거리는 $D_{i2}^{1} + D_{2j}^{1}$ 이다.
- 모든 쌍 i와 j에 대하여 D_{ij}^{2}를 계산하는 것이
  그 다음으로 큰 부분 문제들이다. 단, i≠2, j≠2이다.

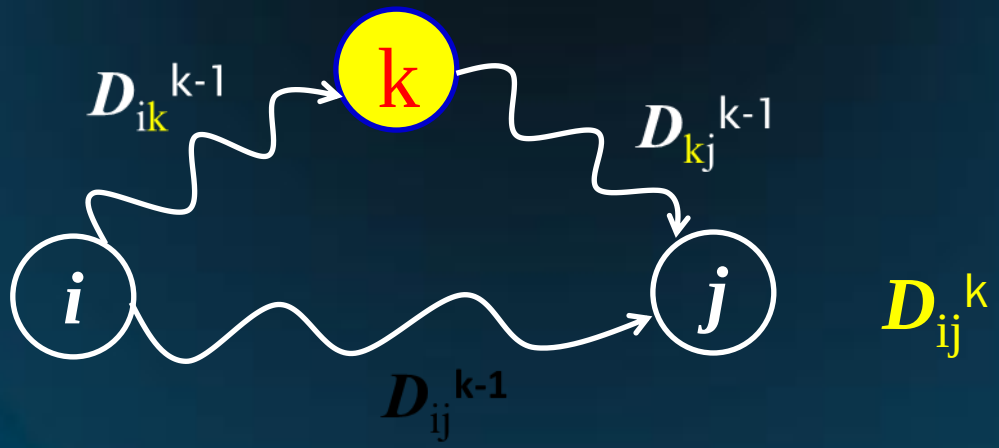


부분문제들 사이의 의존적 관계(3)

- 점 i에서 점 k를 경유하여 j로 가는 경로의 거리와

  D_{ij}^{k-1} 중에서 짧은 것을 로 정한다.

  단, 점 k를 경유하는 경로의 거리는 $D_{ik}^{k-1} + D_{kj}^{k-1}$ 이고, i≠k, j≠k이다.



부분문제들 사이의 의존적 관계(4)
→ 함축적인 순서

- 이런 방식으로 k가 1에서 n이 될 때까지 D_{ij}^{k}를 계산해서,

  D_{ij}^{n}, 즉, 모든 점을 경유 가능한 점들로 고려된

  모든 쌍 i와 j의 최단 경로의 거리를 찾는 방식이

  플로이드의 모든 쌍 최단 경로 알고리즘이다.

모든 쌍 최단 경로알고리즘

AllPairsShortest

입력: 2차원 배열 D, 단, D[i,j]=선분 (i,j)의 가중치,
만일 선분 (i,j)이 존재하지 않으면 D[i,j]=∞,
모든 i에 대하여 D[i,i]=0이다.

출력: 모든 쌍 최단 경로의 거리를 저장한 2-d 배열 D

1. for k = 1 to n
2. for i = 1 to n (단, i≠k)
3. for j = 1 to n (단, j≠k, j≠i)
4. D[i,j] = min { D[i,k]+D[k,j] , D[i,j] }

음수 사이클 제약

- AllPairsShortest 알고리즘의 입력 그래프에는

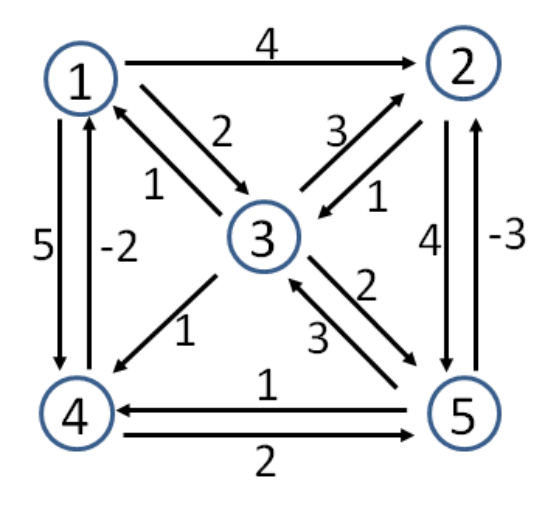
  사이클 상의 선분들의 가중치 합이 음수가 되는 사이클은 없어야 한다.

- 이러한 사이클을 음수 사이클 (negative cycle)이라 하는데,

  최단 경로를 찾는데 음수 사이클이 있으면,

  이 사이클을 반복하여 돌아 나올 때마다 경로의 거리가 감소되기 때문이다.

AllPairsShortest 알고리즘 수행 과정



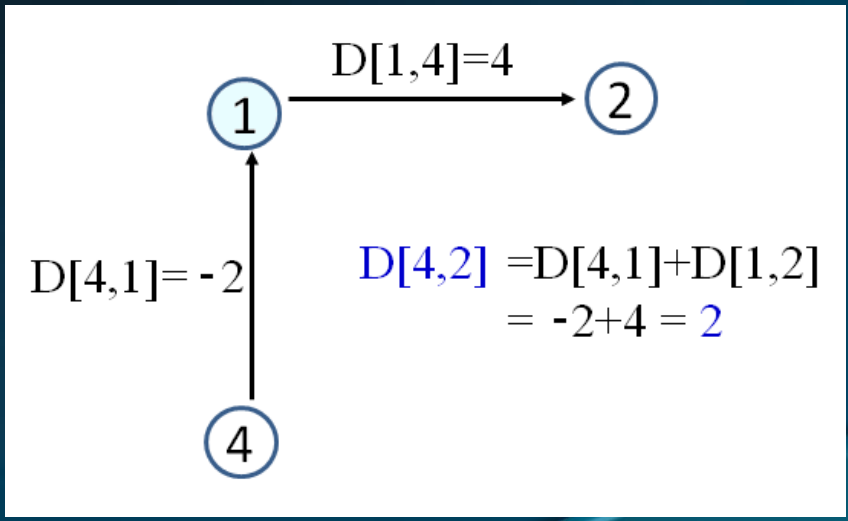
D	1	2	3	4	5
1	0	4	2	5	∞
2	∞	0	1	∞	4
3	1	3	0	1	2
4	-2	∞	∞	0	2
5	∞	-3	3	1	0

- 배열 D의 원소들이 k가 1부터 5까지 증가함에 따라서 갱신되는 것을 살펴보자.

AllPairsShortest 알고리즘 수행 과정(2)

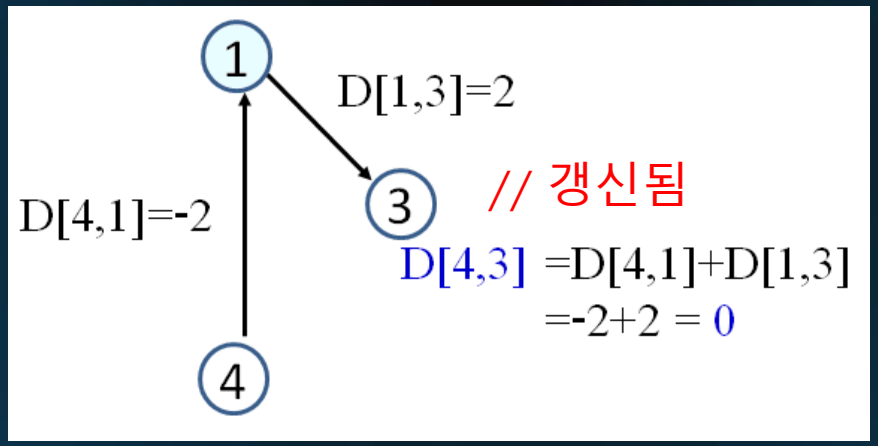
- k=1일 때:
  - D[2,3] = min{D[2,3], D[2,1]+D[1,3]} = min{1, ∞+2} = 1
  - D[2,4] = min{D[2,4], D[2,1]+D[1,4]} = min{∞, ∞+5} = ∞
  - D[2,5] = min{D[2,5], D[2,1]+D[1,5]} = min{4, ∞+∞} = 4
  - D[3,2] = min{D[3,2], D[3,1]+D[1,2]} = min{3, 1+4} = 3
  - D[3,4] = min{D[3,4], D[3,1]+D[1,4]} = min{1, 1+5} = 1
  - D[3,5] = min{D[3,5], D[3,1]+D[1,5]} = min{2, 1+∞} = 2
  - D[4,2] = min{D[4,2], D[4,1]+D[1,2]} = min{∞, -2+4} = 2

// 갱신됨



AllPairsShortest 알고리즘 수행 과정(3)

- D[4,3] = min{D[4,3], D[4,1]+D[1,3]} = min{∞, -2+2} = 0



- D[4,5] = min{D[4,5], D[4,1]+D[1,5]} = min{2, -2+∞} = 2
- D[5,2] = min{D[5,2], D[5,1]+D[1,2]} = min{-3, ∞+4} = -3
- D[5,3] = min{D[5,3], D[5,1]+D[1,3]} = min{3, ∞+2} = 3
- D[5,4] = min{D[5,4], D[5,1]+D[1,4]} = min{1, ∞+5} = 1

AllPairsShortest 알고리즘 수행 과정(4)
(빠른 계산법)

- k=1일 때 D[4,2], D[4,3]이 각각 2, 0으로 갱신된다. 다른 원소들은 변하지 않았다.

D	1	2	3	4	5
1	0	4	2	5	∞
2	∞	0	1	∞	4
3	1	3	0	1	2
4	-2	∞	∞	0	2
5	∞	-3	3	1	0

D	1	2	3	4	5
1	0	4	2	5	∞
2	∞	0	1	∞	4
3	1	3	0	1	2
4	-2	**2**	**0**	0	2
5	∞	-3	3	1	0

AllPairsShortest 알고리즘 수행 과정(5)
(빠른 계산법)

- k=2일 때:
  - D[1,5]가 1 → 2 → 5의 거리인 8로 갱신된다.
  - D[5,3]이 5 → 2 → 3의 거리인 -2로 갱신된다.

D	1	2	3	4	5
1	0	4	2	5	∞
2	∞	0	1	∞	4
3	1	3	0	1	2
4	-2	**2**	**0**	0	2
5	∞	-3	3	1	0

D	1	2	3	4	5
1	0	4	2	5	**8**
2	∞	0	1	∞	4
3	1	3	0	1	2
4	-2	2	0	0	2
5	∞	-3	**-2**	1	0

AllPairsShortest 알고리즘 수행 과정(6)
(빠른 계산법)

- k=3일 때 총 7개의 원소가 갱신된다.

D	1	2	3	4	5
1	0	4	2	5	**8**
2	∞	0	1	∞	4
3	1	3	0	1	2
4	-2	2	0	0	2
5	∞	-3	**-2**	1	0

D	1	2	3	4	5
1	0	4	2	**3**	**4**
2	**2**	0	1	**2**	**3**
3	1	3	0	1	2
4	-2	2	0	0	2
5	**-1**	-3	-2	**-1**	0

AllPairsShortest 알고리즘 수행 과정(7)
(빠른 계산법)

- k=4일 때 총 3개의 원소가 갱신된다.

D	1	2	3	4	5
1	0	4	2	**3**	**4**
2	**2**	0	1	**2**	**3**
3	1	3	0	1	2
4	-2	2	0	0	2
5	**-1**	-3	-2	**-1**	0

D	1	2	3	4	5
1	0	4	2	3	4
2	**0**	0	1	2	3
3	**-1**	3	0	1	2
4	-2	2	0	0	2
5	**-3**	-3	-2	-1	0

AllPairsShortest 알고리즘 수행 과정(8)
(빠른 계산법)

- k=5일 때 총 3개의 원소가 갱신되고, 이것이 주어진 입력에 대한 최종해이다.

D	1	2	3	4	5
1	0	4	2	3	4
2	**0**	0	1	2	3
3	**-1**	3	0	1	2
4	-2	2	0	0	2
5	**-3**	-3	-2	-1	0

D	1	2	3	4	5
1	0	**1**	2	3	4
2	0	0	1	2	3
3	-1	**-1**	0	1	2
4	-2	**-1**	0	0	2
5	-3	-3	-2	-1	0

시간복잡도

- AllPairsShortest의 시간복잡도는 위의 예제에서 보았듯이

  각 k에 대해서 모든 i, j 쌍에 대해 계산되므로,

  총 nxnxn = n^3회 계산이 이루어지고,

  각 계산은 O(1) 시간이 걸린다.

- 따라서 AllPairsShortest의 시간복잡도는 $O(n^3)$이다.

응 용

- 맵퀘스트 (Mapquest)와 구글 (Google) 웹사이트의 지도 서비스
- 자동차 네비게이션 서비스
- 지리 정보 시스템 (GIS)에서의 네트워크 분석
- 통신 네트워크와 모바일 통신 분야
- 게임
- 산업 공학, 경영 공학의 OR(Operations Research) 문제
- 로봇 공학
- 교통 공학
- VLSI 디자인 분야 등

Summary

- 동적 계획알고리즘은 최적화 문제를 해결하는 알고리즘으로서,
  - 입력 크기가 작은 부분문제들을 모두 해결한 후에 그 해들을 이용하여, 보다 큰 크기의 부분문제들을 해결해 나가는 과정을 거치면서, 최종적으로 원래 주어진 입력의 문제를 해결하는 알고리즘이다.
  - 동적 계획 알고리즘에는 부분문제들 사이에 의존적 관계가 존재한다.
- 모든쌍 최단경로 문제의 Floyd-Warshall 알고리즘은 $O(n^3)$ 시간에 해를 찾는다.
  - 핵심 아이디어는 경유 가능한 점들을 점 1로부터 시작하여, 점 1과 2, 그 다음엔 점 1, 2, 3으로 하나씩 추가하여, 마지막에는 점 1에서 점 n까지의 모든 점을 경유 가능한 점들로 고려하면서, 모든 쌍의 최단 경로의 거리를 계산하는 것이다.

동적 계획의 핵심

- 문제의 본질을 꿰뚫어보는 통찰력(insight)으로,
  - 문제 자체에 내재되어 있는 원리을 발견하고,
  - 가장 작은 부분문제(elementary subproblem)을 찾아서,
  - 이들간의 의존적인 관계를 바탕으로 하여,
  - 전체 문제를 재구성하는 **함축적인 순서** 규칙를 찾는다.

실 습

실습1: 모든쌍 최단경로 문제를 플로이드 알고리즘으로 해결한 파이선 프로그램 작성하기

숙제:

1. 모든 쌍 최단 경로 문제를 다익스트라 알고리즘으로 구현하고, 플로이드 알고리즘과의 성능 비교실험하기.

Q&A