

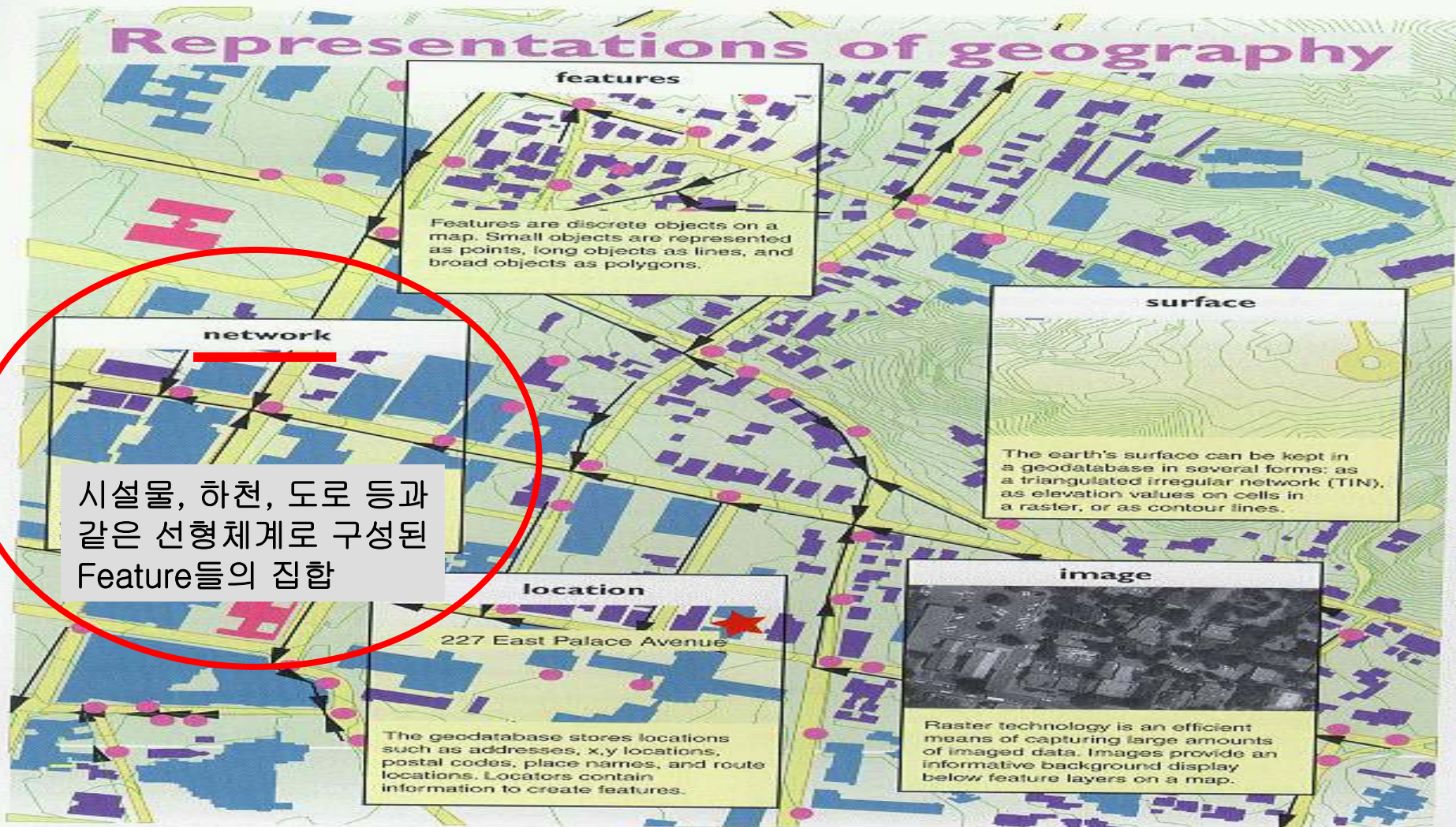
교통 네트워크 모델링과 분석



1. 네트워크의 개요

- GIS에서 실세계의 표현 방법
 - 네트워크의 이해
 - 네트워크의 표현(도식적 vs. 기하학적)
 - 교통 네트워크의 추상화
 - 교통 네트워크 데이터 모델과 구조
 - 기본 노드-링크 구조의 문제점
 - 문제 해결의 대안
 - Dynamic Segmentation의 이해
 - 교통 네트워크의 활용
-

GIS에서 실세계의 표현 방법



Network의 이해 (1/4)

- 네트워크 (Network)에 관한 기초적인 수학적 모델의 기원
→ 그래프 이론

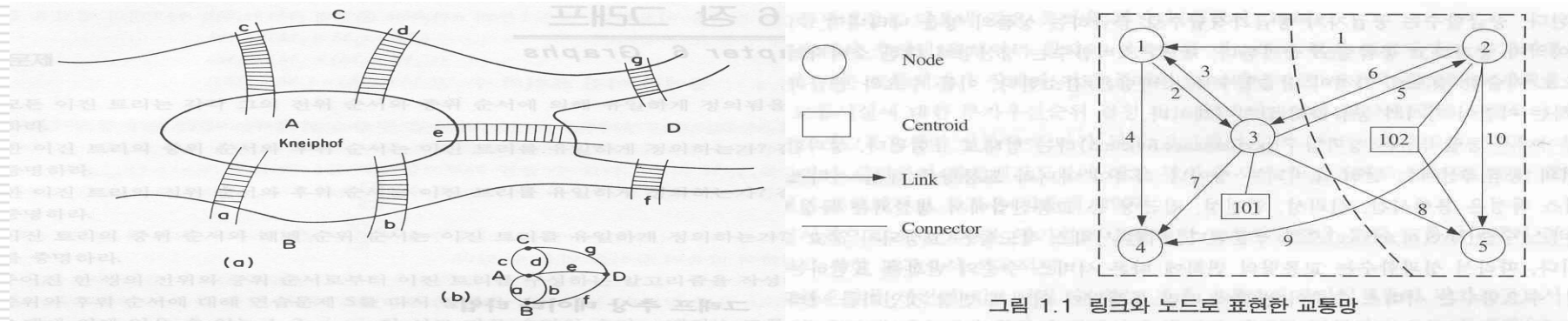
 - 그래프
 - 집합을 구성하는 원소들 사이의 관계나 연결성을 표현 및 분석
 - 점(point or vertex) ~ 정수로 표현되는 unique label로 표시
 - 선(arc, edge or line segment) ~ 점들간의 논리적 관계나 물리적 연결을 표시
 - 그래프 이론에서는 점들의 연결성에만 초점을 맞추므로 방향성은 갖지 않는 것이 일반적

 - 평면 그래프
 - 두 개의 선이 교차할 때마다 하나의 점이 존재하게 됨
-

Network의 이해 (2/4)

□ 네트워크 (Network)

- 점과 선으로 구성된 집합인 그래프(graph)의 특수 형태
- 선에 수치 파라미터(parameter or weight)가 부여된 그래프
 - 수치 파라미터(+): 선의 길이, 선을 통과하는 단위 비용 또는 시간 등
- 시각적이고 개념적인 표현이 가능하므로, 실세계 문제를 네트워크로 표현하면 효율적 최적화 알고리즘을 이용하여 해결 가능
- 관련 분야
 - 교통, 통신, 에너지 분배, 재고, 물류, 관개 시스템, 컴퓨터 네트워크, 사업계획, 설비대체, 인력관리 등 과학, 사회, 경제, 공학 등 많은 분야에서 네트워크 이론을 사용하고 있음



Network의 이해 (3/4)

□ 네트워크의 수학적 정의

■ $G = (N, A)$

□ N : $n = |N|$ 인 노드들의 색인 집합, n =노드들의 수

□ A : $m = |A|$ 인 방향성 아크들의 집합, m = 아크들의 수

■ 네트워크에서 각 아크는 방향을 갖는 노드들의 쌍으로 표현

□ (i, j) : 노드 i 에서 노드 j 로 가는 아크를 의미

■ GIS에서 네트워크 아크는 네트워크 링크라고 불리기도 함

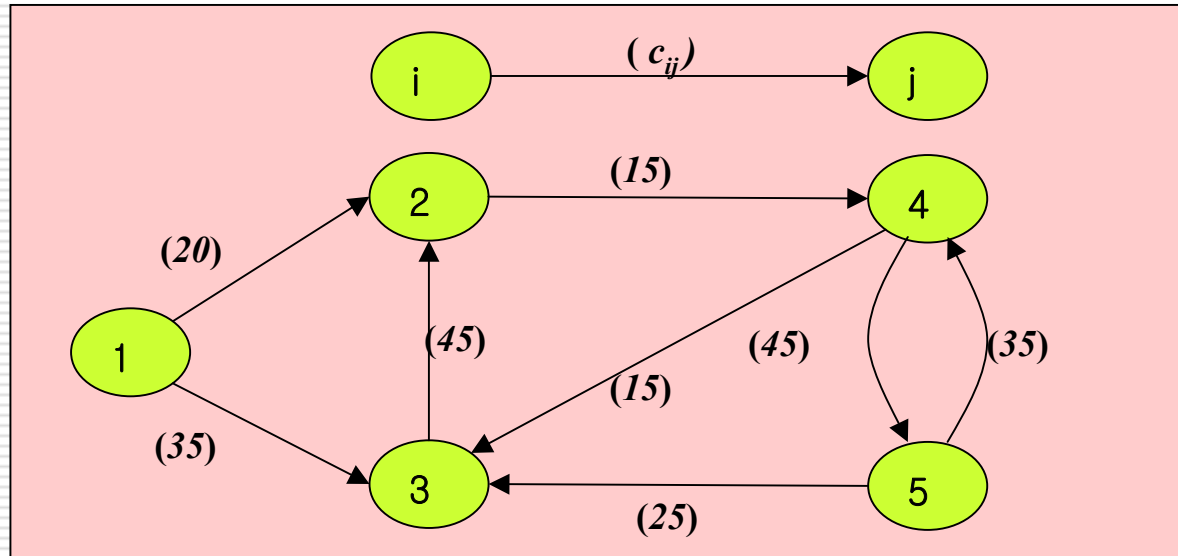
□ 교통 네트워크

■ 교차로와 인터체인지에서 서로 연결되는 선형 feature들의 구성 체계

■ 교차로와 인터체인지는 노드라고 함

■ 임의의 노드들의 쌍에 의해 연결된 선형 feature들은 아크라고 함

Network의 이해 (4/4)



□ 예제 네트워크

- 노드 – 숫자를 원으로 둘러싼 형태로 표현하고 있으며, 각 원 내의 숫자는 노드 식별자(node identifier: ID)를 의미
- 아크 – 노드들을 연결하고 있는 방향성 화살표로 표현
- 각 아크에 부여된 숫자는 그 아크를 통행하는데 소요되는 거리 또는 시간을 의미

Network의 표현 (1/4)

□ 두 가지 표현 방법

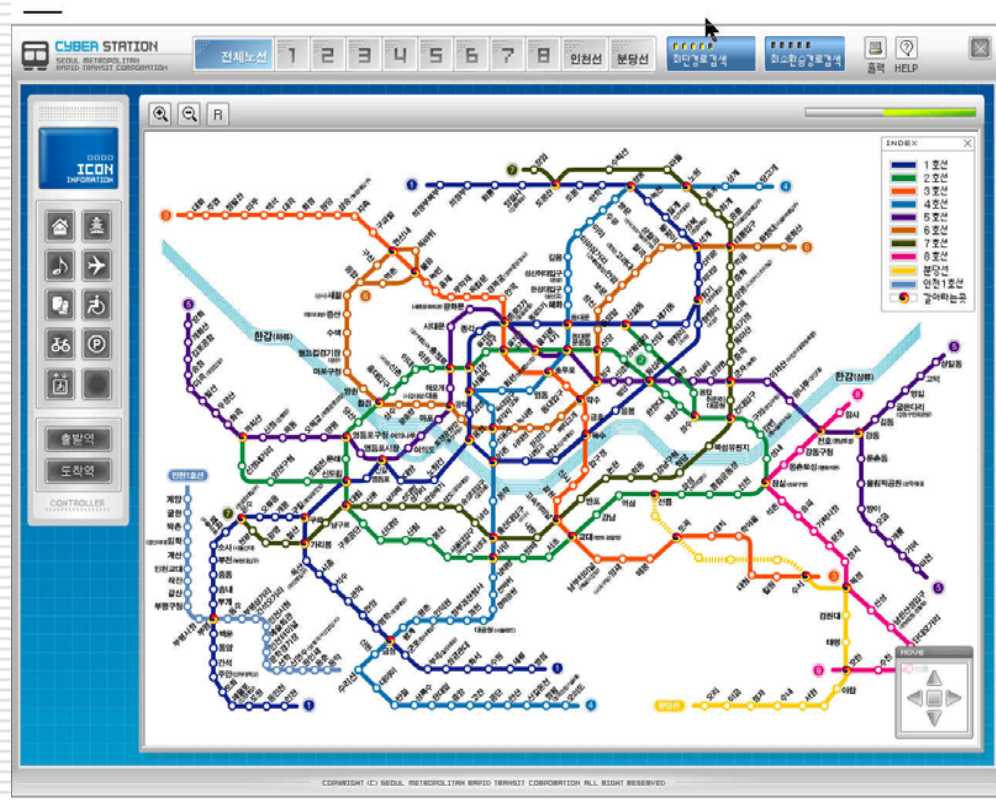
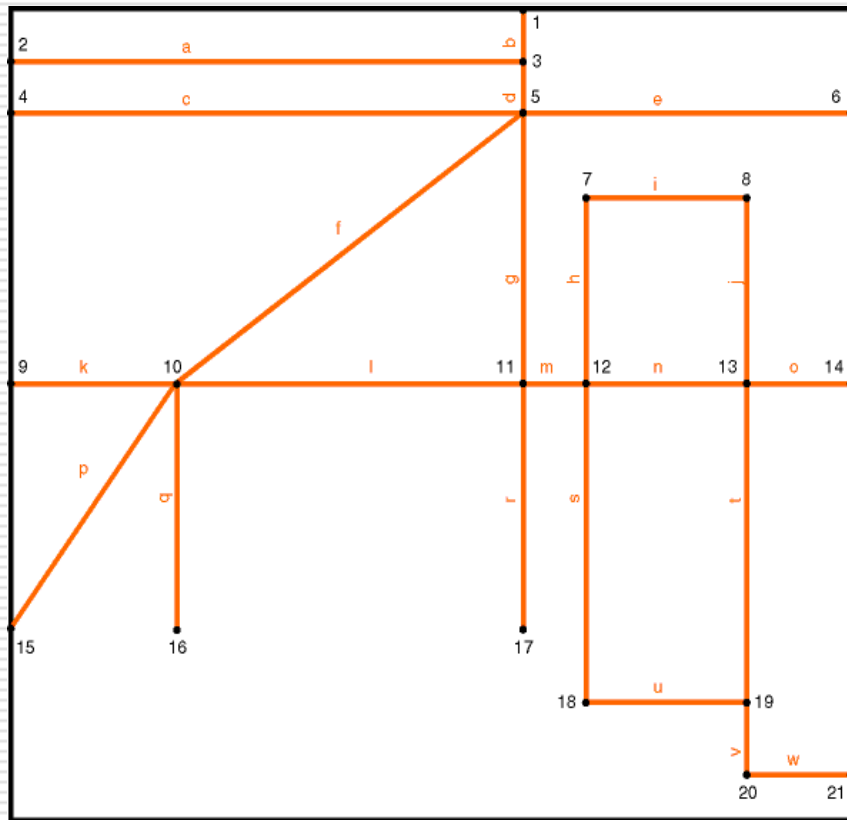
- 도식적 구조
- 기하학적 구조

□ 도식적 구조(schematics)로서의 네트워크

- 네트워크는 링크(*links*)와 노드(*nodes*)들로 이루어짐
 - 노드에는 그 위치를 표현하는 좌표가 연계됨
 - 이러한 좌표는 편의상 또는 미적인 표현을 위해 의도적으로 왜곡되어 표현하는 경우가 많음
 - 예: 지하철 또는 도시철도 네트워크 다이어그램
 - 각종 관련 속성들이 이러한 도식적 표현에 연계될 수 있음
 - 예: 가로명(street name), 차로수(number of lanes), 통행시간(travel time) 등
-

Network의 표현 (2/4)

□ 도식적 구조(schematics)로서의 네트워크 예



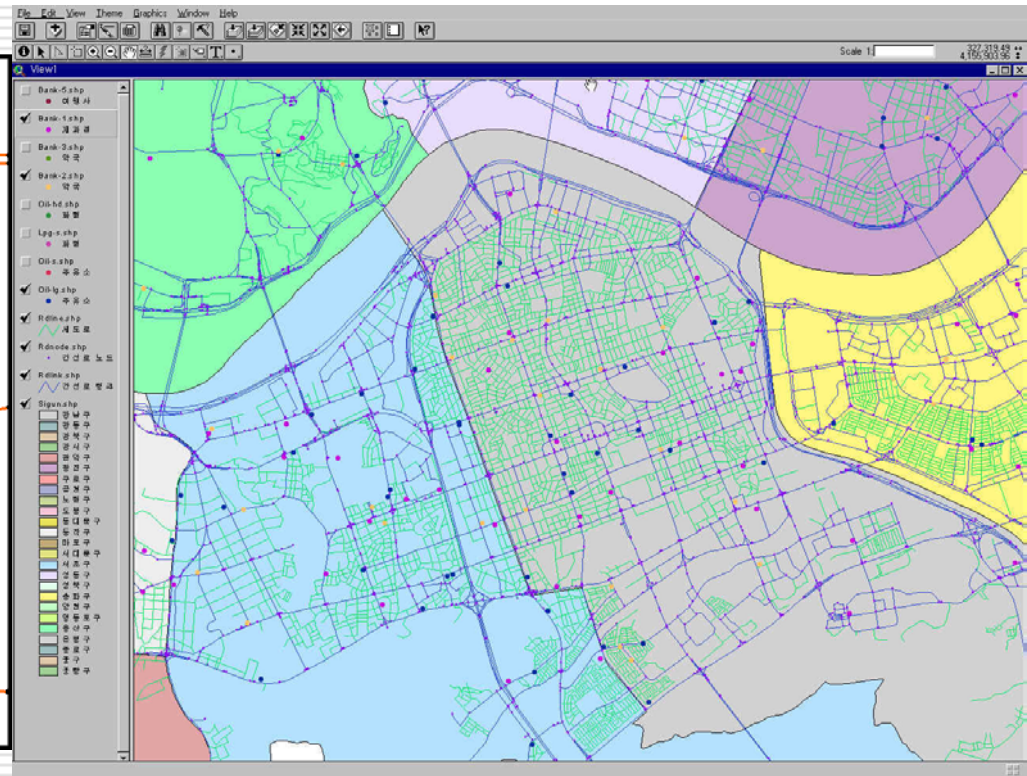
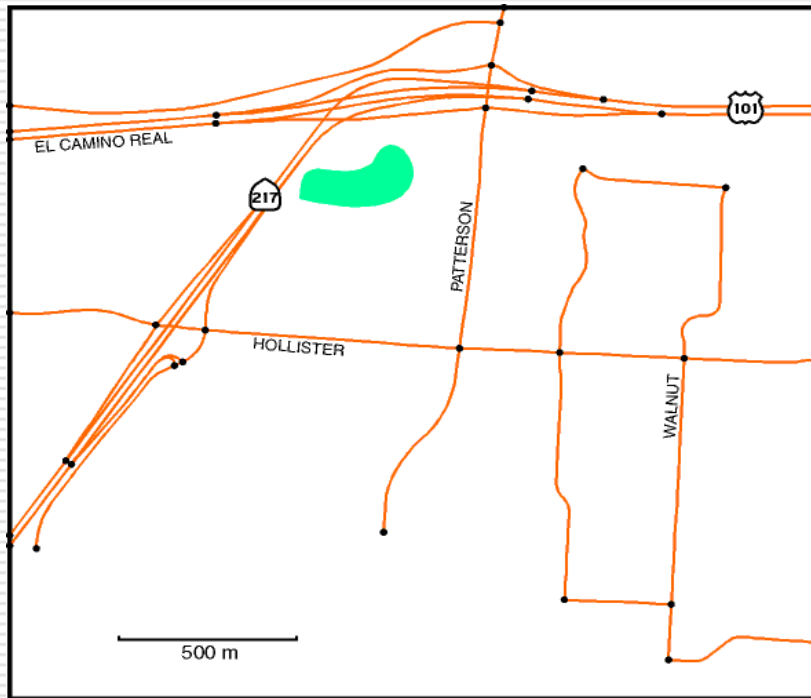
Network의 표현 (3/4)

□ 기하학적 구조로서의 네트워크

- 링크의 기하학적 경로(*geometric path*) 표현이 GIS 측면에서는 더 중요하며 (예: road, railroad), 이것은 어떤 링크를 표현할 때 그 링크의 형태를 구성하는 점(*shape point*)들의 좌표로 표현됨
 - 교통 네트워크 분석과 같은 실제 지리적 환경을 고려한 연구 등에서는 이러한 표현이 더 많이 이용됨
 - 예: 교통 링크의 경제적 영향 분석, 환경적 영향 분석, 지진피해 분석 등
 - 기하학적 표현은 표현하고자 하는 지도 축척에 따라 다양하게 일반화될 수 있음 (레벨 정의)
 - 예: 고속도로는 양방향 표현도 가능하지만, 도로 중심선 하나로 표현할 수 있으며, 램프는 표현될 수도 있고 그렇지 않을 수도 있음
-

Network의 표현 (4/4)

□ 지리 데이터베이스로서의 네트워크

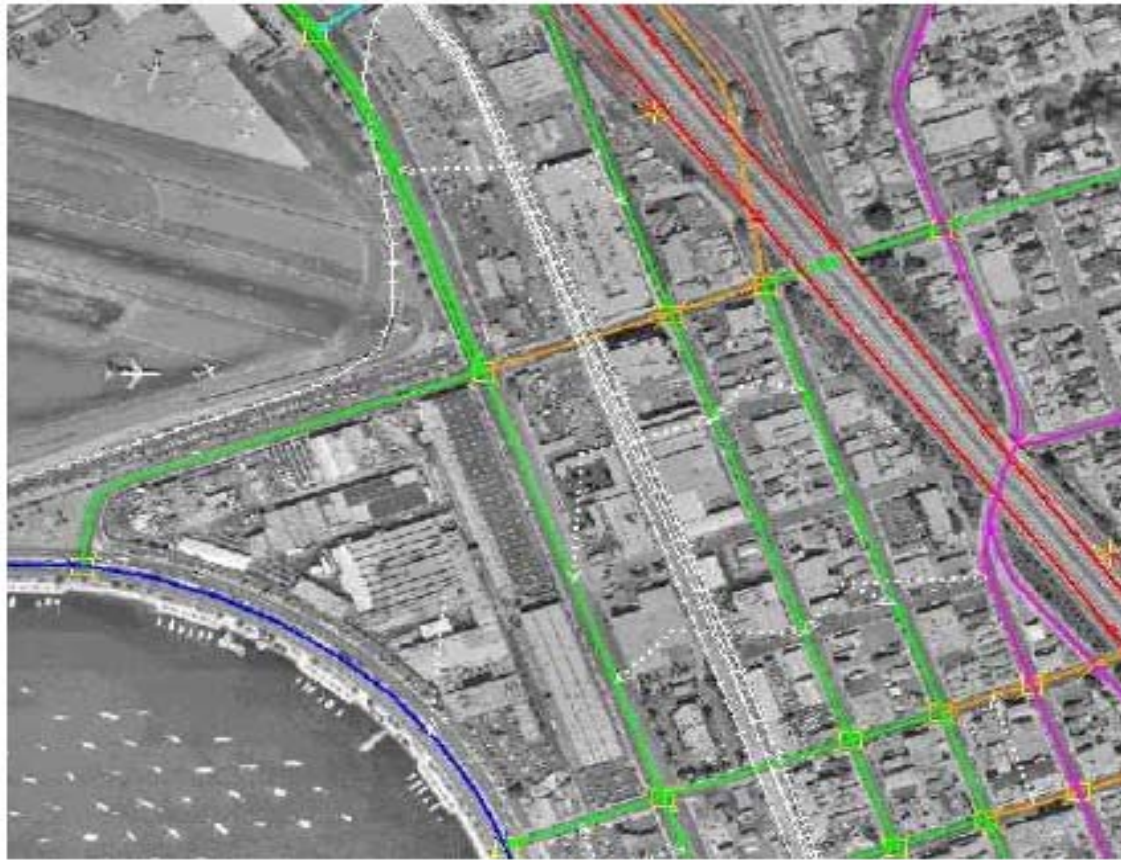


교통 네트워크의 추상화

- 도로 중심선(*centerlines*)을 이용하여 추상화하는 것이 일반적
 - 도로 중심선이 교통활동의 분석처리를 위해 더 유용하기 때문



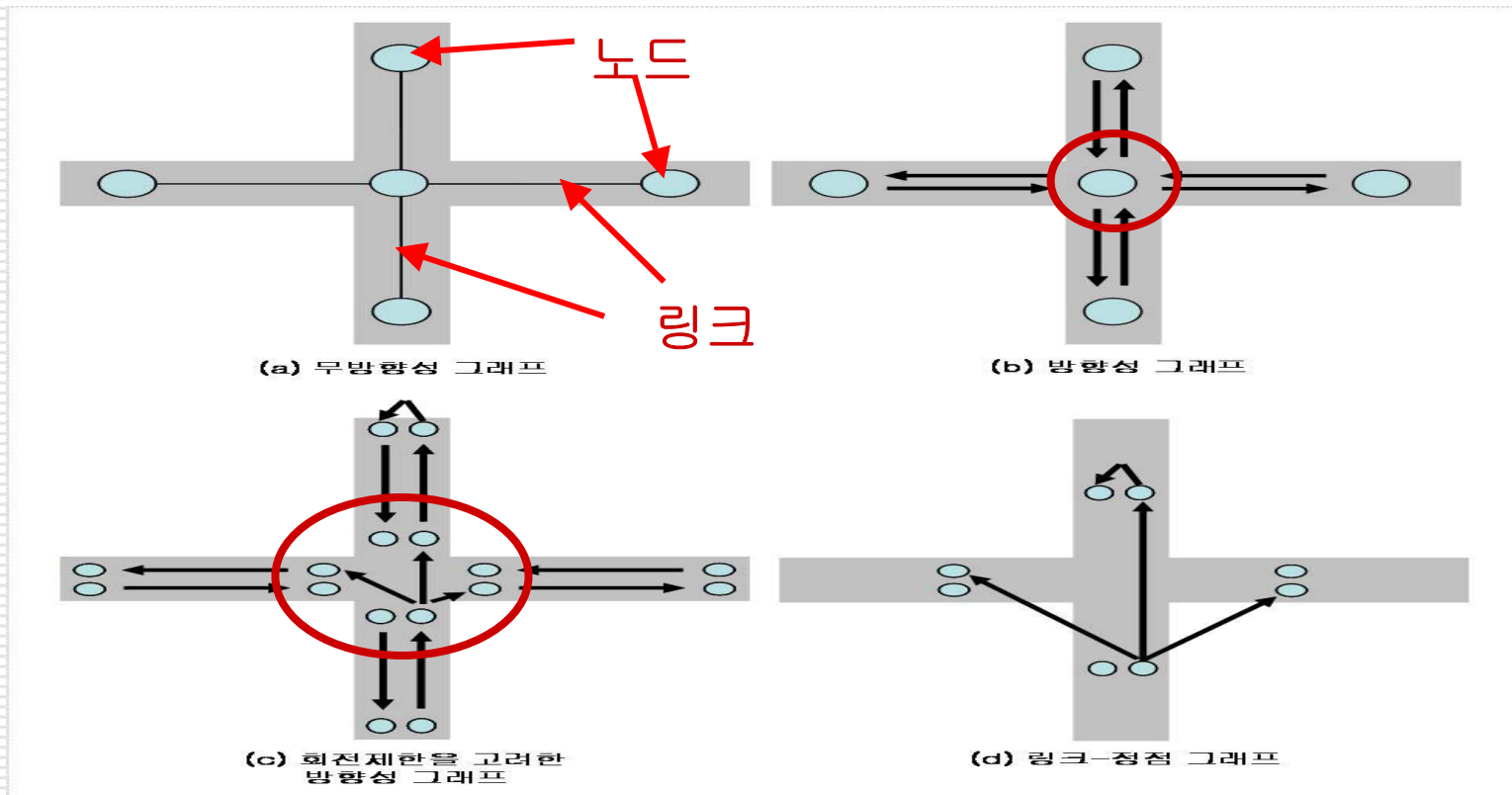
교통 분석을 위한 네트워크 표현의 예



Digital Orthophoto Image overlaid with Street Features

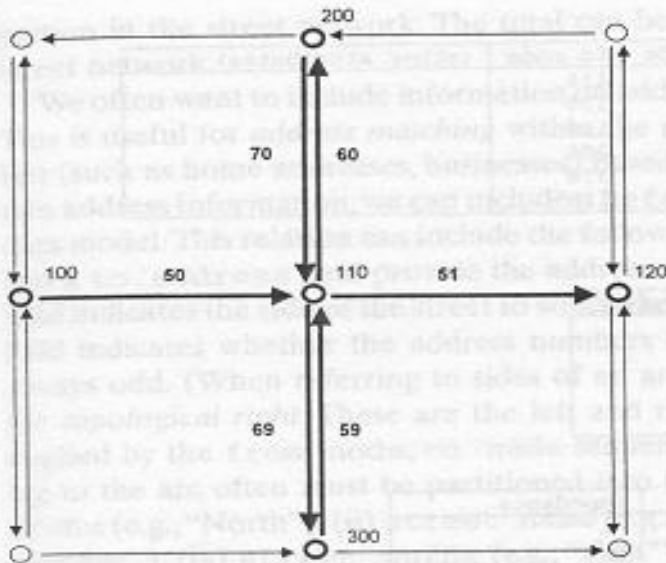
교통 네트워크의 추상화

□ 도시 교차로의 표현



교통 네트워크 데이터 모델과 구조 (1/4)

□ 기본 노드-링크 구조 ➔ 관계형 데이터 모델



50	Arc ID
100	Node ID

Arc			
arc id	from node	to node	(other attributes)
50	100	110	
51	110	120	
59	300	110	
60	110	200	
69	110	300	
70	200	110	

Node	
node id	(attributes)
100	
110	
120	
200	
300	

Turn Table		
from arc	to arc	impedance
50	60	-1
50	51	1
50	69	2
70	69	1
70	51	3
59	60	1
59	51	2

예제 네트워크와 관계형 데이터 모델의 표현

교통 네트워크 데이터 모델과 구조 (2/4)

□ 기본 노드-링크 구조 → 관계형 데이터 모델

■ 기본 노드 테이블

NODE	x (optional in schematic)	y (optional in schematic)
1	58	100
2	0	87
3	56	96

■ 기본 링크 테이블

LINK	FROM-NODE	TO-NODE	SHAPE POINT COORDINATES (optional in schematic)
a	2	3	
b	1	3	
c	4	5	

■ 링크는 그 링크를 구성하는 노드들의 순서에 의한 방향성을 가짐

□ 링크 **a**는 노드 2에서 시작하여 노드 3으로 가는 방향성이 정의됨

□ 그 역방향 링크 (3 to 2)는 -a로 표현 가능.

교통 네트워크 데이터 모델과 구조 (3/4)

□ 기본 노드-링크 구조

■ 노드 속성 테이블(Node Attribute Table)

NODE	VALENCE	LIST OF NODES	LIST OF LINKS
3	3	1, 2, 5	-b, -a, d
5	5	3, 4, 6, 10, 11	-d, -c, e, f, g
12	4	7, 11, 13, 18	-h, -m, n, s

□ **Valence**: incident links (or nodes) forming intersections.

■ 링크 속성 테이블(Link Attribute Table)

LINK	STREET NAME	ADDRESSES (L)	ADDRESSES (R)	CLASS	LENGTH (km)	LANES	SPEED LIMIT (km/h)
c	El Camino Real=Hwy101	—	—	Freeway	1.42	3	100
-c	El Camino Real=Hwy101	—	—	Freeway	1.44	3	100
m	Hollister Ave	1201-1299	1200-1298	Arterial	0.23	2	80
t	Walnut Lane	598-200	599-201	Residential	0.68	1	45

교통 네트워크 데이터 모델과 구조 (4/4)

□ 기본 노드-링크 구조

■ 관련 회전정보(turn) 테이블

□ *From-Link*와 *To-Link*가 레코드의 *key*가 됨

FROM-LINK	TO-LINK	IMPEDANCE
a	-b	2.8
b	-a	1.5
c	f	99999*

□ 매우 큰 저항값(impedance)은 회전 불가능을 표현

기본 노드-링크 구조의 문제점

- 모든 링크의 교차점들에서 노드가 존재하게 됨
 - Underpass나 Overpass 같은 현실세계의 교통 네트워크 특징들을 반영할 수 없음
 - 단일 링크는 균질(homogeneous)하게만 표현됨
 - 속성(제한속도, 차로수, 포장재질 등)이 변하는 부분을 표현할 수 없음
 - 즉, 교통 개체들간의 일대다(one-to-many) 관계를 지원하지 못함
 - 데이터베이스의 규모가 커지게 됨 (small problem)
 - 유지관리 (big problem)
 - 다른 기관이나 사용자들과의 데이터 공유가 어려워지게 됨
 - 동일 지형지물에 대한 다중 참조(multiple references)는 데이터 변경 및 불일치를 유발하게 됨
-

문제 해결의 대안 (1/2)

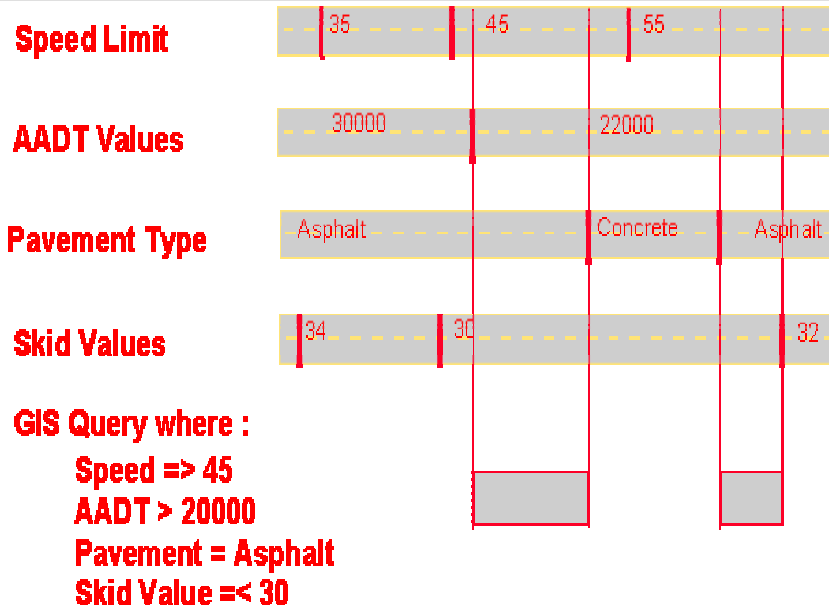
□ *Dynamic Segmentation* (동적 분할)

- 기존 네트워크 자료구조를 변화하지 않고 동적으로 기존의 링크 위주의 데이터에 이벤트(event) 데이터를 등록시키고 이를 기반으로 섹션(section)을 구성하여 이러한 섹션의 포함관계를 기존의 링크와 대응시키는 데이터 모델링 기법
 - 테이블에 저장된 선형 참조 데이터(event)를 지도에 디스플레이 할 수 있는 feature로 변환하는 과정
 - Applications
 - 사고분석, 교통량 분석, 혼잡관리, 포장 및 파이프라인 관리, 교통체계 유지 및 계획 등
 - 상용 소프트웨어의 예
 - ESRI – Dynamic Segmentation Module
 - TransCAD – Dynamic Segmentation
 - Intergraph – Segment Manager
-

문제 해결의 대안 (2/2)

□ *Dynamic Segmentation* (동적 분할)

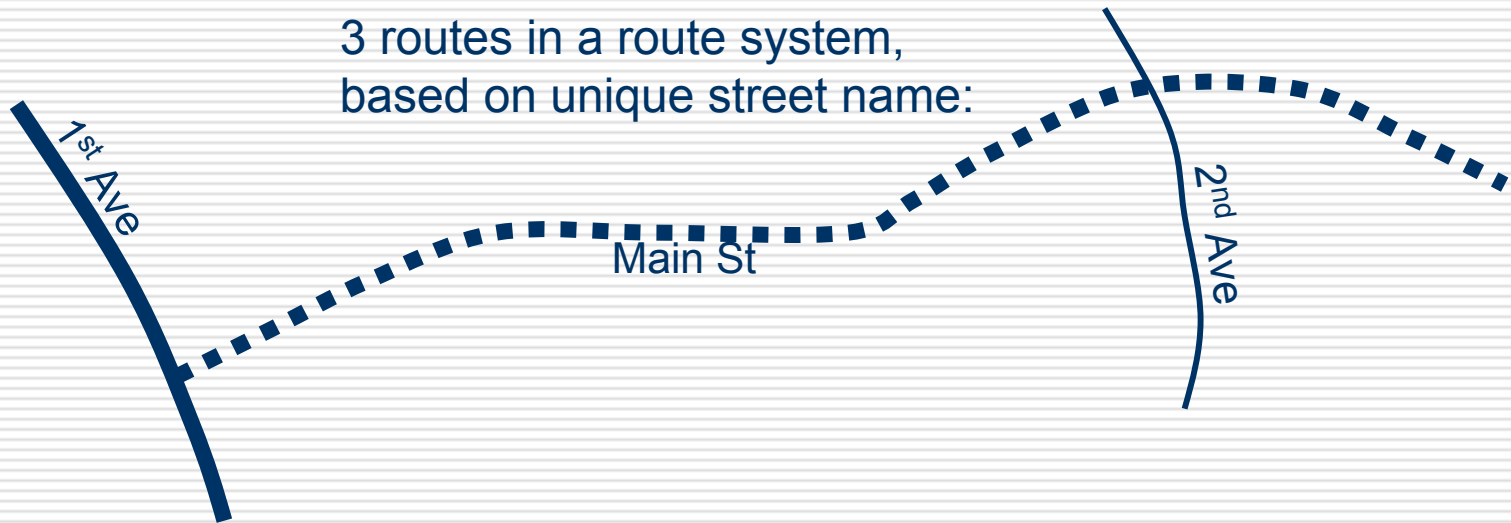
- 기본 요소들: routes, sections, 및 events
- *Route*는 section들의 집합
- *Section*은 라인 커버리지의 아크들에 직접적으로 연관되며 Route system에 척도를 제공
- *Events*는 route system에서 발생하는 일련의 사건



Dynamic Segmentation의 이해 (1/7)

□ Routes

- An aggregation of connected network segments that have a common attribute value
- Routes store “measure” (i.e. distance) values along their length



Dynamic Segmentation의 이해 (2/7)

□ Linear Referencing Systems (LRS)

- System of determining the position of an **event (i.e. pavement condition)** relative to a **route system (i.e. streets)** by specifying a **start position**, **direction** and **distance**

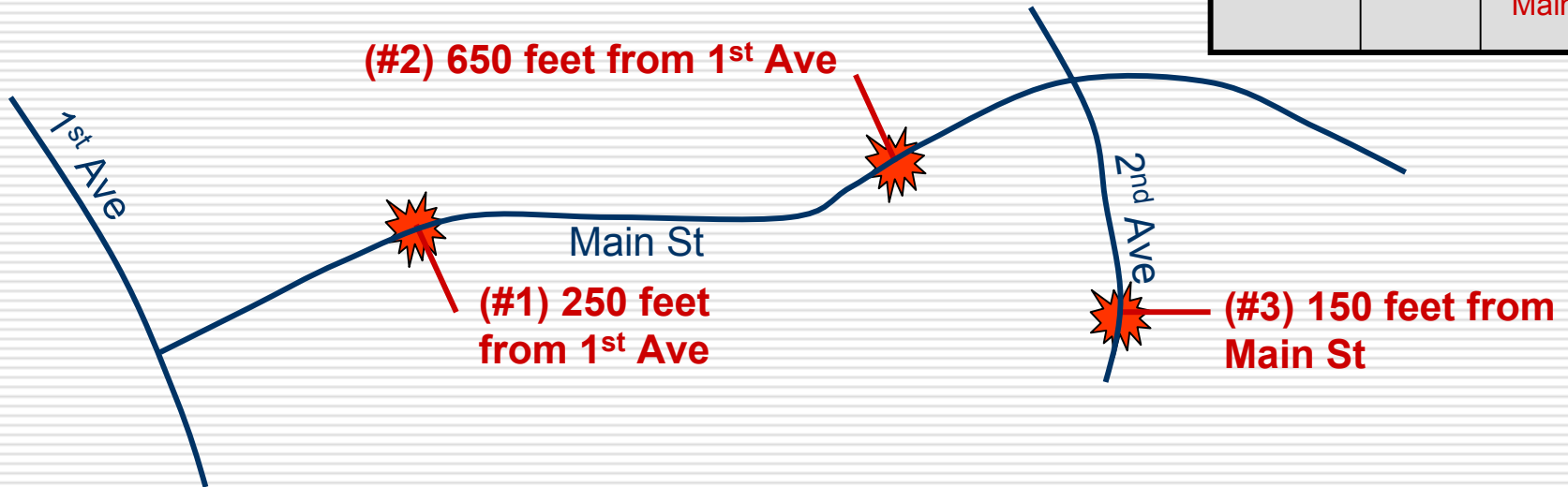


Dynamic Segmentation의 이해

□ Dynamic Segmentation

- A method of locating events in a **table** "on the fly", using the measure values of a **route system**

Accident ID #	On Street	Measure Value
1	Main St	250 feet From 1 st Ave
2	Main St	650 feet From 1 st Ave
3	2 nd Ave	150 feet From Main St



Dynamic Segmentation의 이해 (4/7)

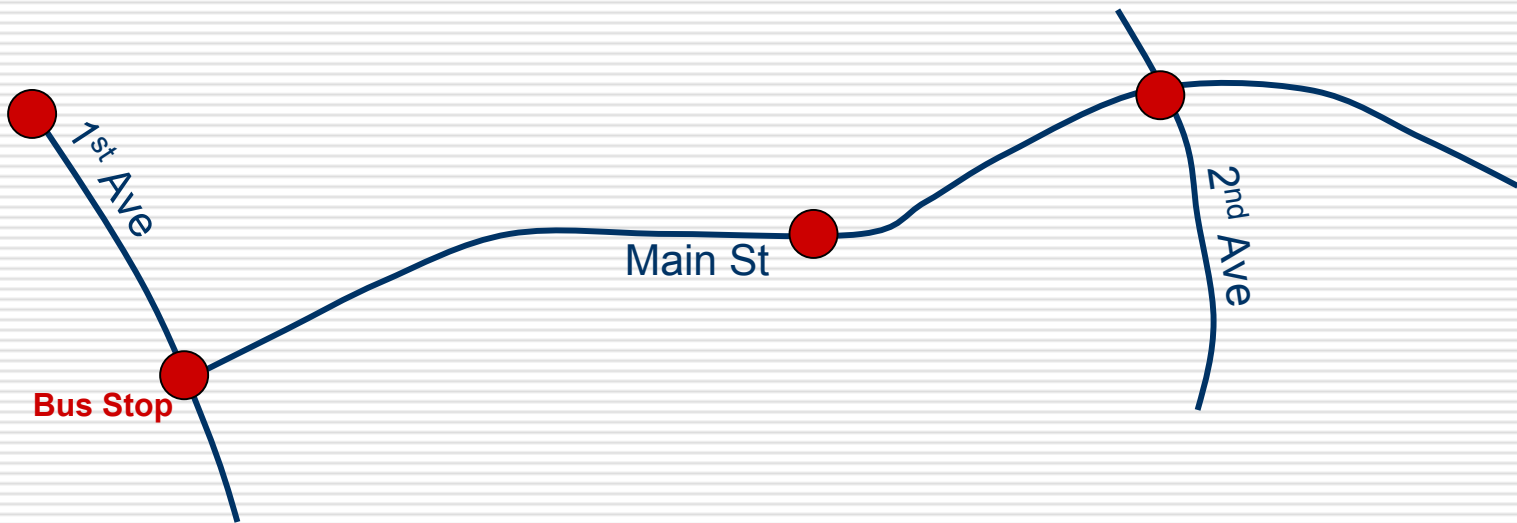
□ Dynamic Segmentation “Event” Types

- Dynamic segmentation “events” can be point features (i.e. accidents, call boxes, bus stops, etc.) or linear features (i.e. pavement condition, pavement type, etc.)

Dynamic Segmentation의 이해 (5/7)

□ Dynamic Segmentation "Event" Types

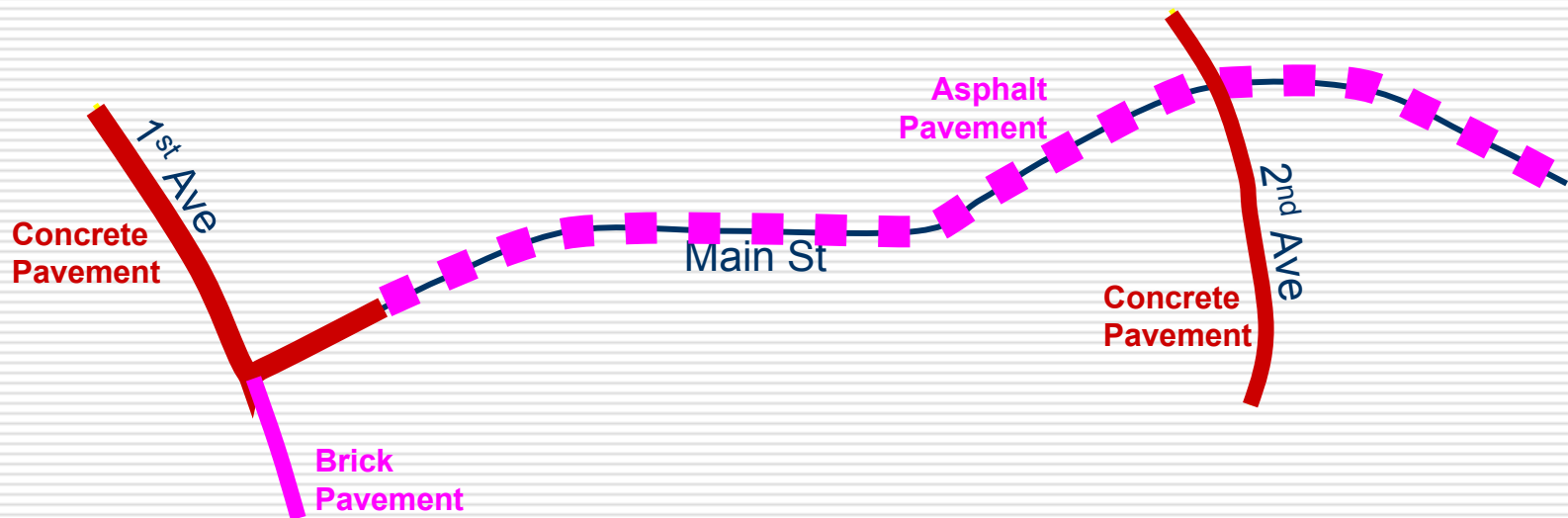
- Dynamic segmentation "events" can be point features (i.e. accidents, call boxes, **bus stops**, etc.) or linear features (i.e. pavement condition, pavement type, etc.)



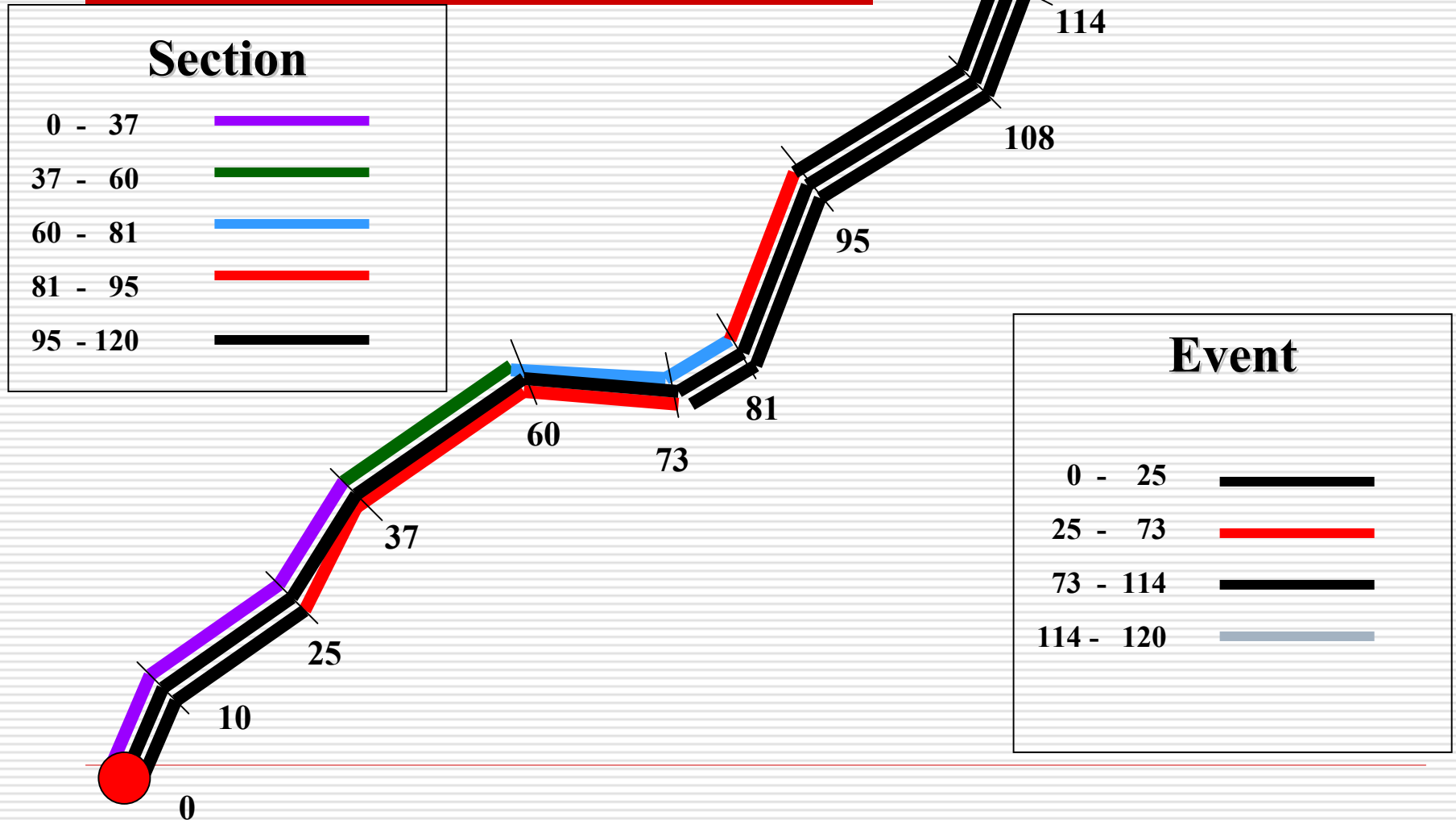
Dynamic Segmentation의 이해 (6/7)

□ Dynamic Segmentation “Event” Types

- Dynamic segmentation “events” can be point features (i.e. accidents, call boxes, bus stops, etc.) or linear features (i.e. pavement condition, **pavement type**, etc.)



Dynamic Segmentation의 이해



2. 네트워크 모델링

- GIS에서 네트워크 모델링
 - Geometric Network Model
 - 네트워크의 연결성
 - Geodatabase에서 네트워크 관련 객체들
 - Network Object Model
 - 분석을 위한 네트워크 구조 표현
-

GIS에서 네트워크 모델링 (1/3)

□ 사회기반시설(Infrastructure)

- 실세계 경제 활동의 토대가 되는 시설물들
- 네트워크로 표현되는 사회기반시설
 - 도로, 통신선로, 전력선로, 상·하수관로 등

□ ESRI사의 ArcGIS

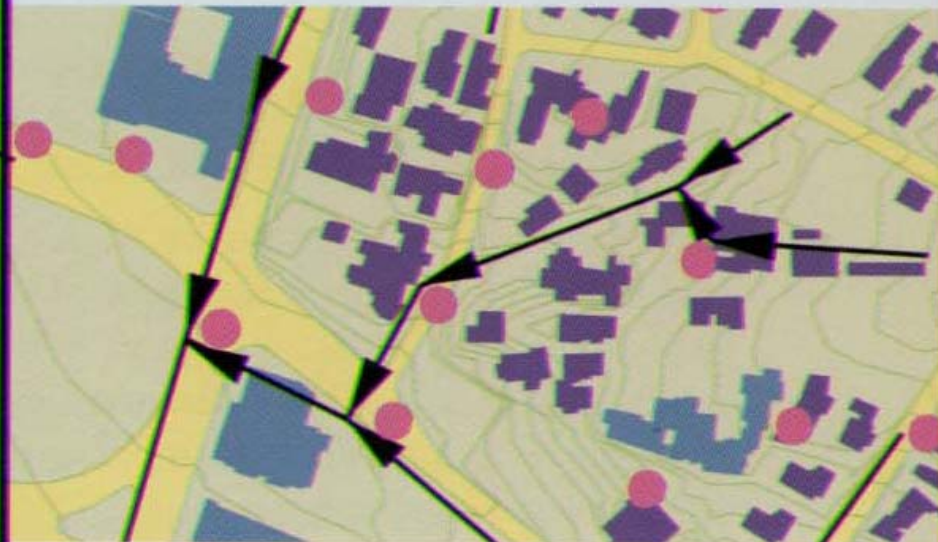
- Geometric Network model을 활용

□ 용어 정의

- 노드: node 또는 junction
 - 링크: link 또는 edge
 - 지형지물: feature
 - 이들은 서로 의미가 같은 것으로 정의함
-

GIS에서 네트워크 모델링 (2/3)

Utility networks



Some utility network tasks are:

- Establishing the direction of commodity flow
- Finding what is upstream of a point
- Closing switches or valves to redirect flow
- Identifying isolated parts of the network
- Finding facilities that serve a set of customers

Networks

You can subdivide networks into two broad categories: transportation and utility.

In a utility network, water and electricity are channeled until delivery to the customer.

In a transportation network, cars and trains are autonomous objects that can move freely.

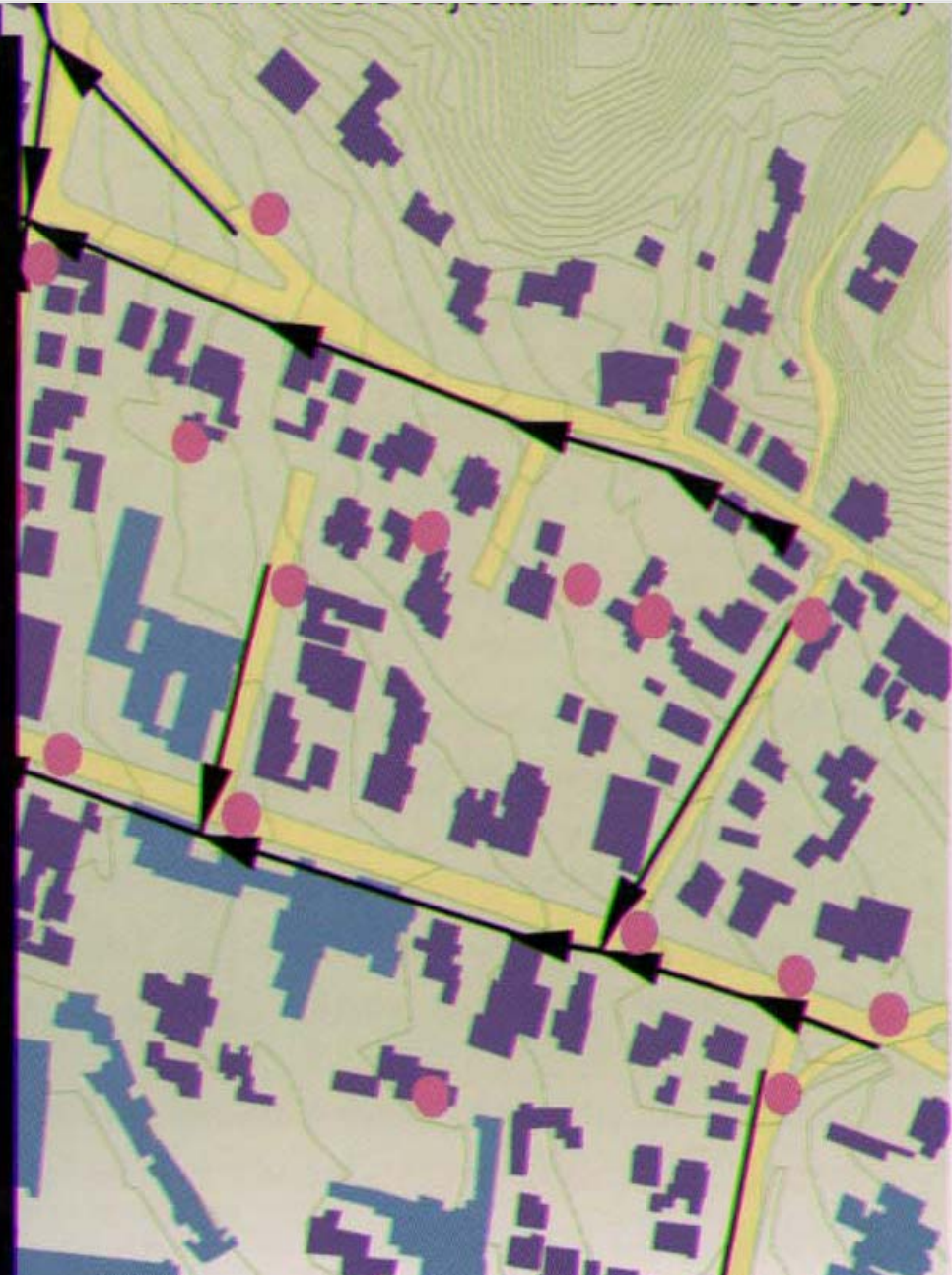
GIS에서 네트워크 모델링 (3/3)

Transportation networks



Some transportation network tasks are:

- Calculating the shortest path between points
- Determining a trade area based on travel time
- Dispatching the closest ambulance
- Finding the best sequence to visit customers
- Routing a garbage truck efficiently



Geometric Network Model (1/3)

□ 선형 시스템의 두 가지 표현

- The Geometric Network (기하학적 네트워크)
- The Logical Network (논리적 네트워크)

□ 기하학적 네트워크

- 노드와 링크가 서로 연결되어 구성된 feature들의 모음
- 하나의 링크는 두 개의 노드들로 구성되며, 하나의 노드는 여러 개의 링크를 가질 수 있음

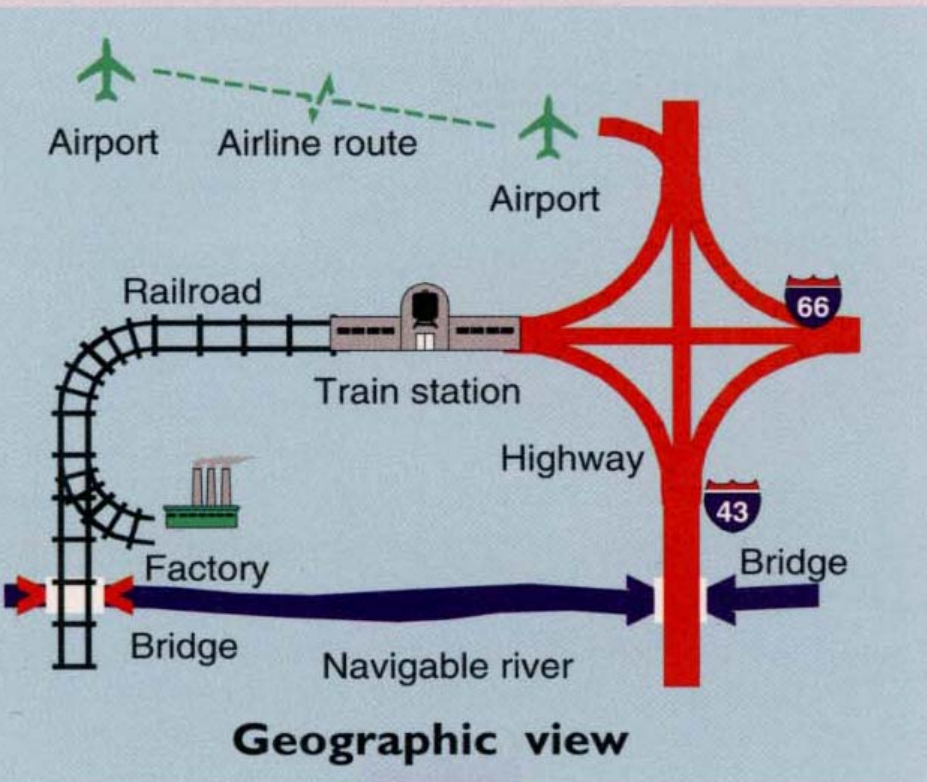
□ 논리적 네트워크

- Geometric network와 유사하게 노드와 링크로 구성
 - 그 차이는 좌표값을 포함하지 않는다는 것. 즉 기하학적 정보가 없음
 - 논리적 네트워크의 주요한 목적은 포함하고자 하는 어떤 속성값에 따라 연결성 정보를 저장하는 것임
 - 기하학적 네트워크 내 feature들과 논리적 네트워크 내 feature들 사이에는 1:1 또는 1:n의 관계가 존재함
-

Geometric Network Model (2/3)

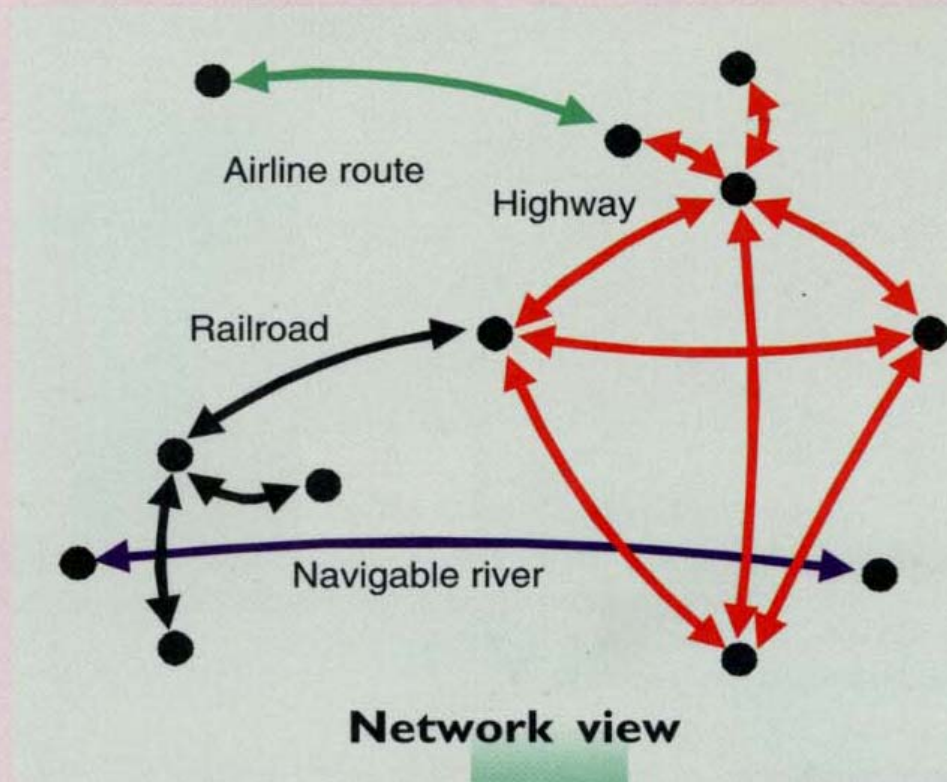
Two views of a network

You can view a network as a collection of geographic objects such as rails, roads, stations, and bridges and also as a pure network of edges and junctions.



A geometric network is the representation of geographic features that comprise a network.

Geometric network



A logical network is a pure graph of junction elements and edge elements.

Logical network

Geometric Network Model (3/3)

Geometric network

The geometric network maintains relationships between connected junction features and edge features. When you move a junction feature, the connected edge features are rubberbanded.

Railroads

id	geometry

Edge
feature
classes

Roads

id	geometry

A network feature can be related to one or many network elements. This allows a single feature to represent a complex part of a network.

Train stations

id	geometry

Junction
feature
classes

Rivers

id	geometry

Features

Features from an arbitrary number of edge and junction feature classes correspond to network elements in the edge table and junction table.

Airports

id	geometry

Network features can be organized in any number of network feature classes.

Logical network

Network elements are stored in an edge table and junction table with a connectivity table describing how they connect.

A logical network has no geometry or coordinates. It is a pure graph of how junctions and edges are connected.

Edges

Logical
network
tables

Connectivity

	junction IDs	adjacent junctions and edges

The connectivity table keeps track of how edge and junction elements are connected.

Junctions

The logical network is invisible in ArcMap and ArcCatalog, but it is the foundation for the geometric network's rich model and high performance.

You interact with the network through network features. When you add or remove a network feature in a geometric network, ArcInfo adds or removes the matching network elements. When you perform network analysis, ArcInfo submits a solver to the logical network.

The geometric network and logical network are always synchronized.

네트워크의 연결성 (1/3)

Geometric network



Junction feature table

id	geometry
j123	
j124	
j125	
j126	

Edge feature table

id	geometry
e1	
e2	
e3	

A geometric network contains the features that participate in a network. Feature classes contain either edge features or junction features.

Logical network

Connectivity table

Junction	Adjacent junction and edge		
j123	j124, e1		
j124	j124, e1	j125, e2	j126, e3
j125	j124, e2		
j126	j124, e3		

A logical network contains the connectivity of the network. The connectivity table lists all the adjacent junctions to a given junction, along with the edge that connects them.

■ 논리적 네트워크

➤ 논리적 네트워크의 주요 특징은 연결 테이블(connectivity table)에 있으며, 이는 네트워크들이 어떻게 연결되어 있는가를 설명함

➤ 네트워크의 모든 노드에 대해, 연결 테이블은 인접한 노드와 링크들의 정보를 나열하고 있음

➤ 즉, 연결 테이블은 기하학적 네트워크가 어떻게 네트워크의 통합성을 유지하는가를 보여주고 있음

네트워크의 연결성 (2/3)

Truck routes

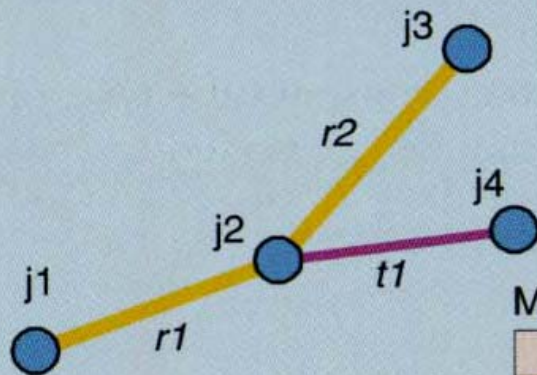
id	geometry
t1	

Cities

id	geometry
j1	
j2	
j3	
j4	

Major rails

id	geometry
r1	
r2	



A geometric network can have any number of participating feature classes. In this example, there is one junction feature class (Cities) and two edge feature classes that connect the junctions (Major rails and Truck routes).

Geometric Network

Junction element table

Feature Class	Feature ID	Element ID
1	j1	0
1	j2	1
1	j3	2
1	j4	3

Edge element table

Feature class	Feature ID	Element ID
2	r1	10
2	r2	11
3	t1	12

Connectivity table

Junction	Adjacent junction and edge elements		
0	1, 10		
1	0, 10	2, 11	3, 12
2	1, 11		
3	1, 12		

The logical network tracks feature IDs by feature class. For each feature class and feature ID combination, the logical network creates its own internal element ID.

Logical Network

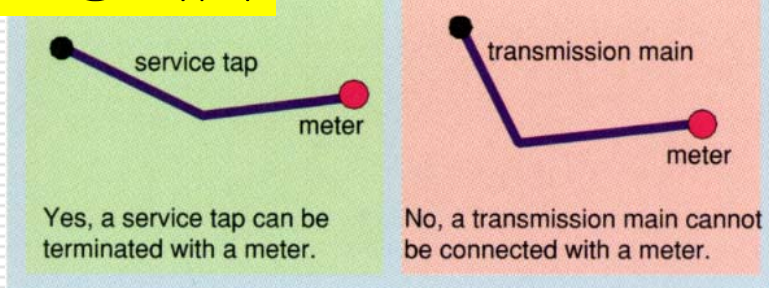
노드 원소 테이블과 링크 원소 테이블은 feature class와 feature의 조합으로 구성된 유일한 원소 ID를 제공함

네트워크의 연결성 (3/3)

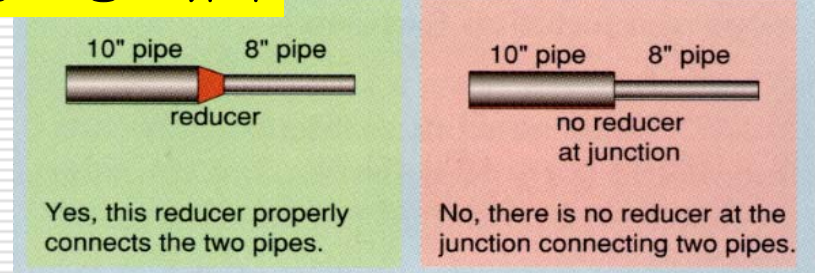
□ 연결 규칙(Connectivity Rule)

- a) 서로 연결될 수 있는 네트워크 feature들의 유형(type)과
b) 서로 다른 유형을 갖는 feature들이 연결될 수 있는 특정 유형의 feature들의 수를 정의
- 기하학적 네트워크 내에서 네트워크 feature들의 무결성(integrity) 유지를 용이하게 함
- 네트워크 feature들이 갖는 연결성 규칙의 예

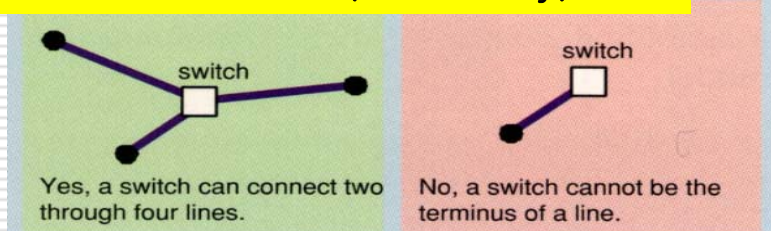
노드-링크 규칙



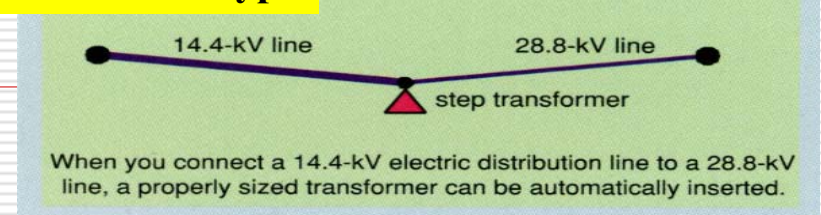
링크-링크 규칙



노드-링크 관계차수(cardinality) 규칙



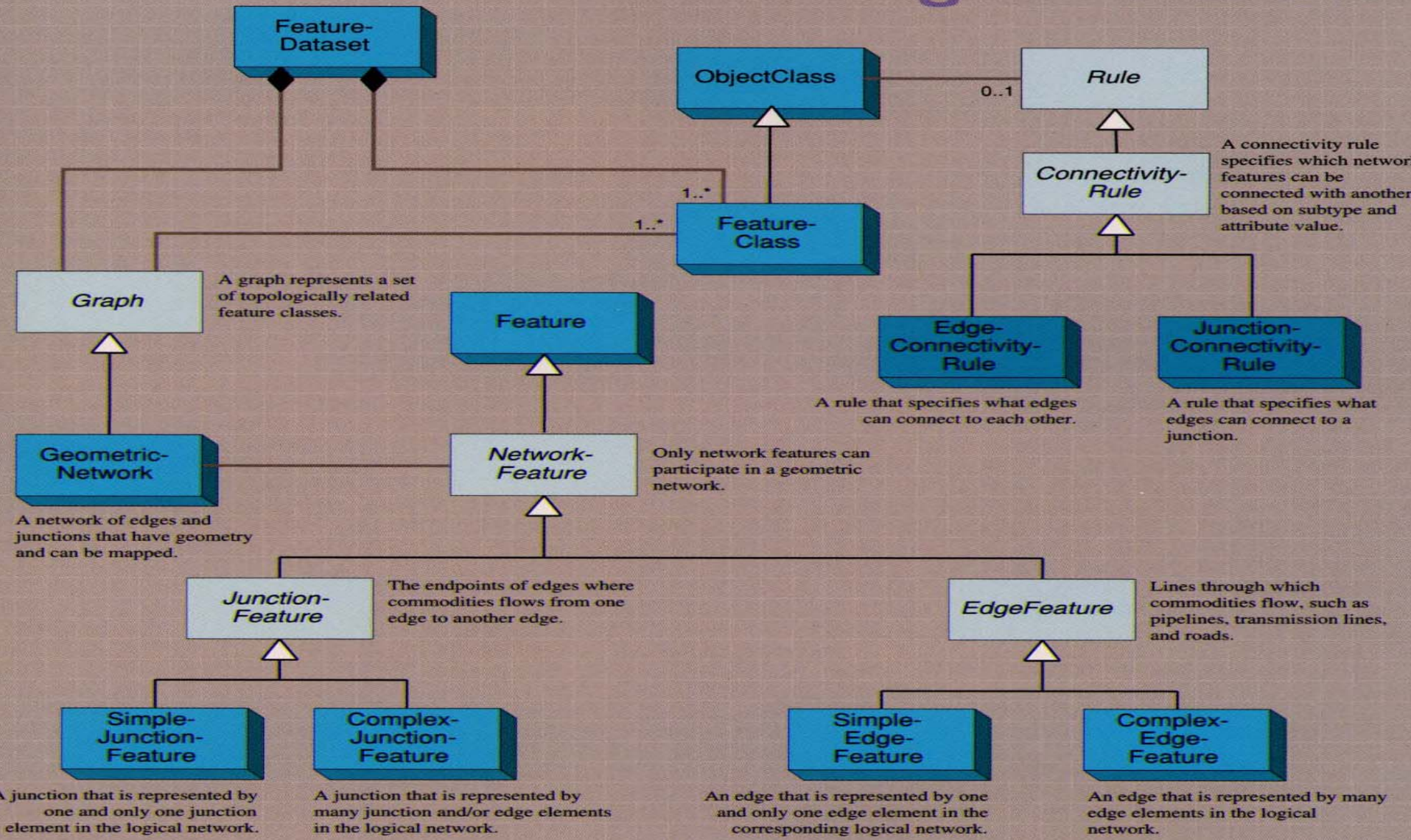
Default node type



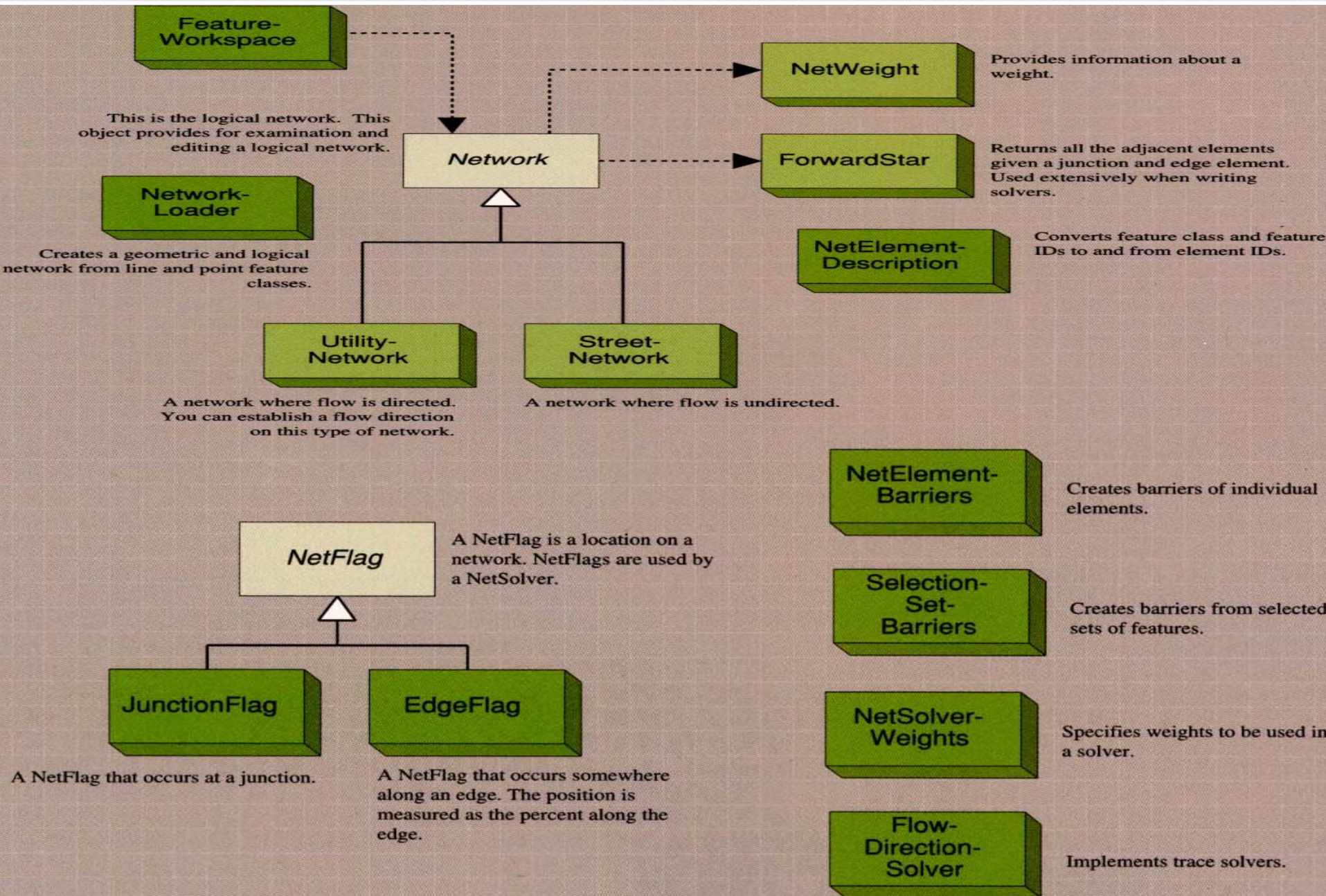
Geodatabase에서 네트워크 관련 객체들

This is a simplified UML diagram showing selected parts of the geodatabase data access objects that relate to geometric networks.

Network related objects in the geodatabase



네트워크 객체 모델



분석을 위한 네트워크 구조 표현 (1/7)

- 컴퓨터 메모리 내에서의 표현을 의미
 - 전산학의 자료구조(data structure) 과목에서 심도 있게 다루는 분야
 - 어떤 네트워크 표현 구조를 갖느냐에 따라 네트워크 분석의 수행 성능에 큰 차이를 보이게 됨
 - 즉, 최단경로 알고리즘 등을 이용한 분석의 수행 성능은
 - 그 자체의 알고리즘뿐만 아니라
 - 컴퓨터 내에서 사용되는 네트워크 표현 방식과
 - 계산의 중간 결과들을 관리하고 갱신하는데 사용하는 저장 구조에 직접적 영향을 받게 됨
-

분석을 위한 네트워크 구조 표현 (2/7)

□ 두 가지 유형의 정보가 저장·관리됨

- 네트워크의 노드와 아크 구조를 표현하는 네트워크 위상관계
- 네트워크의 노드와 아크에 관련된 시간, 비용, 용량 및 거리 등의 속성

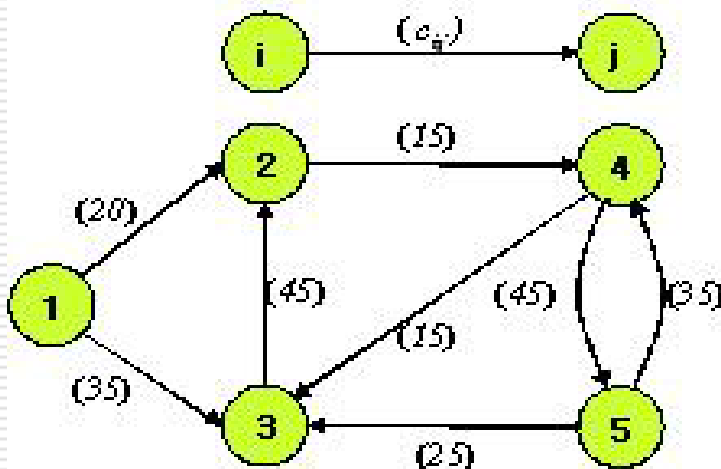
□ 네트워크 구조 표현의 네 가지 방법

- 노드-아크 접속행렬(node-arc incidence matrix)
 - 노드-노드 인접행렬(node-node adjacency matrix)
 - 인접 리스트(adjacency lists)
 - 포워드 및 백워드 스타 표현(forward and reverse star representations)
-

분석을 위한 네트워크 구조 표현 (3/7)

□ 노드-아크 접속 행렬(Node-Arc Incidence Matrix)

- 네트워크를 행렬에 저장하는데, 이때 각 행은 네트워크의 노드 정보를 포함하고 각 열은 아크 정보를 포함
- 아크 (i, j) 에 해당하는 열은 두 개의 영이 아닌(nonzero) 원소들만으로 구성되는데, 노드 i 에 해당하는 행에는 +1 그리고 노드 j 에 해당하는 행에는 -1의 계수값을 기록
- 그러나 행렬 내에 영(zero)인 계수값을 갖는 원소들이 너무 많게 되어 메모리 공간의 활용 측면에서 비효율적



(a)

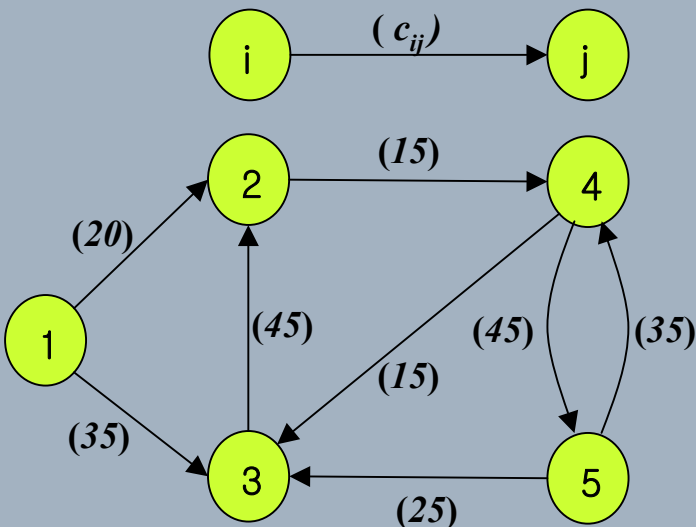
	(1,2)	(1,3)	(2,4)	(2,3)	(3,4)	(3,5)	(4,5)
1	1	1	0	0	0	0	0
2	-1	0	1	-1	0	0	0
3	0	-1	0	1	-1	0	0
4	0	0	-1	0	1	1	0
5	0	0	0	0	0	-1	1

(b)

분석을 위한 네트워크 구조 표현 (4/7)

□ 노드-노드 인접 행렬(Node-Node Adjacency Matrix)

- 각 노드에 해당하는 하나의 행과 열로 구성된 $n \times n$ 행렬로서 표현
- 아래 <그림>과 같이 각 행은 시작 노드를 나타내고, 각 열은 종점 노드를 나타내서 아크 (i, j) 에 해당하는 행과 열은 1의 계수값을 갖게 되고 다른 나머지는 모두 0의 계수값을 갖도록 표현
- 네트워크가 산재(sparse)적인 경우에는 저장 공간이 비효율적이지만, 충분히 밀집되어 있는 네트워크인 경우 메모리 공간을 효과적으로 절약 가능



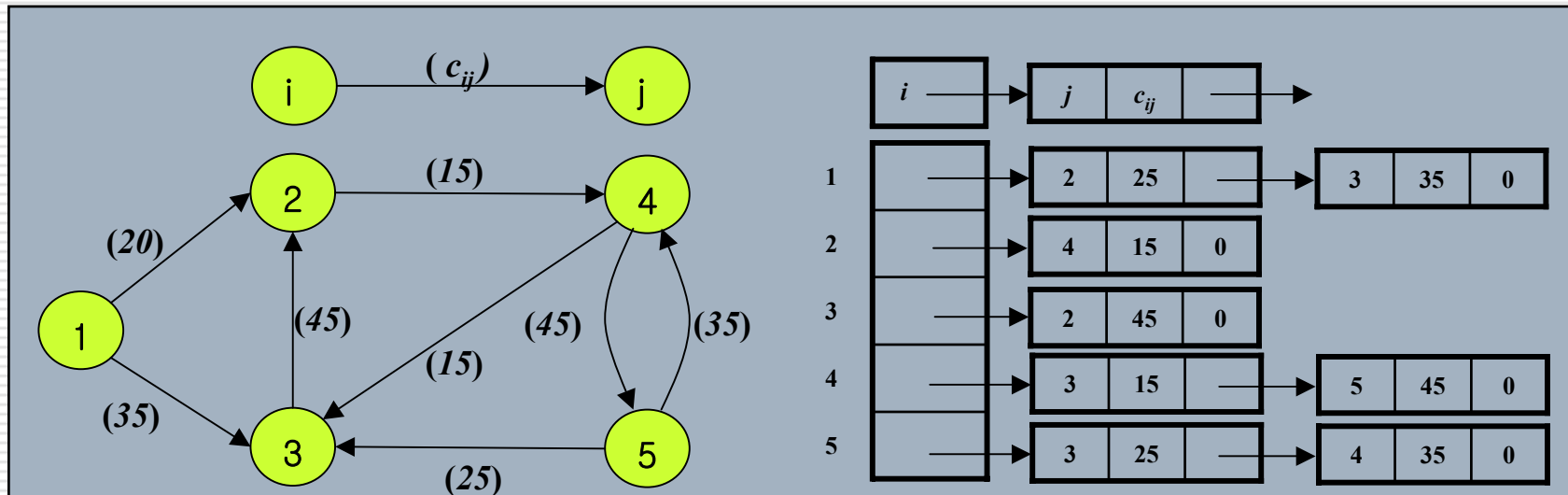
	1	2	3	4	5
1	0	1	1	0	0
2	0	0	0	1	0
3	0	1	0	0	0
4	0	0	1	0	1
5	0	0	1	1	0

	1	2	3	4	5
1	0	20	35	0	0
2	0	0	0	15	0
3	0	45	0	0	0
4	0	0	15	0	45
5	0	0	25	35	0

분석을 위한 네트워크 구조 표현(5/7)

□ 인접 리스트(Adjacency List)

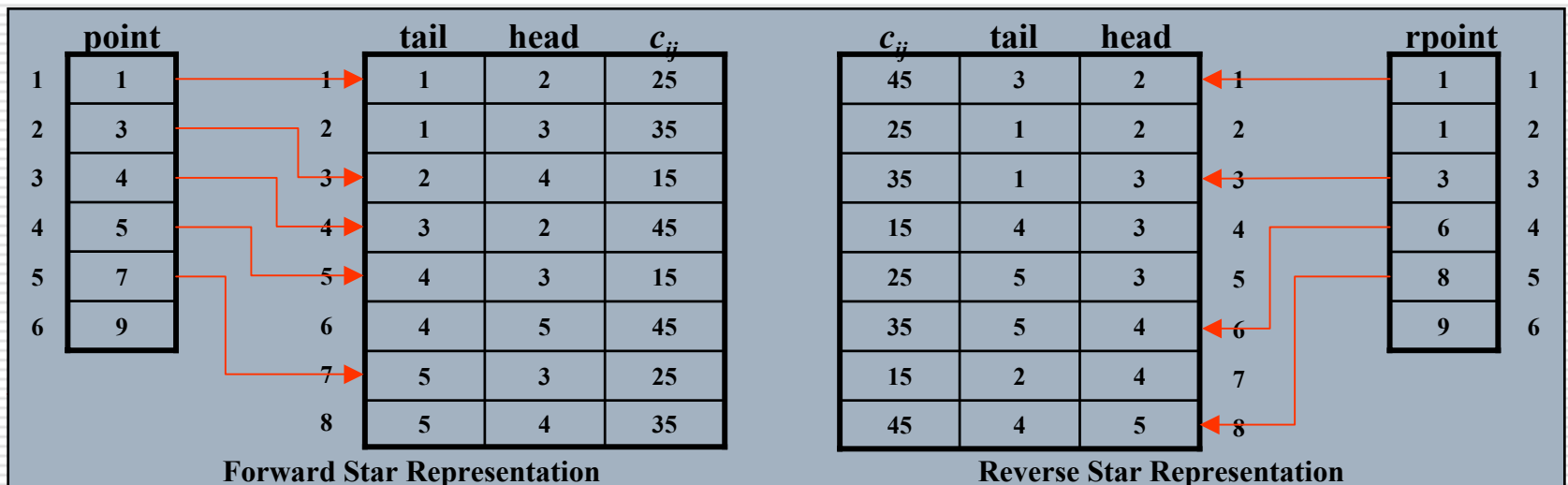
- 각 노드에 인접하고 있는 노드들의 리스트를 데이터 필드와 링크 필드로 구성된 단일 연결 리스트(singly linked list)에 저장
- 데이터 필드에는 그 노드에 관련된 정보들을 저장하고, 링크 필드에는 그 노드의 다음 노드를 가리키는 포인터를 저장
- 따라서 동일 노드에서 시작하는 모든 아크들은 함께 저장되며, 그 각각은 그 아크가 끝나는 노드와 거리 (또는 비용)만을 저장하도록 하는 방식
- 프로그래밍 언어를 이용한 구현이 간단하고, 아크와 노드 추가 및 삭제 등의 조작이 효율적이므로 많이 사용되는 방식



분석을 위한 네트워크 구조 표현(6/7)

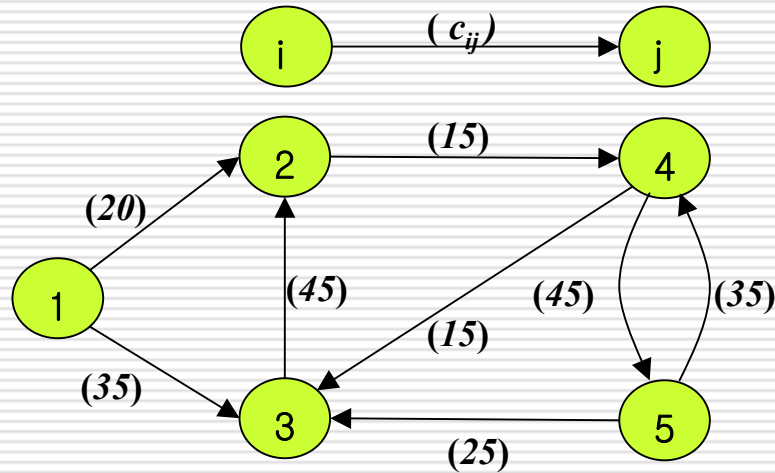
□ 포워드 스타 및 백워드 스타 표현(Forward and Reverse Star)

- 각 노드의 인접 노드 리스트를 연결 리스트가 아닌 일차원 배열에 저장하는 방식
- 포워드 스타 방식은 네트워크의 각 노드에서 나가는 아크들을 저장하므로 임의의 노드에서 나가는 아크들의 집합을 결정하는데 효율적
- 백워드 스타 방식은 각 노드로 들어오는 아크들을 저장
- 두 방식 모두 각 아크에 특정한 순서로 유일한 일련번호를 부여한 후, 각 아크를 아크 리스트에 관련된 정보와 함께 순차적으로 저장
- 공간 및 조작 효율성이 좋아 가장 많이 사용하는 방식



분석을 위한 네트워크 구조 표현 (7/7)

□ 포워드 스타 및 백워드 스타 표현(Forward and Reverse Star)



Forward Star Representation					Reverse Star Representation				
	point		tail	head	c_{ij}		tail	head	c_{ij}
1	1	1	1	2	25	45	3	2	1
2	3	2	1	3	35	25	1	2	2
3	4	3	2	4	15	35	1	3	3
4	5	4	3	2	45	15	4	3	4
5	7	5	4	3	15	25	5	3	5
6	9	6	4	5	45	35	5	4	6
		7	5	3	25	15	2	4	7
		8	5	4	35	45	4	5	8

3. 최단경로 탐색 알고리즘

- 최단경로 개요
 - 최단경로 정의
 - 최단경로 문제의 유형
 - 최단경로 문제의 해법
 - 최단경로 탐색 알고리즘
 - Dijkstra 알고리즘
 - 최단경로 탐색 알고리즘의 최대 병목지점
 - 최단경로 탐색 알고리즘 관련 연구
 - 최단경로 탐색 알고리즘 성능평가 실험
-

최단경로 개요

□ 최단경로 분석 (Shortest Path Analysis)

- 대부분의 네트워크 분석에서 매우 중요한 요소
 - 네트워크 흐름의 가장 두드러진 핵심 요소들을 담고 있어 더 복잡한 네트워크 모델을 연구하는 출발점이자 현실에서 빈번하게 요구되는 다양한 문제들을 해결하기 위한 핵심 모듈
 - 1950년대 이후 수 많은 연구와 알고리즘들이 발표되었음
 - 분석에 포함된 네트워크가 매우 클 경우, 매우 시간이 많이 소요되는 과정이 됨
 - 포함된 네트워크 내 아크와 노드의 수에 비례함
 - 따라서 가장 빠르고 효율적인 최단경로 알고리즘의 선택 과정은 네트워크 분석에 있어 매우 중요한 과정이 됨
-

최단경로의 정의 (1/2)

□ 네트워크의 정의

- $G = (N, A)$

- N : $n = |N|$ 인 노드들의 색인 집합, n = 노드들의 수

- A : $m = |A|$ 인 방향성 아크들의 집합, m = 아크들의 수

- 네트워크에서 아크의 표현 \rightarrow 방향을 갖는 노드들의 쌍

- (i, j) : 노드 i 에서 노드 j 로 가는 아크를 의미

□ 경로(path)의 정의

- 기점 노드에서 종점 노드까지 구성되는 방향성 아크들의 순열, a_1, a_2, \dots, a_n (for $i = 1, 2, \dots, n$)

- 경로의 길이 \rightarrow 그 경로를 구성하는 아크들의 길이의 총합

□ 최단경로(Shortest path)

- 기점 노드에서 종점 노드까지 여러 경로들 가운데 최소의 길이를 갖는 경로

최단경로의 정의 (2/2)

□ 가정(assumptions)

- 모든 아크 길이(비용)는 정수임
 - 네트워크는 노드 s 로 부터 다른 모든 노드까지 가는 방향성 경로를 포함함
 - 네트워크는 음의 사이클을 포함하지 않음(즉, 음의 길이를 갖는 방향성 사이클)
 - 네트워크는 방향성을 가짐
-

최단경로 문제의 유형

- 일-대-일(one-to-one)
 - 주어진 하나의 기점 노드 i 에서 다른 종점 노드 j 까지 하나의 최단경로를 탐색하는 문제
 - 일-대-다(one-to-all)
 - 주어진 하나의 기점 노드 i 에서 네트워크의 다른 모든 종점 노드들까지 최단경로를 탐색하는 문제
 - 다-대-일(all-to-one)
 - 일-대-다의 역
 - 다-대-다(all-to-all)
 - 네트워크 내 모든 노드에서 다른 모든 노드들까지의 최단경로 탐색
 - 기타 다양한 일반화 유형들이 존재 → one-to-some 등
-

최단경로 문제의 해법 (1/2)

□ 표지(labeling) 알고리즘

- 알고리즘의 각 반복 단계마다 각 노드에 하나의 표지를 할당하는 방식
- 표지의 종류

□ 영구표지(permanent label)

- 기점에서 임의의 특정 노드까지의 최단 거리가 확정된 노드에 부여하는 표지

□ 임시표지(temporarily label)

- 현재까지 계산된 경로가 최단경로가 아닐 수도 있는 경우에 부여하는 표지로 실제 거리의 최고상한값(upper bound)으로 표현

■ 표지 알고리즘의 종류

□ 표지-지정(label-setting) 알고리즘

□ 표지-수정(label-correcting) 알고리즘

최단경로 문제의 해법 (2/2)

□ 표지(labeling) 알고리즘

■ 표지-지정(label-setting) 알고리즘

□ 각 단계에서 하나의 표지씩 영구로 지정해 가는 방법

■ 표지-수정(label-correcting) 알고리즘

□ 최단경로가 결정되는 마지막 단계에서 모든 노드들이 영구 표지로 지정되며, 그 이전에는 임시표지로 관리하는 방법

■ 표지가 지정된 노드들을 관리하기 위한 자료구조와 표지 지정을 위해 조사할 노드들을 선택하는 방식에 큰 차이가 있음

최단경로 탐색 알고리즘

□ Bellman(1958)의 최적원리

- “최단경로를 구성하는 부분경로 역시 최단경로”

- 기점에서 특정 노드 j 까지 구성된 경로가 최단경로이면 이들 사이에 존재하는 모든 부분경로는 최단경로가 됨을 의미

□ Dijkstra 알고리즘

- 비음의 길이를 갖는 네트워크의 시작노드로부터 다른 모든 노드로의 최단경로 탐색에 가장 많이 활용
 - 표지지정 방식
 - 최소의 임시표지를 갖는 노드를 선택하여 그 노드를 영구표지로 지정하고, 그 노드에 인접하고 있는 노드들의 거리표지를 갱신하기 위해 점점 확장해 나가는 방법
-

Dijkstra algorithm (1/16)

□ 개요

- 알고리즘은 기점 노드에서부터 시작
 - 이전 노드로부터 누적 통행시간이 가장 낮은 노드를 찾으면, 이 노드는 Reached 상태가 됨
 - 노드 A의 경우, 이전 노드가 없으므로 이미 Reached 상태가 되고, 따라서 Reached Table에 기록
 - 그런 후, 알고리즘은 마지막으로 reached된 노드에 인접한 모든 노드들을 고려
 - 이때 이 인접 노드들은 SCANNED Table에 기록
 - 알고리즘의 각 단계에서, SCANNED Table 내에서 가장 작은 누적 통행시간을 갖는 노드는 SCANNED Table에서 삭제한 후, Reached Table에 기록
 - 이러한 과정을 종점 노드에 도달할 때까지 반복
-

Dijkstra algorithm (2/16)

□ 예제 교통 네트워크

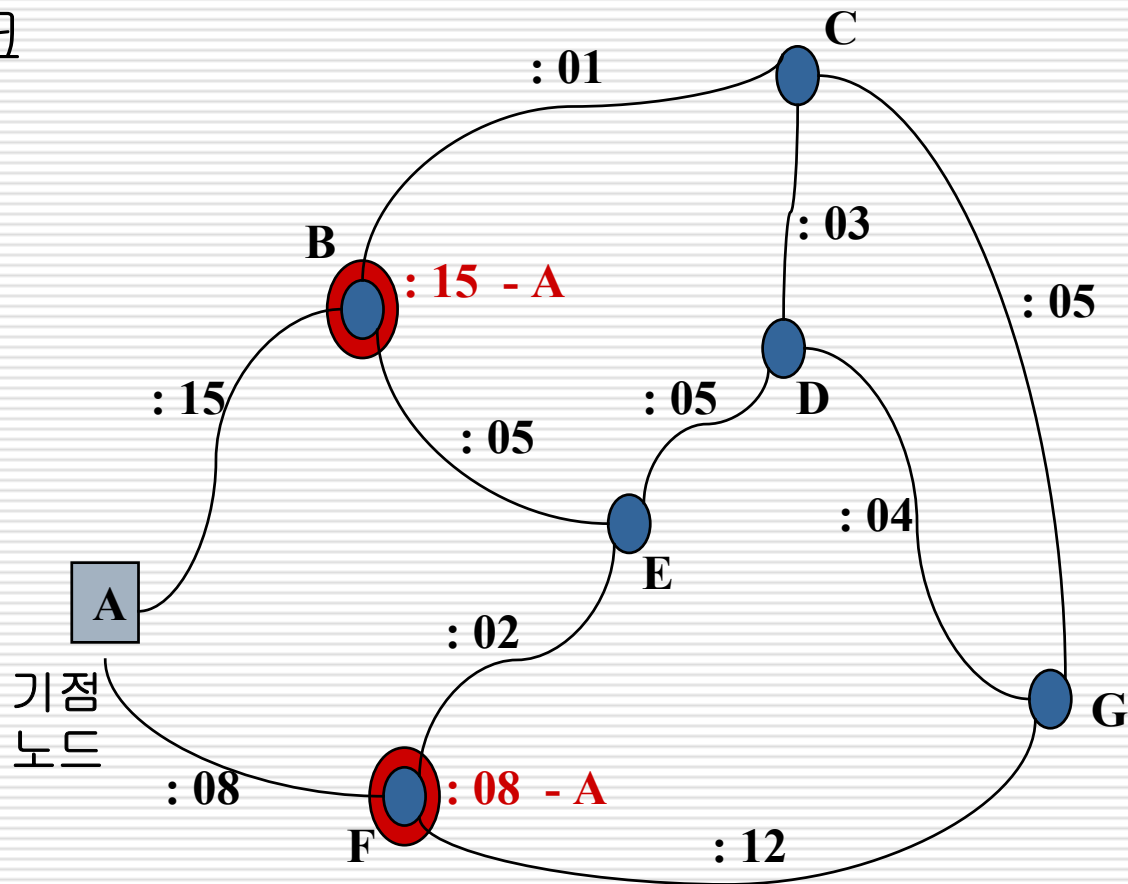
● 검사되지 않은(Unscanned) 노드

● 검사된(Scanned) 노드

■ A 도달된(Reached) 노드

: 08 링크를 통과하는 고정비용, 시간

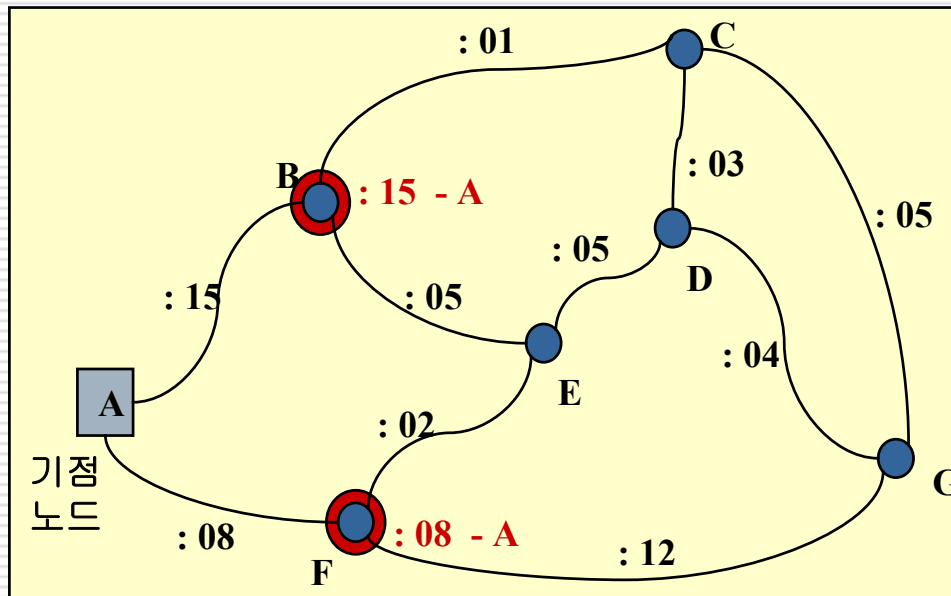
: 15 - A 누적 비용과 이전 노드



Dijkstra algorithm (3/16)

□ Step 1

- 기점 노드를 도달상태로 지정하고 Reached Table에 기점 노드를 기록
- 인접한 노드들을 검사하고 검사된 노드들은 Scanned Table에 기록



REACHED TABLE

NODE	CUMULATIVE COST	PREVIOUS NODE
A	:00	none
B		
C		
D		
E		
F		
G		

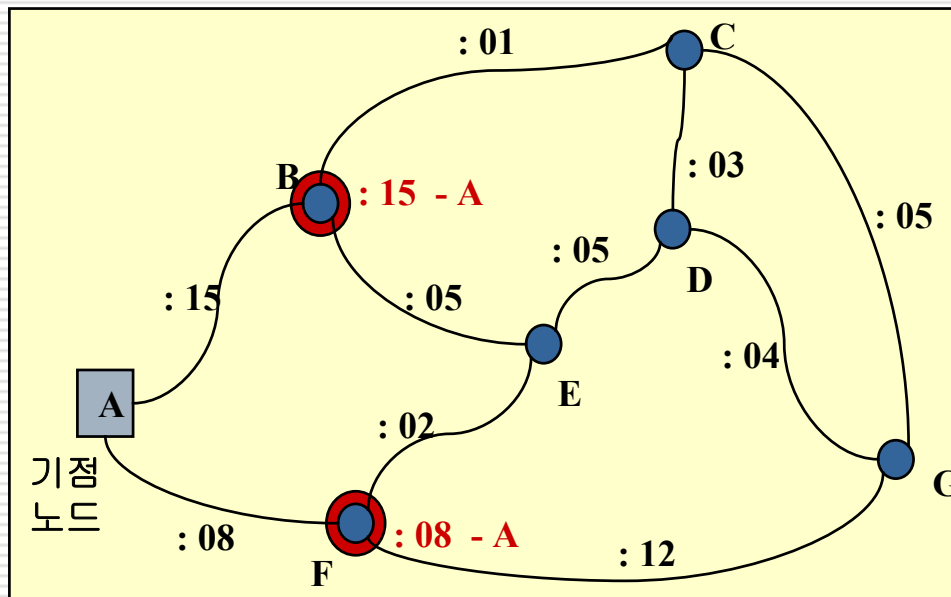
SCANNED TABLE

NODE	CUMULATIVE COST	PREVIOUS NODE
B	:15	A
F	:08	A

Dijkstra algorithm (4/16)

□ Step 2

- 가장 작은 누적비용을 갖는 검사된 노드를 선택하여 Reached Table로 이동



REACHED TABLE

NODE	CUMULATIVE COST	PREVIOUS NODE
A	:00	none
B		
C		
D		
E		
F	:08	A
G		

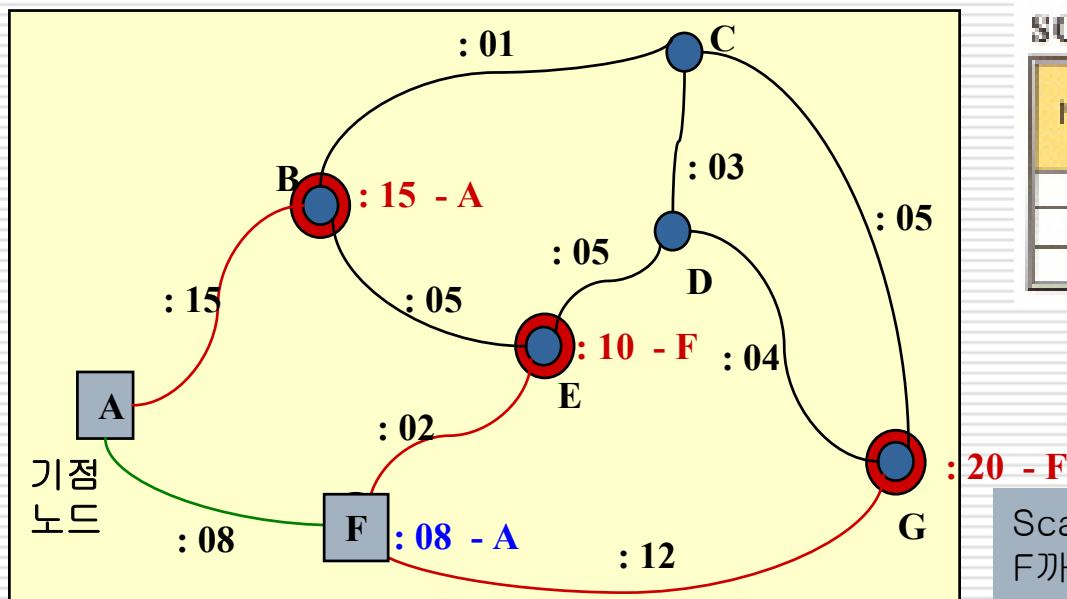
SCANNED TABLE

NODE	CUMULATIVE COST	PREVIOUS NODE
B	:15	A
F	:08	A

Dijkstra algorithm (5/16)

Step 3

- 이제 막 도달된 상태의 노드(F)에 인접한 노드들(E, G, A)을 검사하고, 이들을 Scanned Table에 기록한 후, F는 도달상태로 기록
 - A는 이미 도달상태이므로 제외



SCANNED TABLE

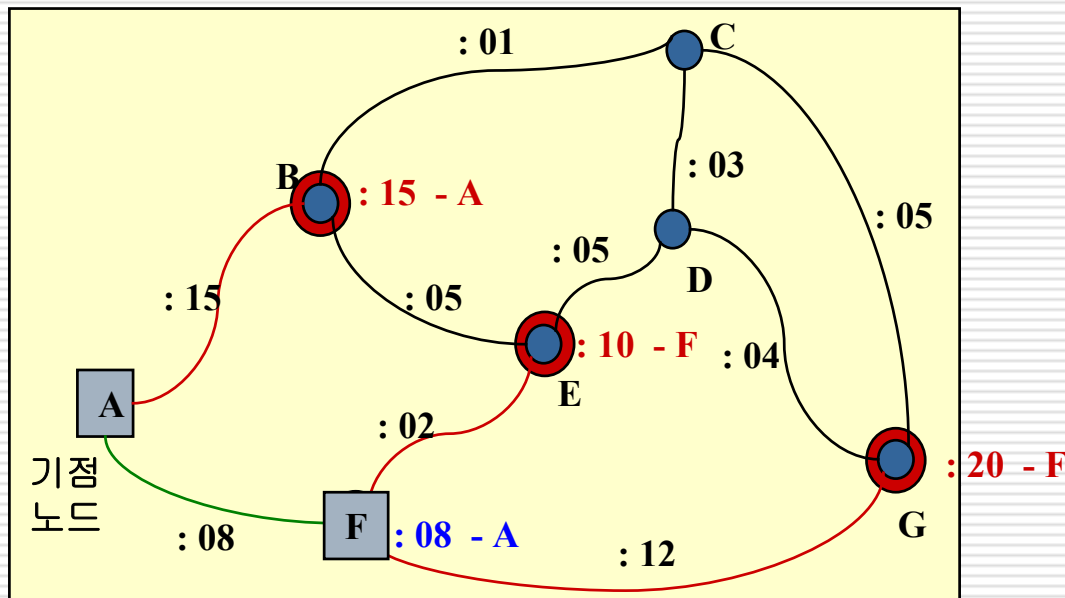
NODE	CUMULATIVE COST	PREVIOUS NODE
B	:15	A
E	:10	F
G	:20	F

Scanned Table 내 각 노드의 누적비용은 F까지 도달하기 위한 비용(8) + 노드 E(2)와 F까지 도달하기 위한 비용(8) + 노드 G(12)임

Dijkstra algorithm (6/16)

Step 4

- Scanned Table에서 가장 작은 누적비용을 갖는 검사된 노드(E)를 선택하고 그것을 Reached Table로 이동



REACHED TABLE

NODE	CUMULATIVE COST	PREVIOUS NODE
A	:00	none
B		
C		
D		
E	:10	F
F	:08	A
G		

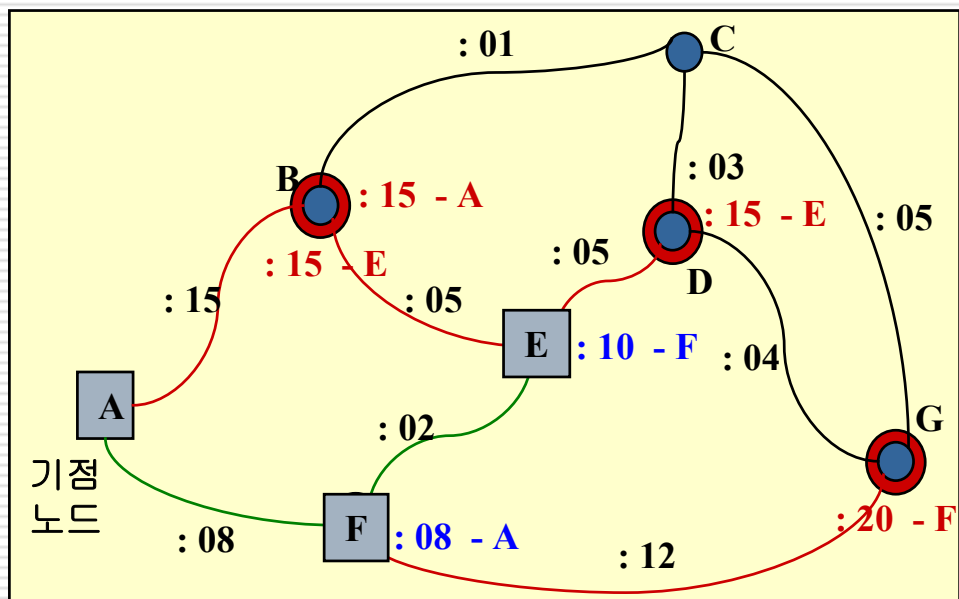
SCANNED TABLE

NODE	CUMULATIVE COST	PREVIOUS NODE
B	:15	A
E	:10	F
G	:20	F

Dijkstra algorithm (7/16)

Step 5

- 이제 막 도달된 상태의 노드(E)에 인접한 노드들(B, D, F)을 검사하고, 이들을 Scanned Table에 기록한 후, E는 도달상태로 기록
- F는 이미 도달상태이므로 제외



SCANNED TABLE

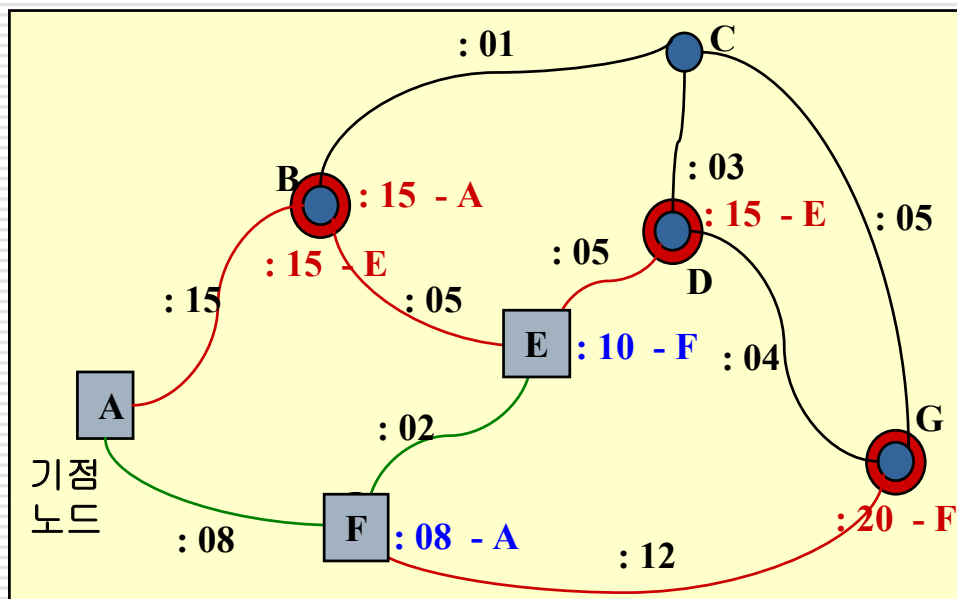
NODE	CUMULATIVE COST	PREVIOUS NODE
B	:15	A
G	:20	F
B	:15	E
D	:15	E

노드 B는 현재 2번 검사되었음 (A와 E로 부터).
누적비용은 두 경우 모두 15로 같음.
노드 D도 역시 15의 누적비용을 가짐.
따라서 알고리즘은 임의로 이들 셋 가운데 하나를 선택하게 됨.

Dijkstra algorithm (8/16)

□ Step 6

- Scanned Table에서 가장 작은 누적비용을 갖는 검사된 노드를 선택하고 그것을 Reached Table로 이동 – 알고리즘이 임의로 노드 D를 선택했다고 가정



REACHED TABLE

NODE	CUMULATIVE COST	PREVIOUS NODE
A	00	none
B		
C		
D	15	E
E	10	F
F	08	A
G		

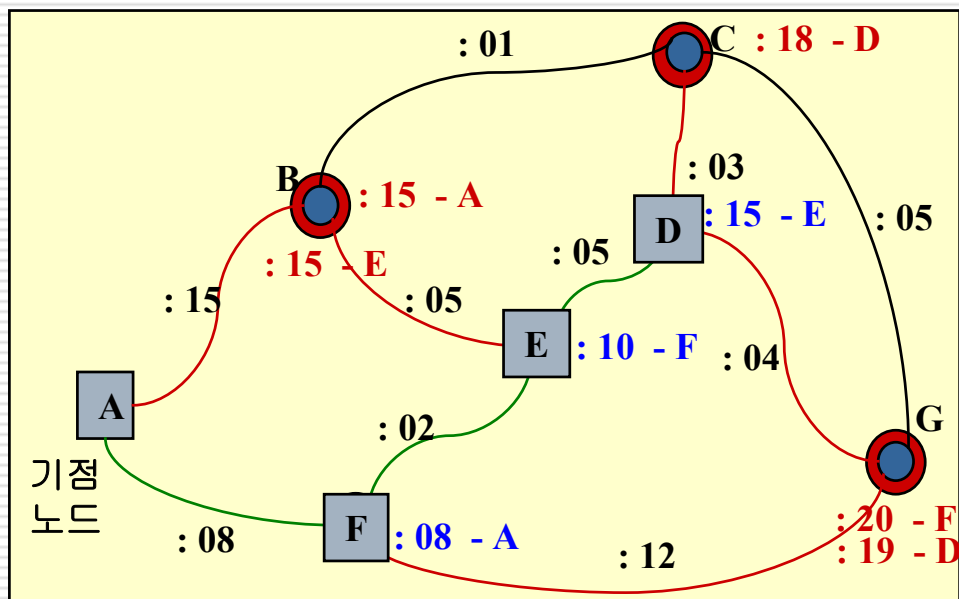
SCANNED TABLE

NODE	CUMULATIVE COST	PREVIOUS NODE
B	15	A
G	20	F
B	15	E
D	15	E

Dijkstra algorithm (9/16)

□ Step 7

- 이제 막 도달된 상태의 노드(D)에 인접한 노드들(C, E, G)을 검사하고, 이들을 Scanned Table에 기록한 후, D는 도달상태로 기록 - E는 이미 도달상태이므로 제외



SCANNED TABLE

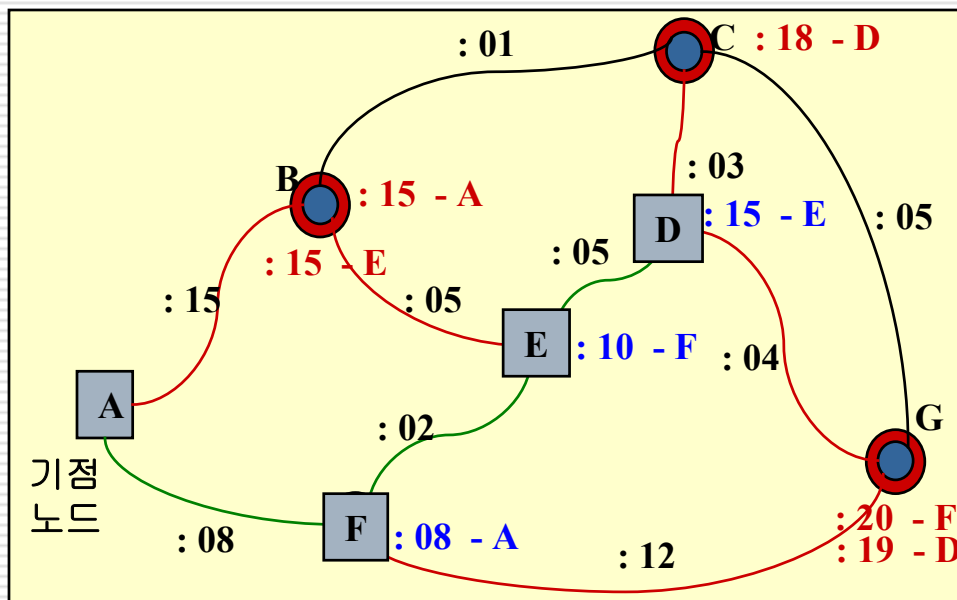
NODE	CUMULATIVE COST	PREVIOUS NODE
B	:15	A
G	:20	F
B	:15	E
C	:18	D
G	:19	D

노드 G는 현재 2번 검사되었음 (D와 F로 부터).
누적비용은 D로 부터 오는 경우 19, F로 부터
오는 경우는 20임.

Dijkstra algorithm (10/16)

Step 8

- Scanned Table에서 가장 작은 누적비용을 갖는 검사된 노드(B)를 선택하고 그것을 Reached Table로 이동 - 노드 B는 A와 E로 부터 오는데, 두 가지 모두 동일한 누적비용이므로 임의로 선택(E)



REACHED TABLE

NODE	CUMULATIVE COST	PREVIOUS NODE
A	:00	None
B	:15	E
C	:18	D
D	:15	E
E	:10	F
F	:08	A
G		

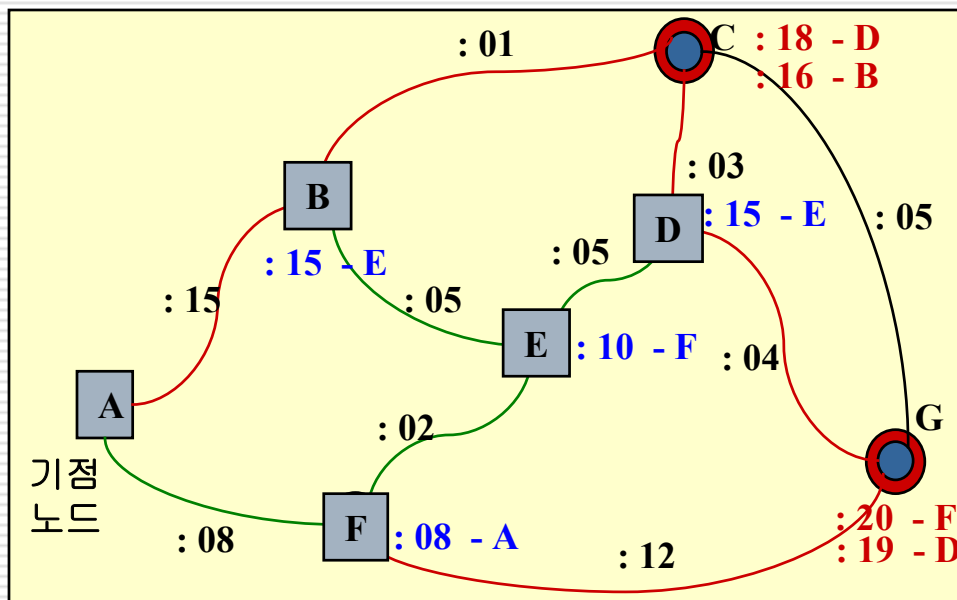
SCANNED TABLE

NODE	CUMULATIVE COST	PREVIOUS NODE
B	:15	A
G	:20	F
D	:15	E
C	:18	D
E	:19	D

Dijkstra algorithm (11/16)

□ Step 9

- 이제 막 도달된 상태의 노드(B)에 인접한 노드들(A, C, E)을 검사하고, 이들을 Scanned Table에 기록한 후, B는 도달상태로 기록
 - A와 E는 이미 도달상태이므로 제외



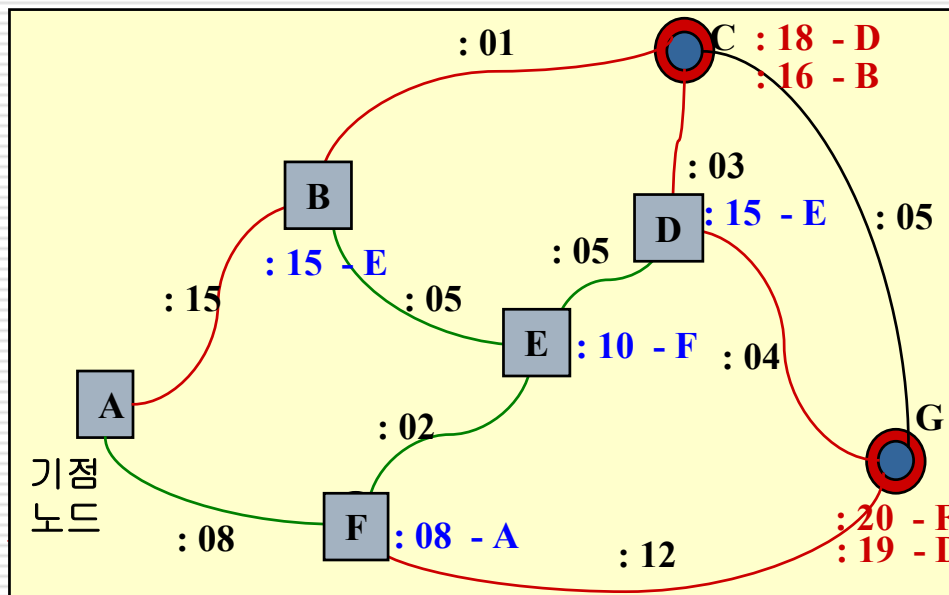
SCANNED TABLE

NODE	CUMULATIVE COST	PREVIOUS NODE
B	:15	A
G	:20	F
C	:18	D
G	:19	D
C	:16	B

Dijkstra algorithm (12/16)

Step 10

- Scanned Table의 노드 누적비용 15를 갖는 노드 B는 이제 도달상태이므로, Scanned Table에서 제거
- Scanned Table에서 가장 작은 누적비용을 갖는 검사된 노드(C)를 선택하고 그것을 Reached Table로 이동



SCANNED TABLE

NODE	CUMULATIVE COST	PREVIOUS NODE
B	15	A
G	20	F
C	18	D
G	19	D
C	16	B

Node B has been reached. Disregard this entry.

REACHED TABLE

NODE	CUMULATIVE COST	PREVIOUS NODE
A	00	none
B	15	E
C	16	B
D	15	E
E	10	F
F	08	A
G		

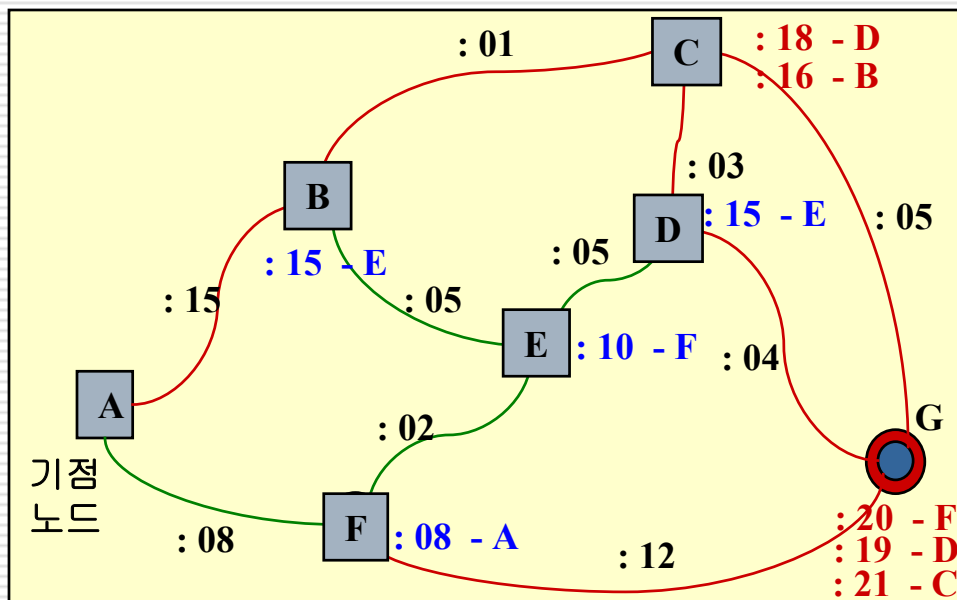
SCANNED TABLE

NODE	CUMULATIVE COST	PREVIOUS NODE
G	20	F
C	18	D
G	19	D
C	16	B

Dijkstra algorithm (13/16)

□ Step 11

- 이제 막 도달된 상태의 노드(C)에 인접한 노드들(B, D, G)을 검사하고, 이들을 Scanned Table에 기록한 후, C는 도달상태로 기록
- B와 D는 이미 도달상태이므로 제외



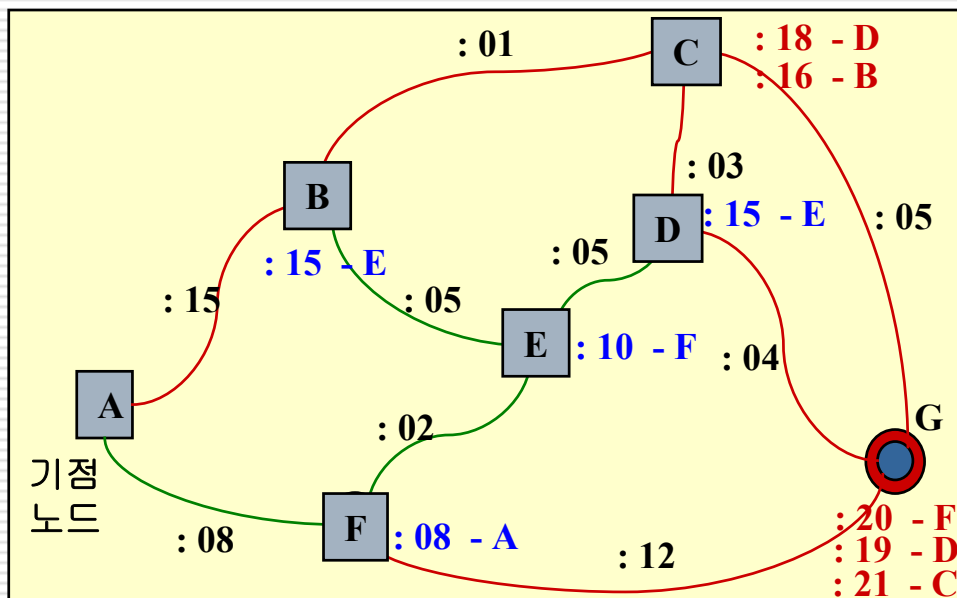
SCANNED TABLE

NODE	CUMULATIVE COST	PREVIOUS NODE
G	20	F
C	18	D
G	19	D
G	21	C

Dijkstra algorithm (14/16)

□ Step 12

- Scanned Table의 노드 누적비용 18를 갖는 노드 C는 이제 도달상태이므로, Scanned Table에서 제거
- Scanned Table에 남은 노드 G의 누적비용 가운데 가장 작은 D로부터 오는 G를 선택하고 그것을 Reached Table로 이동



SCANNED TABLE

NODE	CUMULATIVE COST	PREVIOUS NODE
G	20	F
C	18	B
G	19	D
G	21	C

Node C has been reached. Disregard this entry.

REACHED TABLE

NODE	CUMULATIVE COST	PREVIOUS NODE
A	00	none
B	15	E
C	16	B
D	15	E
E	10	F
F	08	A
G	19	D

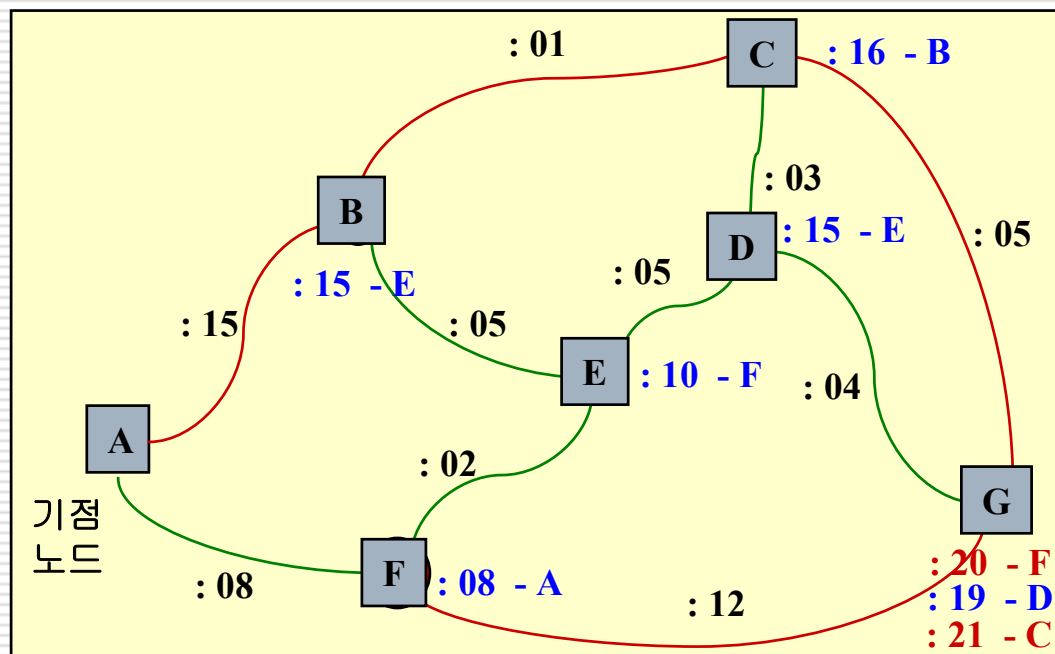
SCANNED TABLE

NODE	CUMULATIVE COST	PREVIOUS NODE
G	20	F
G	19	D
G	21	C

Dijkstra algorithm(15/16)

□ Step 13

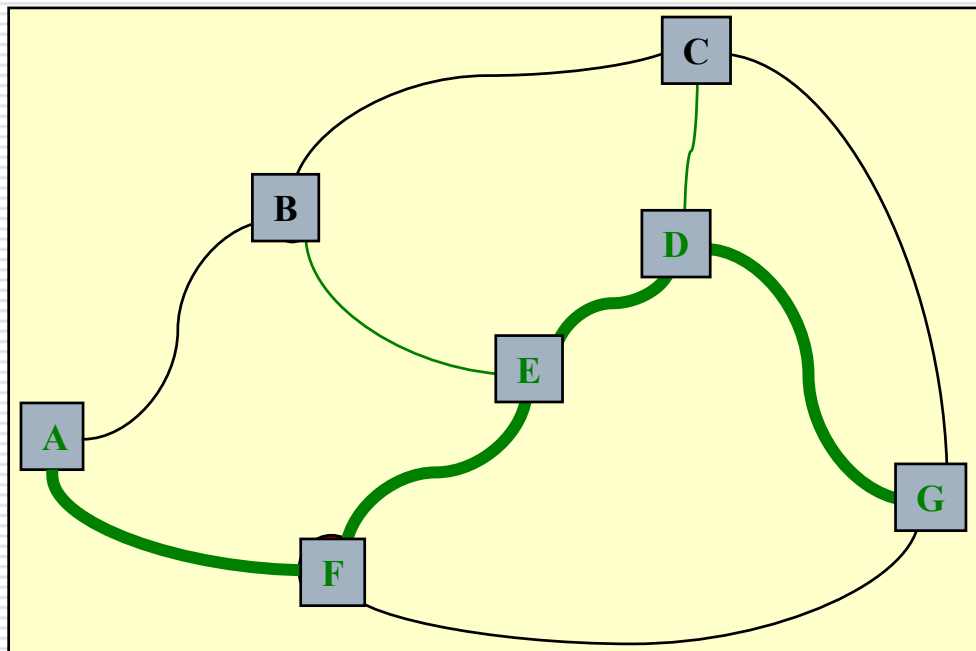
- 이제 Scanned Table에 남은 유일한 노드는 G이며, G에 인접하면서 도달되지 않은 상태의 노드는 없음
- 알고리즘 종료



Dijkstra algorithm (16/16)

□ 최종 결과

- A에서 G까지 최단경로를 도출하기 위해서는, Reached Table을 이용하여 노드 G로부터 역으로 이전노드들을 검사하면 됨.
- 기타 다른 종점 노드들에서 A까지 최단경로도 동일한 방법으로 찾을 수 있음



REACHED TABLE

NODE	CUMULATIVE COST	PREVIOUS NODE	
A	00	none	[5]
B	15	E	
C	16	B	
D	15	E	[2]
E	10	F	[3]
F	08	A	[4]
G	19	D	[1]

최단경로 탐색 알고리즘의 최대 병목지점

Procedure SPT(r);

begin

QINIT(r);

for $i = 1$ to n do $D[i] = \infty$;

$D[r] = 0$;

repeat

QOUT(Q, i);

foreach $j \in S[i]$ do

if $D[j] > D[i] + L[i, j]$ then

begin

$D[j] = D[i] + L[i, j]$;

QIN(Q, j);

end;

until $Q = \emptyset$;

end;

검사할 노드의 선택 전략

- Breadth First Search (FIFO)
- Depth First Search (LIFO)
- Best First Search

Bottleneck Operation

최소 표지를 갖는 노드를
선택하기 위해서는 큐 내에
있는 모든 임시표지 노드를
비교하여야 함

최단경로 탐색 알고리즘 관련 연구

□ Dijkstra 알고리즘의 개선 연구들

- QOUT() 연산시간을 줄이기 위한 자료구조 개선방법 개발에 집중
- 버킷(bucket) 자료구조
- 힙(heap) 자료구조
 - Binary heap, Fibonacci heap, Radix heap 등

□ 평가 연구들

- Gallo & Pallottino(1988)
 - Hung & Divoky(1988)
 - Cherkassky et al.(1996)
 - Zhan & Noon(1998)
-

최단경로 탐색 알고리즘 성능평가 실험

실험에 사용된 Dijkstra 알고리즘들

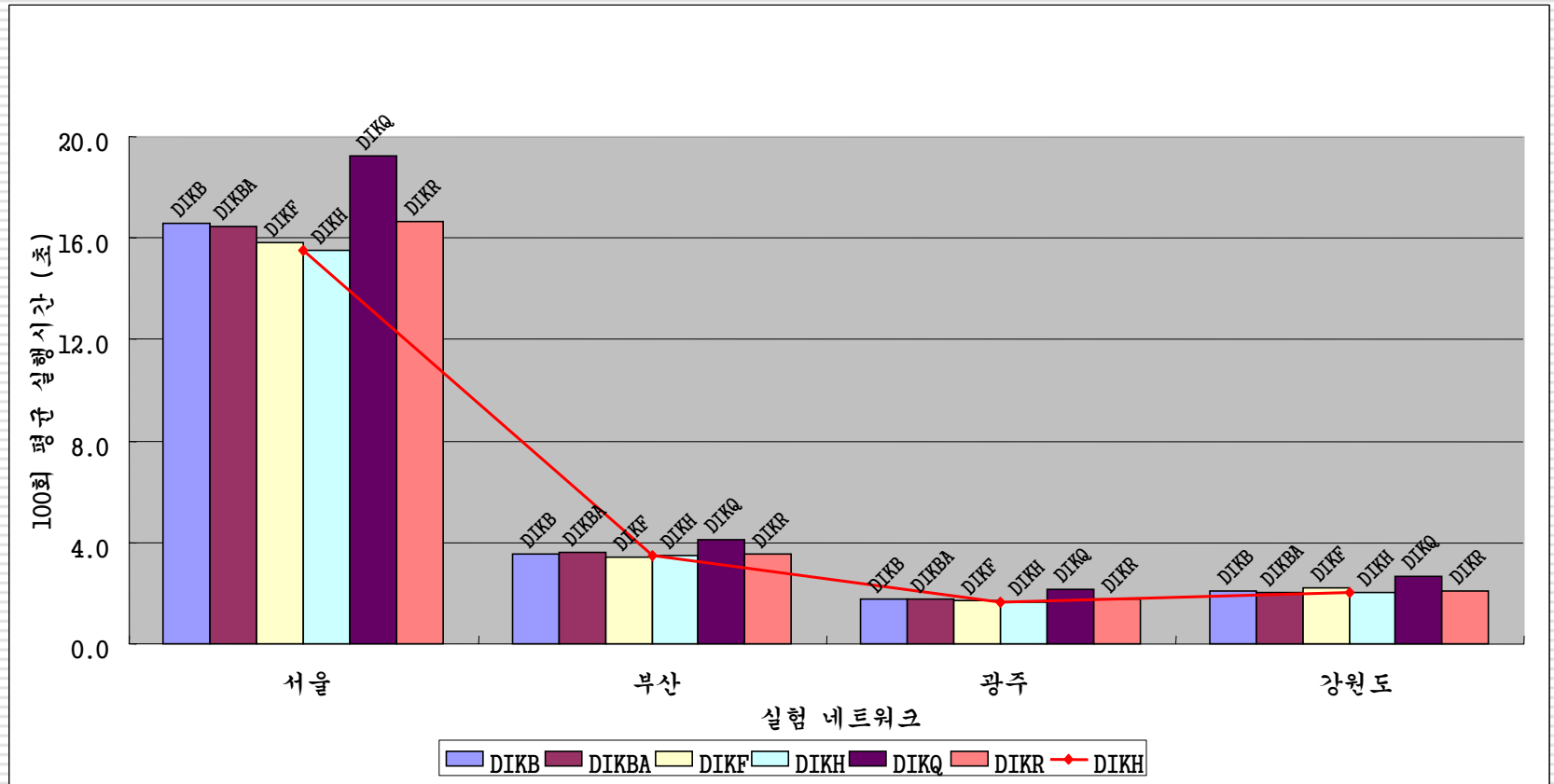
구현이름	설명(다음에 검사할 노드들을 선택하기 위한 전략)
DIKB	버킷을 이용한 최소 표지 선택
DIKBA	근사 버킷을 이용한 최소 표지 선택
DIKF	피보나치 힙을 이용한 최소 표지 선택
DIKH	k차 힙(k-ary heaps) 을 이용한 최소 표지 선택
DIKQ	최소 표지 선택을 위해 매번 정렬을 하게 되는 순수 Dijkstra 알고리즘
DIKR	기수 힙(R-heaps)을 이용한 최소 표지 선택

실험 네트워크의 특성

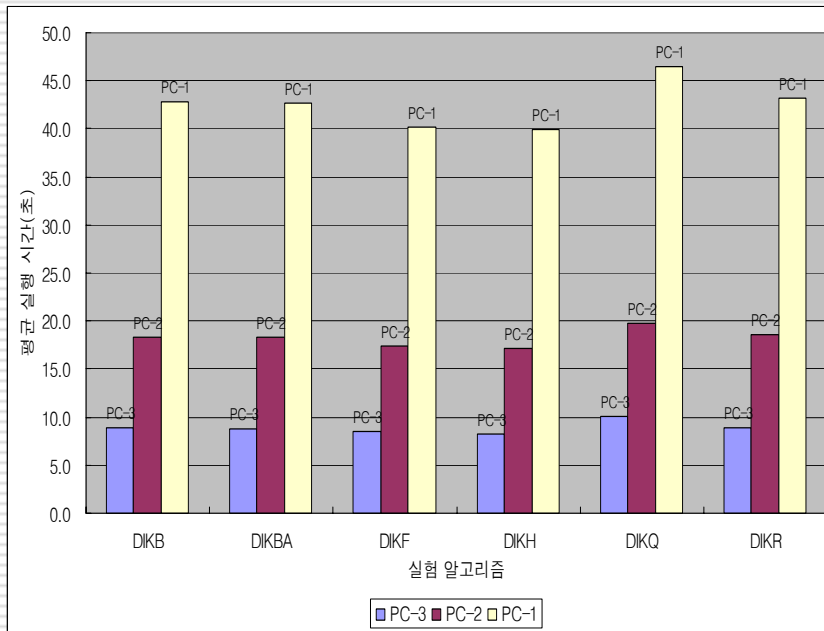
네트워크명	노드수	아크수	아크/노드 비율	최대 아크길이	최소 아크길이	평균 아크길이	아크길이 표준편차
서울(SEL)	52,915	77,339	1.46	5,725	1	104.339	149.900
부산(PUS)	17,992	25,790	1.43	7,966	1	132.826	224.835
광주(KWJ)	10,091	14,265	1.41	5,448	1	127.236	233.416
강원(KAW)	20,069	24,870	1.24	14,253	1	395.683	878.782

[illegible]

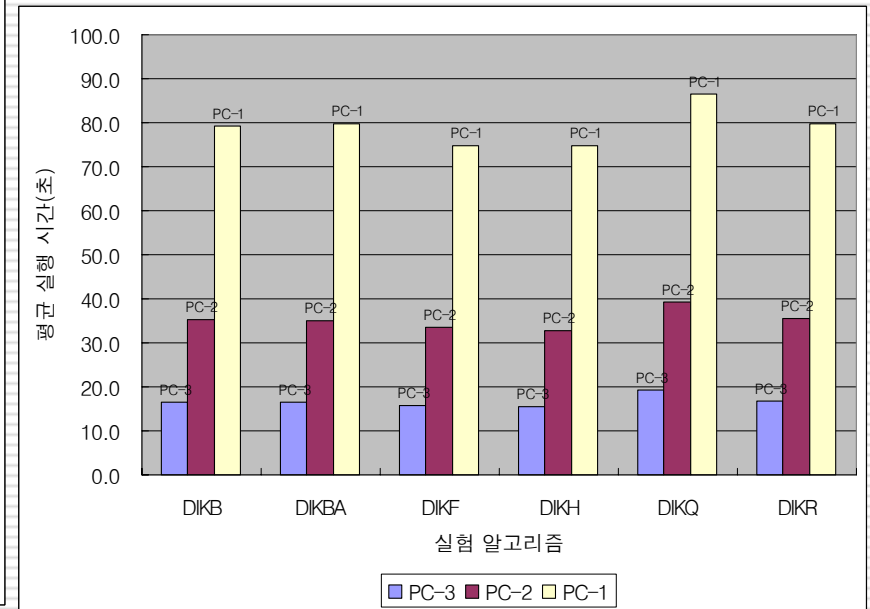
단일 기점 탐색 성능실험 결과



CPU별 탐색 성능실험 결과



단일 기·종점 실험



단일 기점 실험

결과 분석

- 힙을 이용한 알고리즘(DIKH)이 모든 네트워크에 대해 가장 우수한 성능을 보임
 - Zhan and Noon(1998)의 실험과는 다른 결과를 보임
 - DIKBA와 DIKBD의 수행성능이 월등히 좋았음
 - 본 실험에서는 버킷과 힙을 이용한 알고리즘간의 성능 차이가 크지는 않으나 기대했던 결과와는 다른 것임
 - 그 원인은 두 실험에 사용된 네트워크 연결도가 다르고, 회전제약 고려를 위한 코드 추가 등의 다른 실험환경에 기인한 것으로 판단됨
 - 결국 모든 네트워크에 최선인 알고리즘은 존재하지 않는다는 기존의 연구를 다시 확인할 수 있었음
 - 따라서 적용하고자 하는 분야와 네트워크 특성에 맞는 알고리즘을 선택하기 위해서는 이러한 성능평가 실험이 선행되어야 할 것임
-