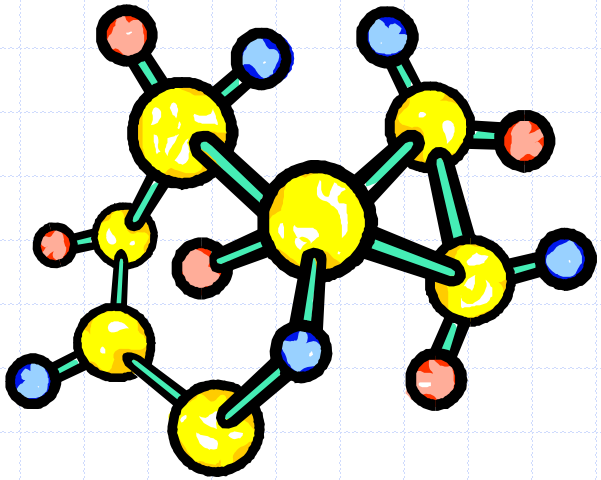
그래프

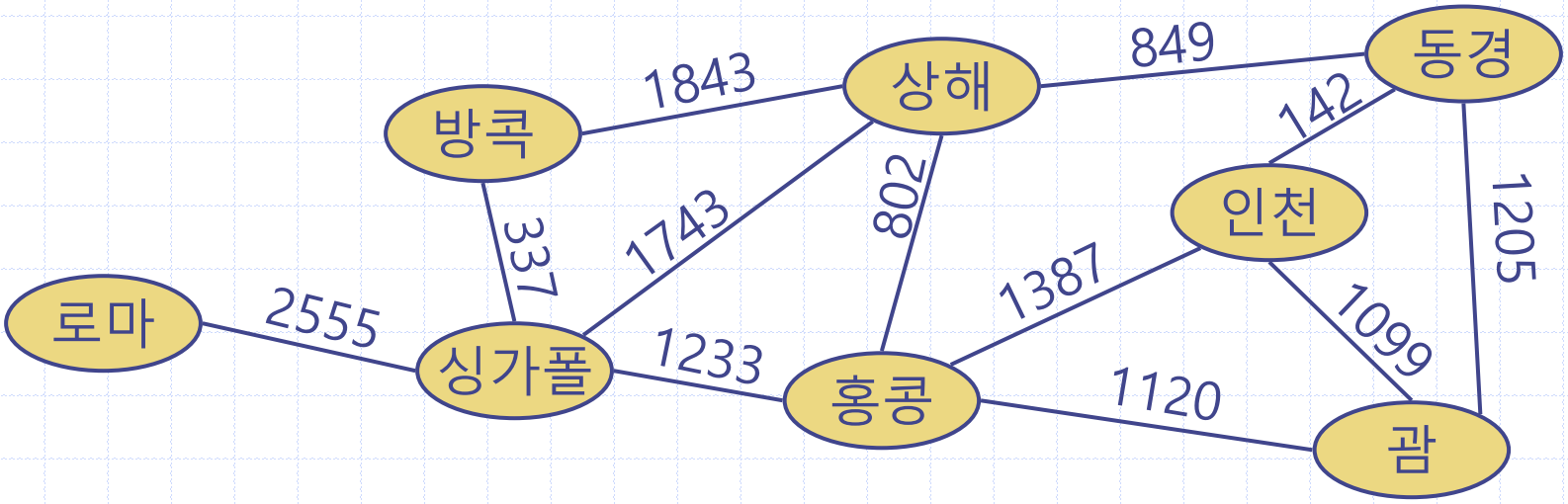


Outline

- 13.1 그래프 ADT
- 13.2 그래프 주요 개념
- 13.3 그래프 ADT 메쏘드
- 13.4 그래프 ADT 구현과 성능
- 13.5 응용문제

그래프 ADT

- **그래프**(graph): (V, E) 쌍 – 여기서
  - V: **정점**(vertex)이라 불리는 노드의 집합
  - E: **간선**(edge)이라 불리는 정점쌍들의 집합
  - 정점과 간선은 **원소**, 즉 **정보**를 저장
- **예**
  - 아래 예에서 **정점**은 공항을 표현하며 공항도시 이름을 저장
  - **간선**은 두 공항 사이의 항로를 표현하며 항로의 거리(mile)를 저장



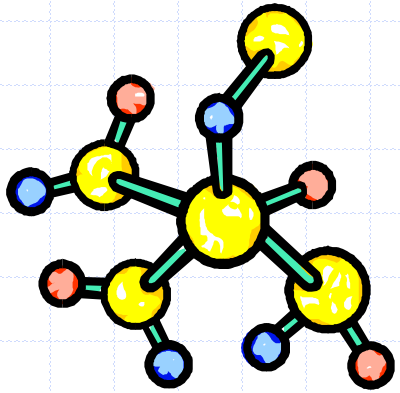
간선에 따른 그래프 유형

- **방향간선**(directed edge)
  - 정점들의 순서쌍 (u, v)
  - u: **시점**(origin)
  - v: **종점**(destination)
  - **예:** 항공편(flight)
- **방향그래프**(directed graph)
  - 모든 간선이 방향간선인 그래프
  - **예:** 항공편망(flight network)
- **무방향간선**(undirected edge)
  - 정점들의 무순쌍 (u, v)
  - **예:** 항로
- **무방향그래프**(undirected graph)
  - 무방향간선으로 이루어진 그래프
  - **예:** 항로망(flight route network)

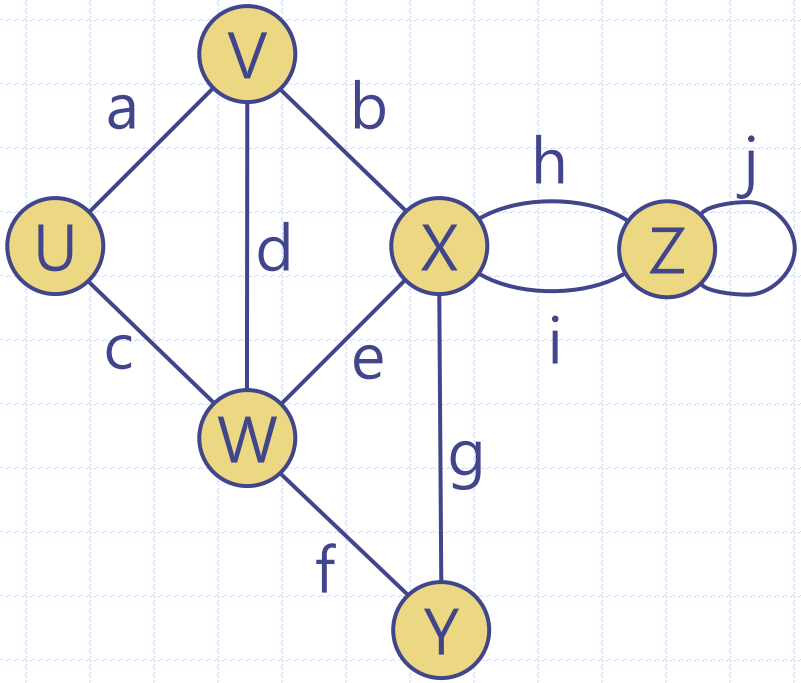
그래프 응용

- 전자회로
  - 인쇄회로기판(printed circuit board, PCB)
  - 집적회로(integrated circuit, IC)
- 교통망
  - 고속도로망
  - 항공노선망
- 컴퓨터 네트워크
  - LAN(local area network)
  - 인터넷
  - 웹
- 데이터베이스
  - 개체-관계 다이어그램(entity-relationship diagram)

그래프 용어

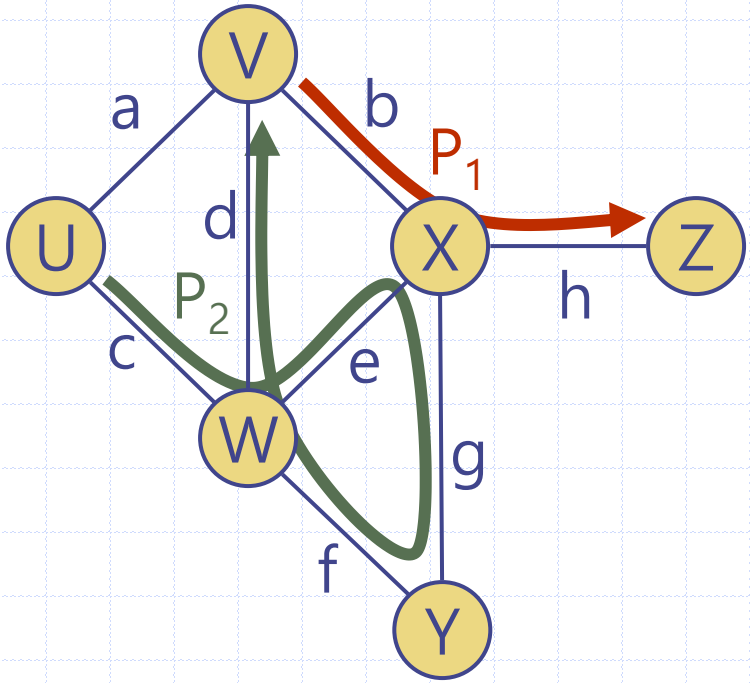


- 간선의 **끝점**(end vertex, 또는 endpoint)
  - 정점 U와 V는 a의 양끝점
- 정점의 **부착**(incident) 간선
  - a, d, b는 V에 부착한다
- 정점의 **인접**(adjacent) 정점
  - U와 V는 인접하다
- 정점의 **차수**(degree)
  - X의 차수는 5다
- **병렬 간선**(parallel edges)
  - h와 i는 병렬 간선
- **루프**(loop 또는 self-loop)
  - j는 루프다



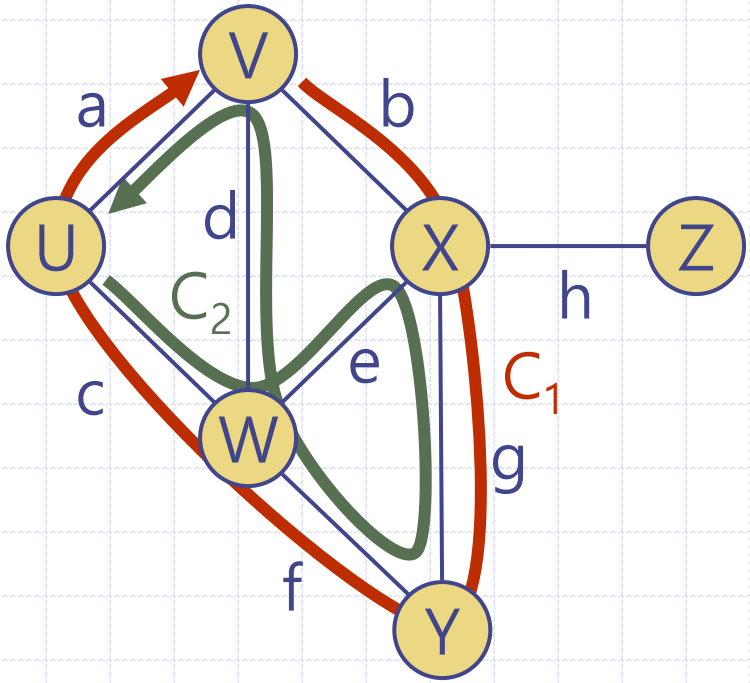
그래프 용어 (conti.)

- **경로**(path)
  - 정점과 간선의 교대열
  - 정점으로 시작하여 정점으로 끝난다
  - 각 간선은 그 양끝점으로 시작하고 끝난다
- **단순경로**(simple path)
  - 모든 정점과 간선이 유일한 경로
- **예**
  - P_1=(V,b,X,h,Z)은 단순경로
  - P_2=(U,c,W,e,X,g,Y,f,W,d,V)는 비단순경로



그래프 용어 (conti.)

- **싸이클**(cycle)
  - 정점과 간선이 교대하는 원형 열
  - 각 간선은 그 양끝점으로 시작하고 끝난다
- **단순싸이클**(simple cycle)
  - 모든 정점과 간선이 유일한 싸이클
- **예**
  - C_1=(V,b,X,g,Y,f,W,c,U,a)는 단순싸이클
  - C_2=(U,c,W,e,X,g,Y,f,W,d,V,a)는 비단순싸이클



속성

속성 1

$\Sigma_v deg(v) = 2m$

증명: 각 간선이 두 번 세어진다

속성 2

루프와 병렬 간선이 없는 무방향그래프에서,

$m \le n(n-1)/2$

증명: 각 정점의 최대 차수는 $(n-1)$

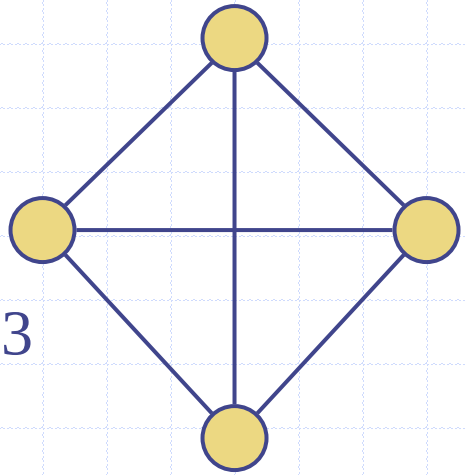
방향그래프에서 m의 상한은?

표기

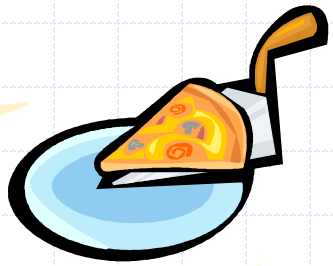
n	정점 수
m	간선 수
$deg(v)$	정점 v의 차수

예

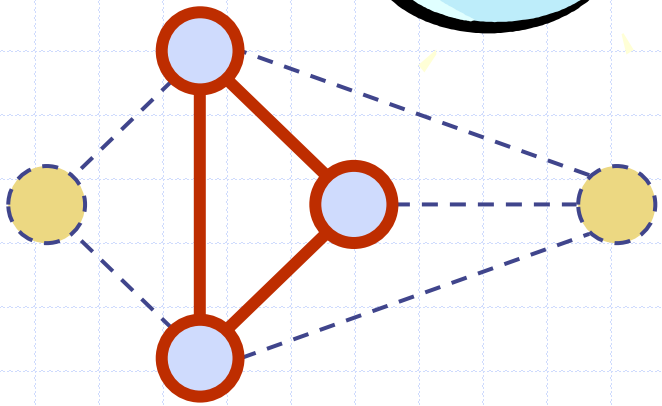
- $n = 4$
- $m = 6$
- $deg(v) = 3$



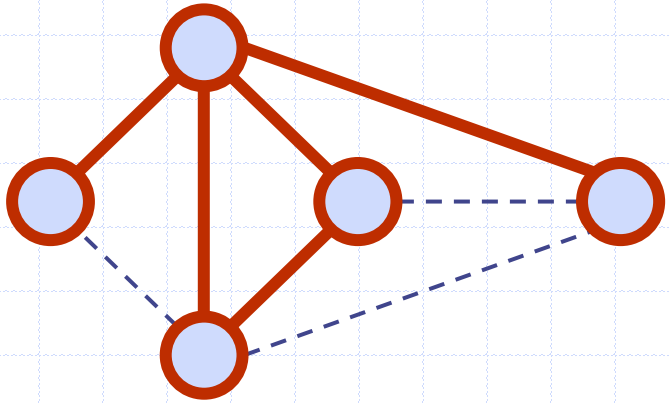
부그래프



- 그래프 $G = (V, E)$의 **부그래프**(subgraph): 다음 정점과 간선으로 구성된 그래프
  - **정점:** V의 부분집합
  - **간선:** E의 부분집합
- 그래프 $G = (V, E)$의 **신장 부그래프**(spanning subgraph): 다음 정점과 간선으로 구성된 그래프
  - **정점:** V
  - **간선:** E의 부분집합

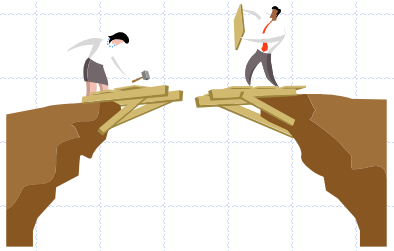


부그래프

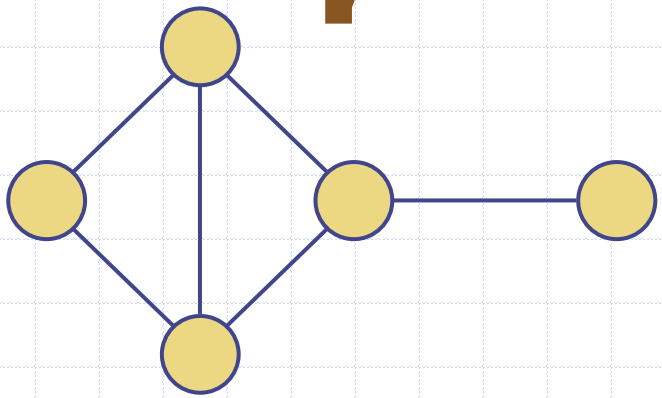


신장 부그래프

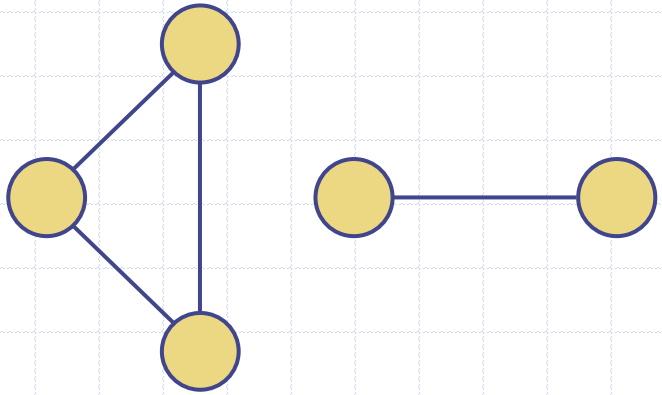
연결성



- 모든 정점쌍에 대해 경로가 존재하면 "그래프가 **연결**(connected)되었다"고 말한다
- 그래프 G의 **연결요소**(connected component): G의 최대 연결 부그래프

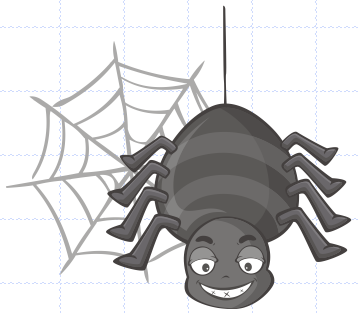


연결그래프

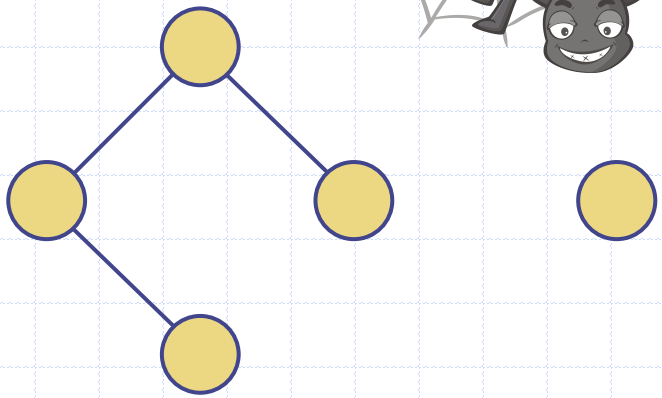


두 개의 연결요소로 구성된 비연결그래프

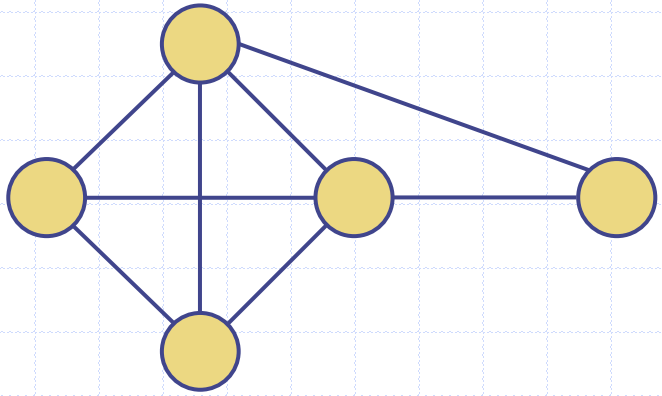
밀집도



- 그래프 알고리즘의 선택은 종종 간선의 **밀집도**에 따라 좌우된다
- **예:** 주어진 그래프 G에 대해, 알고리즘 A와 B가 동일한 문제를 각각 $O(nm)$ 시간과 $O(n^2)$ 시간에 해결할 경우,
  - G가 **희소**하다면, 알고리즘 A가 B보다 빠르다
  - G가 **밀집**하다면, 알고리즘 B가 A보다 빠르다

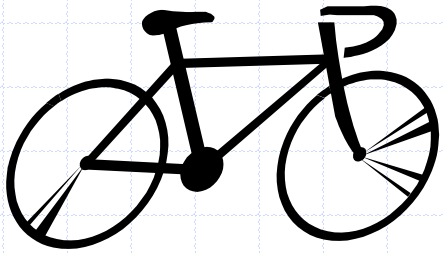


희소그래프



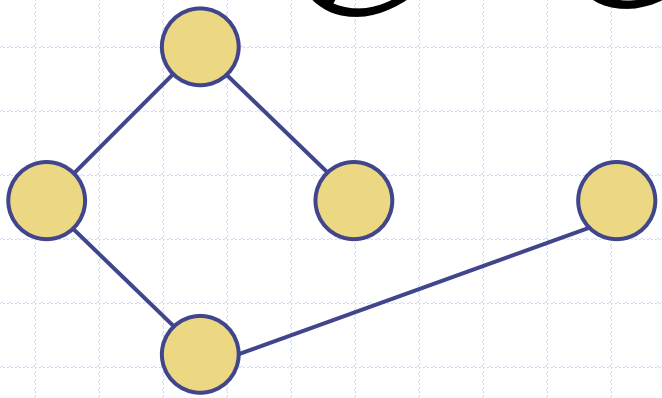
밀집그래프

싸이클

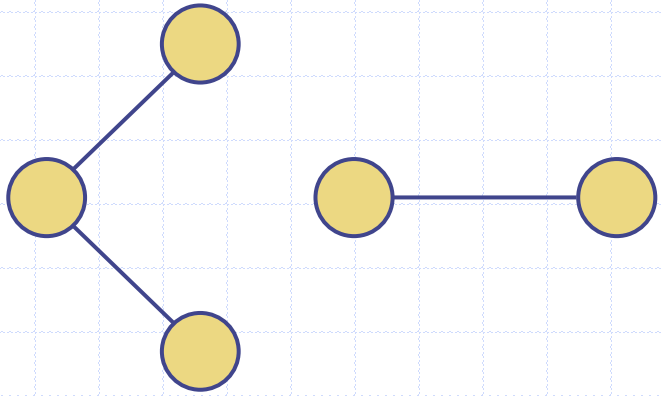


- **자유트리**(free tree), 또는 **트리**: 다음 조건을 만족하는 무방향그래프 T
  - T는 연결됨
  - T에 싸이클이 존재하지 않음

  (위 트리에 대한 정의는 **루트가 있는 트리**에 대한 정의와는 다르다)
- **숲**(forest): 싸이클이 없는 무방향그래프
- 숲의 연결요소는 트리들이다

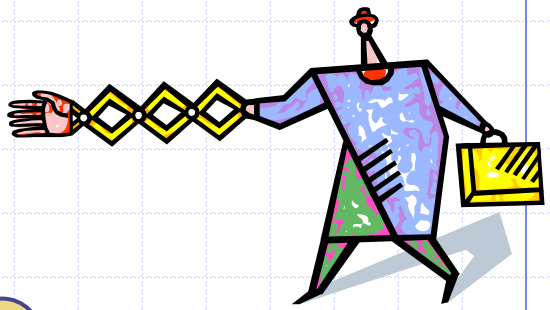


트리

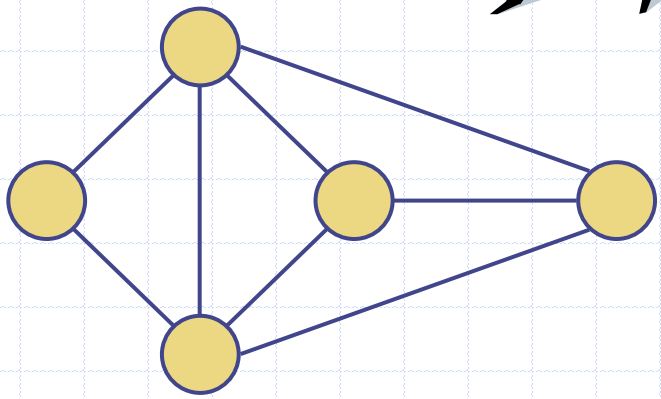


숲

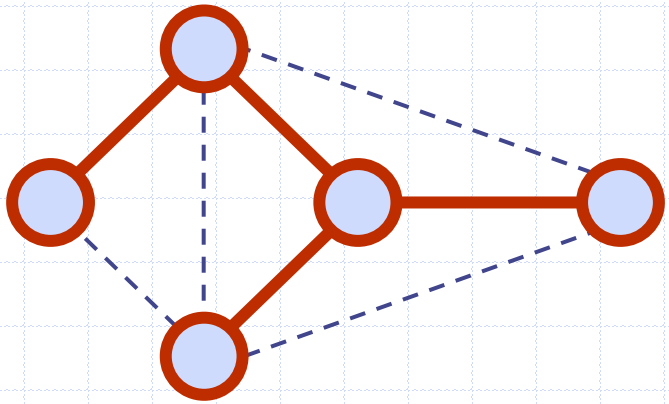
신장



- 연결그래프의 **신장트리**(spanning tree): 신장 부그래프 가운데 트리인 것
- 신장트리는 그래프가 트리가 아닌 한, 유일하지 않다
- 신장트리는 통신망 설계에 응용된다
- 그래프의 **신장숲**(spanning forest): 신장 부그래프 가운데 숲인 것

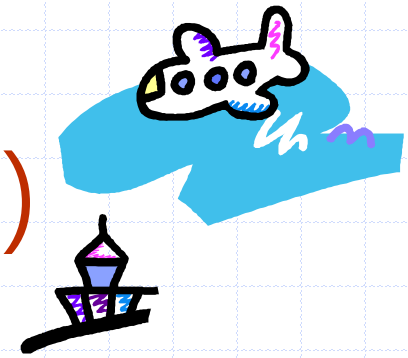


그래프



신장트리

그래프 ADT 메쏘드(공통)



- 정점과 간선들은 **원소**를 저장
- 일반 메쏘드
  - integer numVertices()
  - integer numEdges()
  - iterator vertices()
  - iterator edges()
- 접근 메쏘드
  - vertex aVertex()
- 질의 메쏘드
  - boolean isDirected(e)
- 반복 메쏘드
  - iterator directedEdges()
  - iterator unDirectedEdges()
- 갱신 메쏘드
  - vertex insertVertex(o)
  - removeVertex(v)
  - removeEdge(e)

무방향그래프 ADT 메쏘드

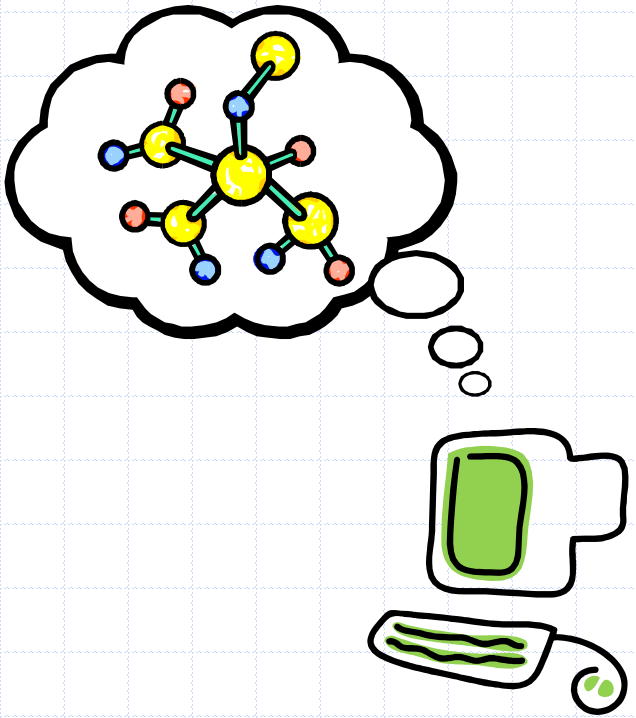
- 접근 메쏘드
  - integer deg(v)
  - vertex opposite(v, e)
- 질의 메쏘드
  - boolean areAdjacent(v, w)
- 반복 메쏘드
  - iterator endVertices(e)
  - iterator adjacentVertices(v)
  - iterator incidentEdges(v)
- 갱신 메쏘드
  - edge insertEdge(v, w, o): 정점 v에서 w로 항목 o를 저장한 무방향간선을 삽입하고 반환

방향그래프 ADT 메쏘드

- 접근 메쏘드
  - vertex origin(e)
  - vertex destination(e)
  - integer inDegree(v)
  - integer outDegree(v)
- 반복 메쏘드
  - iterator inIncidentEdges(v)
  - iterator outIncidentEdges(v)
  - iterator inAdjacentVertices(v)
  - iterator outAdjacentVertices(v)
- 갱신 메쏘드
  - edge insertDirectedEdge(v, w, o): 정점 v에서 w로 항목 o를 저장한 방향간선을 삽입하고 반환
  - makeUndirected(e): 간선 e를 무방향으로 전환
  - reverseDirection(e): 방향간선 e를 역행

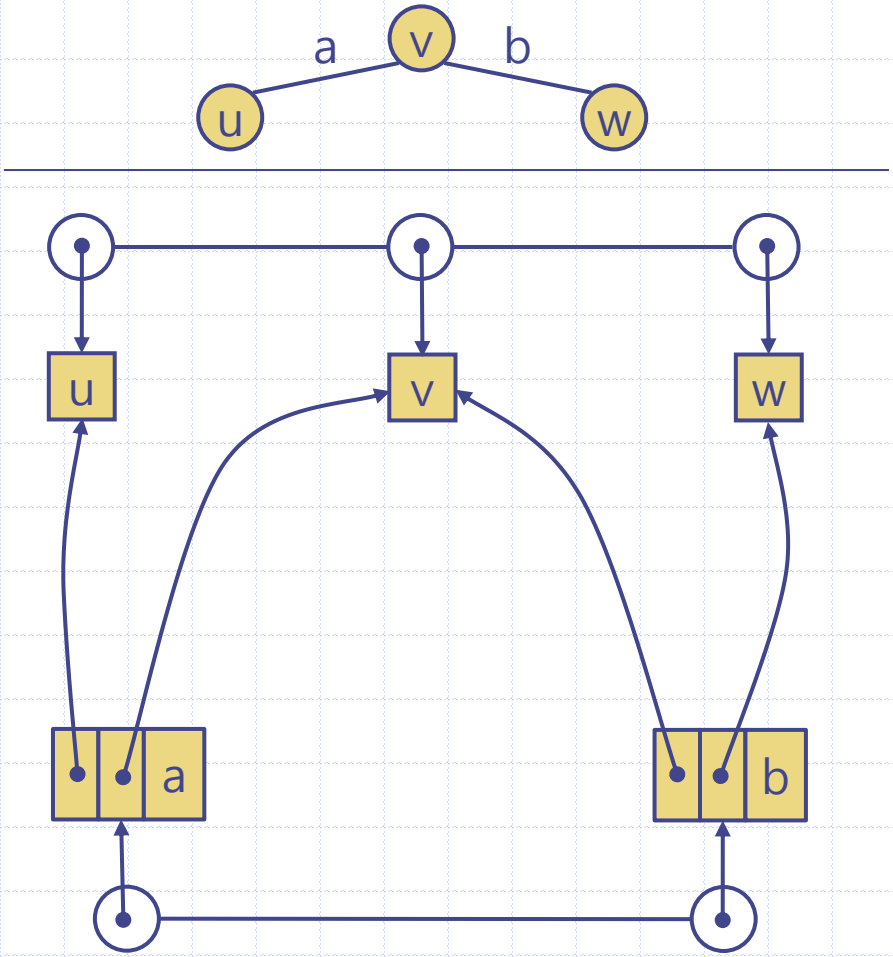
그래프 구현

- 간선리스트(edge list) 구조
- 인접리스트(adjacency list) 구조
- 인접행렬(adjacency matrix) 구조



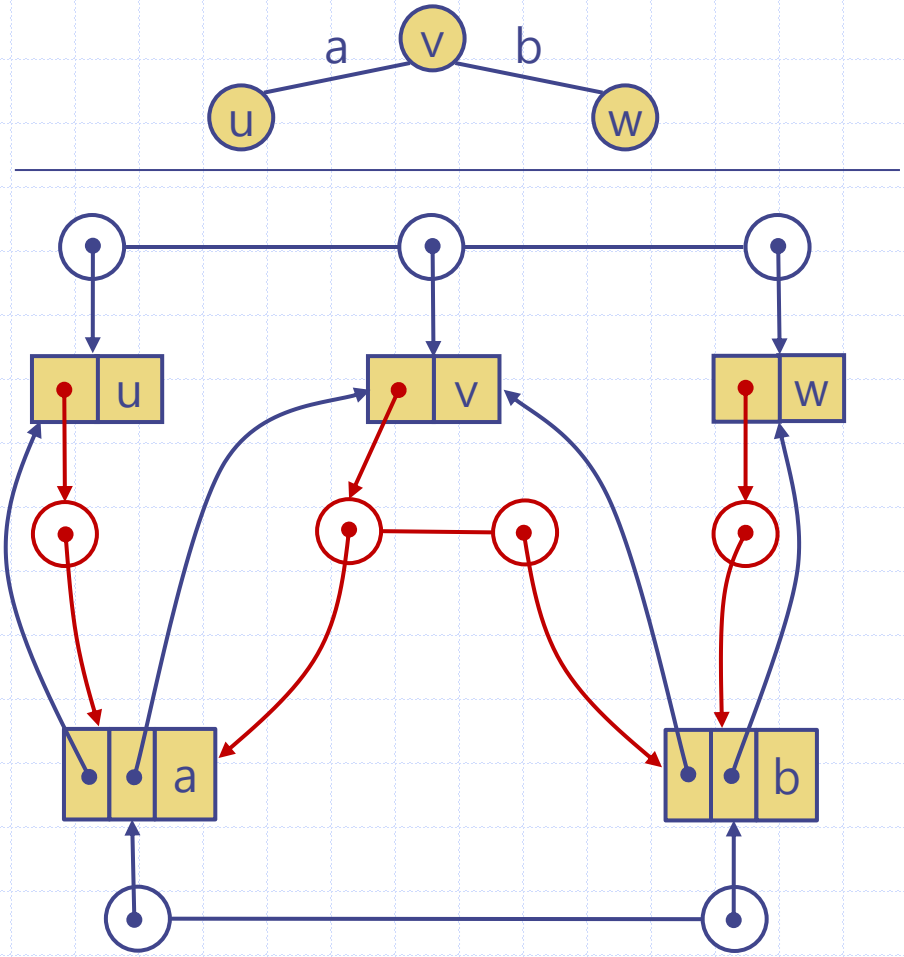
간선리스트 구조

- 정점리스트
  - 정점 노드들에 대한 포인터의 리스트
- 간선리스트
  - 간선 노드들에 대한 포인터의 리스트
- 정점 노드
  - 원소
- 간선 노드
  - 원소
  - 시점 노드
  - 종점 노드



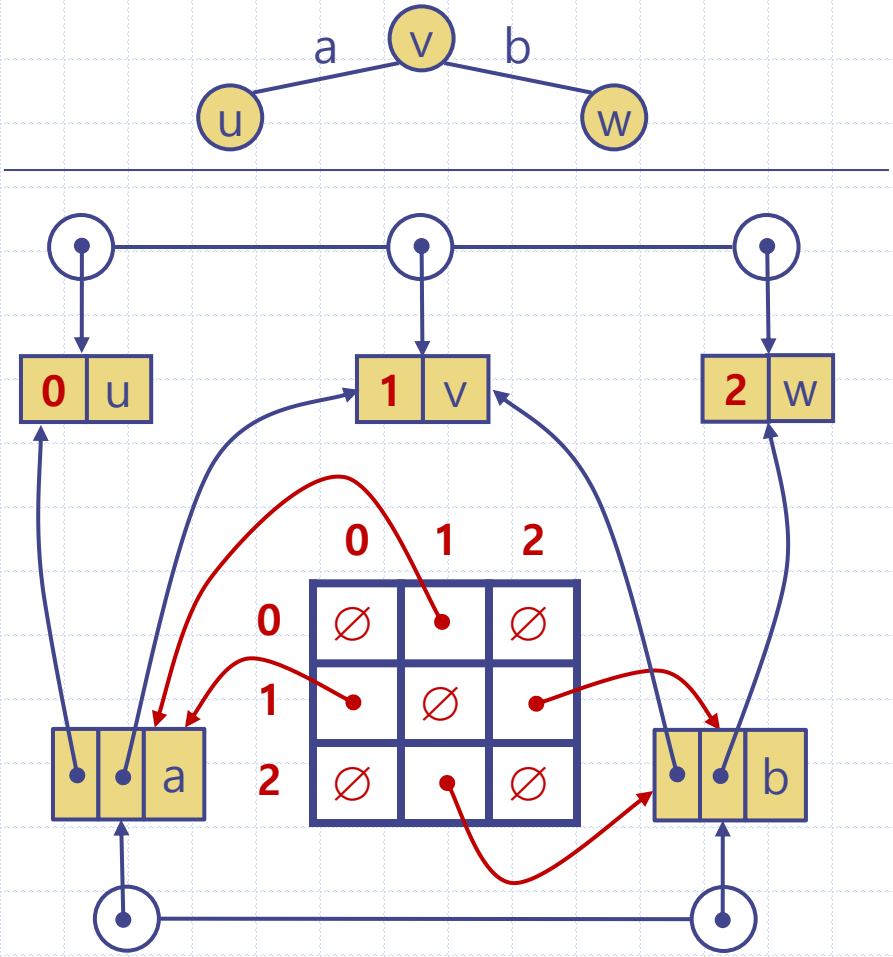
인접리스트 구조

- 간선리스트 구조 + α
- 각 정점에 대한 **부착리스트**
  - 각 정점의 부착간선들을 간선 노드에 대한 참조들의 리스트로 표시

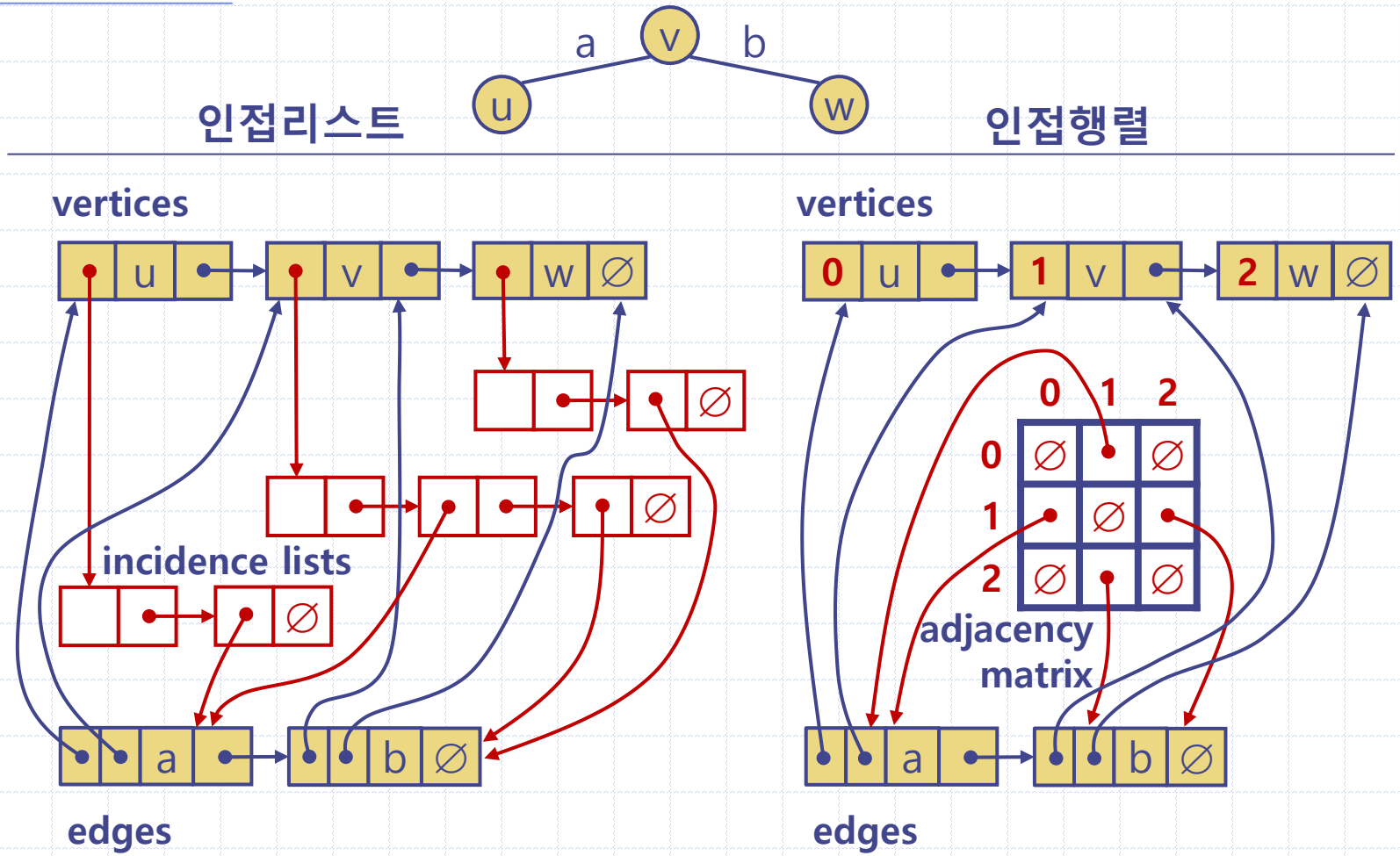


인접행렬 구조

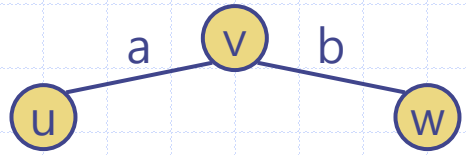
- 간선리스트 구조 + α
- 정점 개체에 대한 확장
  - 정점에 해당하는 정수 키(첨자)
- **인접행렬**
  - $n \times n$ 배열
  - 인접정점 쌍에 대응하는 간선 노드들에 대한 참조
  - 비인접정점 쌍에 대한 널 정보
- "구식 버전"은 간선의 존재여부만을 1(간선 존재)과 0(간선 부존재)으로 표시함



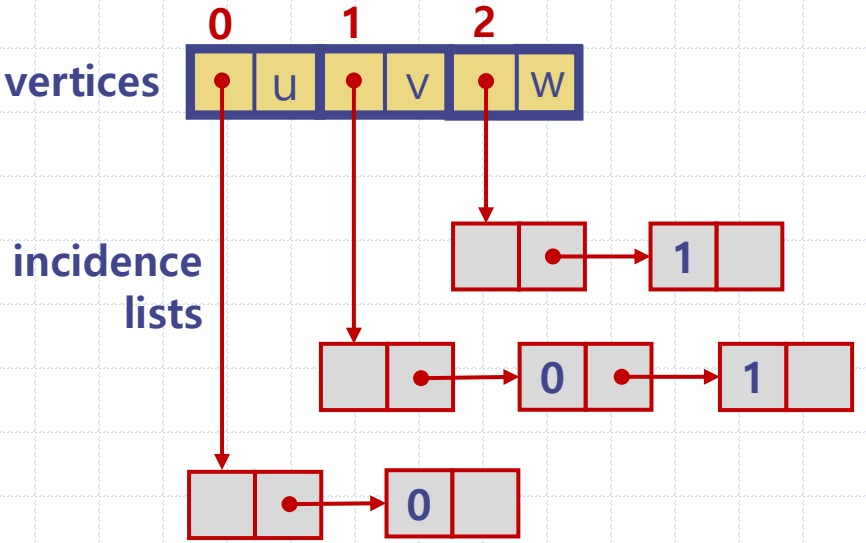
연결리스트를 이용한 상세 구현

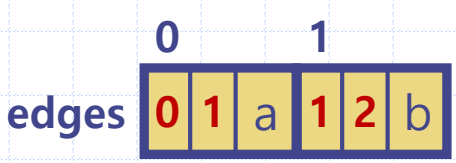


배열을 이용한 상세 구현

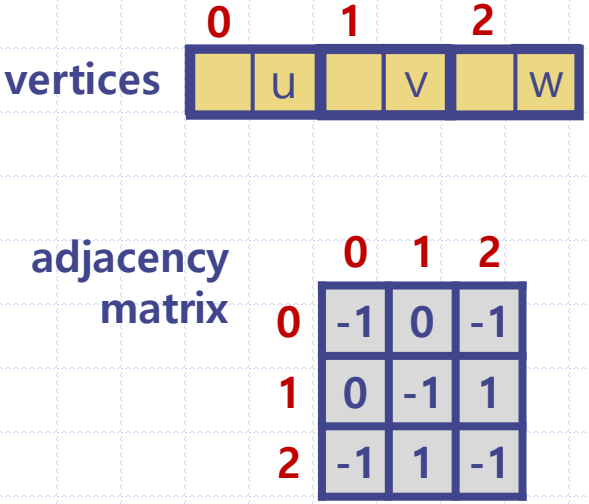


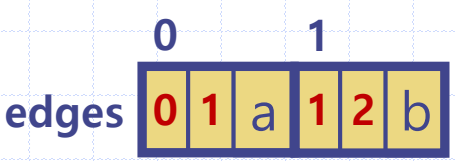
인접리스트





인접행렬

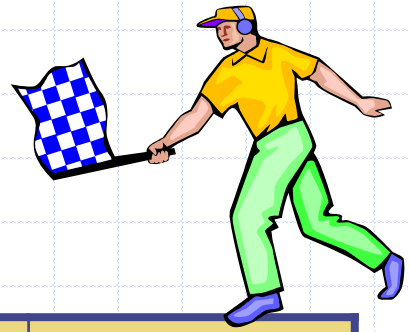




그래프 상세 구현 비교

		인접리스트	인접행렬
연결리스트	정점리스트, 간선리스트	동적메모리 노드의 연결리스트	
	정점, 간선	동적메모리 노드	
	인접 정보	포인터의 연결리스트	2D 포인터 배열
	장점	동적 그래프에 사용 시 유리	
	단점	다수의 포인터 사용으로 복잡	
배열	정점리스트, 간선리스트	구조체 배열	
	정점, 간선	구조체	
	인접 정보	첨자의 연결리스트	2D 첨자 배열
	장점	다수의 포인터를 첨자로 대체하여 단순	
	단점	동적 그래프에 사용 시 불리	

점근 성능 비교



◆ n 정점과 m 간선 ◆ 병렬 간선 없음 ◆ 루프 없음 ◆ "big-Oh" 한계임	간선 리스트	인접리스트	인접행렬
공간	$n + m$	$n + m$	n^2
incidentEdges(v)	m	$deg(v)$	n
adjacentVertices(v)	m	$deg(v)$	n
areAdjacent(v, w)	m	$min(deg(v), deg(w))$	1
insertVertex(o)	1	1	n
insertEdge(v, w, o)	1	1	1
removeVertex(v)	m	$deg(v)$	n
removeEdge(e)	1	1	1