대학교 2학년 시험 대비를 위해, 공유해주신 PDF 파일들을 바탕으로 출제 포인트를 중심으로 핵심 내용을 정리해 드립니다.

시험 대박 나시길 응원하며, 각 장별 중요 포인트를 요약합니다.

08장: 트리 (Tree)

1. 일반 트리의 표현

- 일반 트리는 자식 노드의 수가 일정하지 않아 표현이 어렵습니다. 이를 해결하기 위해 '왼쪽 자식-오른쪽 형제**(Left Child-Right Sibling)**' 표현법을 사용하여 이진 트리로 변환하여 표현할 수 있습니다. [1]

2. 이진트리의 분류 **(**포화 **vs** 완전**)**

- 포화 이진 트리 **(Full Binary Tree):** 트리의 각 레벨에 노드가 꽉 차 있는 트리입니다. 높이가 k일 때 노드의 개수는 정확히 2^k-1개입니다. [2]

- 완전 이진 트리 **(Complete Binary Tree):** 높이가 k일 때, 레벨 1부터 $k-1$까지는 노드가 꽉 차 있고, 마지막 레벨 k에서는 왼쪽부터 오른쪽으로 노드가 순서대로 채워져 있는 트리입니다. 중간에 빈 곳이 없어야 합니다. [3]

3. 이진트리의 표현 **(**링크 표현법**)**

- 배열 표현법은 메모리 낭비가 심할 수 있어, 주로 링크 표현법을 사용합니다.
- 구조**:** 데이터를 저장하는 필드와 왼쪽 자식, 오른쪽 자식을 가리키는 2개의 포인터 필드로 구성됩니다. [4][4][4]
    - struct Node { data; Node* left; Node* right; } 형태를 가집니다. [5]

4. 이진트리의 순회 (Traversal) [6]

- 전위 순회 **(Preorder, VLR):** 루트**(V)** $\rightarrow$ 왼쪽(L) $\rightarrow$ 오른쪽(R) [7]

- 중위 순회 **(Inorder, LVR):** 왼쪽(L) $\rightarrow$ 루트**(V)** $\rightarrow$ 오른쪽(R) [8]

- 후위 순회 **(Postorder, LRV):** 왼쪽(L) $\rightarrow$ 오른쪽(R) $\rightarrow$ 루트**(V)** [9]
    - 시험 팁*:* 트리의 루트 노드가 언제 방문되는지를 기준으로 이름을 기억하면 쉽습니다. (전위-먼저, 중위-중간, 후위-나중)

10장: 교착 상태 (Deadlock)

1. 교착 상태 발생 4가지 필요 조건 10
다음 4가지 조건이 모두 만족되어야 교착 상태가 발생할 수 있습니다.

1. 상호 배제 **(Mutual Exclusion):** 자원은 한 번에 한 프로세스만 사용할 수 있음.
2. 점유와 대기 **(Hold and Wait):** 자원을 가진 상태에서 다른 자원을 기다림.
3. 비선점 **(No Preemption):** 다른 프로세스의 자원을 강제로 뺏을 수 없음.
4. 순환 대기 **(Circular Wait):** 대기 프로세스의 집합이 순환 형태로 자원을 기다림.

2. 자원 할당 그래프와 **Deadlock** 판별

- 자원 할당 그래프**:** 프로세스와 자원 간의 요청 및 할당 관계를 방향 그래프로 표현한 것입니다. [11]

- **Deadlock** 조건 정의 **(**그래프 기준**):**
  - 자원 할당 그래프에 **사이클(Cycle)**이 없다면 교착 상태가 아닙니다.
  - 사이클이 있다면**:**
    - 자원 유형마다 인스턴스가 하나뿐이면 $\rightarrow$ 필연적으로 교착 상태입니다. [12]

    - 자원 유형마다 인스턴스가 여러 개라면 $\rightarrow$ 교착 상태일 수도 있고 아닐 수도 있습니다 (교착 상태의 필요조건). [13]

11장: 그래프 (Graph)

1. 그래프의 정의와 오일러 경로

- 그래프**:** 정점(Vertex)과 간선(Edge)의 집합 $G=(V, E)$입니다. [14]

- 오일러 경로 **(Euler Path):** 그래프의 모든 간선을 한 번씩만 통과하여 출발점으로 돌아오는 경로입니다.
  - 조건*:* 그래프의 모든 정점에 연결된 간선의 개수**(**차수**)**가 짝수여야 오일러 경로가 존재합니다. (홀수 차수 정점이 없어야 함) [15]

2. 그래프 표현 방법

- 인접 행렬 **(Adjacency Matrix):** $N \times N$ 2차원 배열을 사용합니다. 간선이 있으면 1, 없으면 0으로 표현합니다. 밀집 그래프에 적합하며, 두 정점의 연결 여부를 $O(1)$에 알 수 있습니다. [16]

- 인접 리스트 **(Adjacency List):** 각 정점마다 연결된 정점들을 연결 리스트로 표현합니다.

희소 그래프에 적합하며 메모리를 효율적으로 씁니다. [17]

3. 그래프 탐색

- 깊이 우선 탐색 **(DFS):** 시작 정점에서 한 방향으로 갈 수 있는 데까지 가다가 더 이상 갈 수 없으면 가장 가까운 갈림길로 돌아와 다른 방향을 탐색합니다. 스택**(Stack)** 또는 재귀 호출을 사용합니다. [18]

- 너비 우선 탐색 **(BFS):** 시작 정점으로부터 가까운 정점을 먼저 방문하고 멀리 있는 정점을 나중에 방문합니다. **큐(Queue)**를 사용합니다. [19]

4. 신장 트리 **(Spanning Tree) [**중요**]**

- 정의**:** 그래프 내의 모든 정점을 포함하면서 사이클이 없는 트리입니다.
- 특징**:** n개의 정점이 있을 때, 정확히 **$n-1$**개의 간선을 가집니다. [2020]

- 최소 비용 신장 트리 **(MST):** 간선의 가중치 합이 최소가 되는 신장 트리입니다. (Kruskal, Prim 알고리즘 사용) [21]

13장**:** 정렬 **(Sorting)**

1. 선택 정렬 **(Selection Sort)**

- 알고리즘**:** 정렬되지 않은 리스트 부분에서 최솟값을 찾아 정렬된 부분의 바로 다음 위치**(**맨 앞**)**와 교환하는 방식입니다. [22]

- 효율적인 이유 **(**장점**):**
  - 비교 횟수는 $O(n^2)$으로 많지만, 자료의 이동**(**교환**)** 횟수가 미리 결정되며 약 $O(n)$번으로 적습니다.
  - 따라서 자료의 이동 비용이 크고, 비교 비용이 저렴한 경우에 효율적입니다. [23]

보너스 문제 대비

1. 바이브 코딩 **(Vibe Coding)**

- 정의**:** 생성형 AI(LLM)를 활용하여 코드를 작성하는 새로운 방식으로, 엄격한 논리보다는 **직감과 느낌(Vibe)**에 의존하여 빠르게 개발하는 접근법입니다. 2025년 2월 **안드레이 카파시(Andrej Karpathy)**가 제안했습니다. [24]

- 특징**:** 자연어 프롬프트로 지시 $\rightarrow$ AI가 코드 생성 $\rightarrow$ 실행 및

피드백의 반복 루프(Loop)를 가집니다. [25]

- 변화**:** 개발자의 역할이 '코드 작성자'에서 '설계자/검증자(Reviewer)'로 변화합니다. [26]

2. LLM (Large Language Model) / LMM (Large Multimodal Model)

- 바이브 코딩의 핵심 도구로, 방대한 데이터를 학습하여 인간의 언어로 된 지시를 이해하고 코드를 생성해주는 거대 언어 모델입니다. [27]

3. 신장 코딩**? (Spanning Tree vs Coding?)**

- 주의*:* 제공해주신 파일 중 '신장 코딩'이라는 정확한 용어는 없습니다. 시험 범위에 **11장 신장 트리(Spanning Tree)**가 [중요]하게 표시되어 있으므로, **신장 트리(Spanning Tree)**를 묻는 문제일 가능성이 매우 높습니다.
- 혹시 **'신장(Extension) 코딩'**이나 바이브 코딩의 특성인 **'신속한 프로토타이핑'**을 의미하는지 확인이 필요하지만, 문맥상 11장의 **Spanning Tree(**모든 정점을 포함**,** 사이클 없음**, n-1**개 간선**)** 개념을 확실히 암기해 가시는 것을 추천합니다. [28282828]

시험 준비 팁**:**

- 트리 순회**:** 직접 트리를 그려놓고 VLR, LVR, LRV 순서로 노드를 나열해 보세요.
- 그래프**:** 인접 행렬과 인접 리스트를 그리는 법, DFS와 BFS의 방문 순서 차이를 이해하세요.
- 교착 상태**:** 4가지 조건(상호배제, 점유대기, 비선점, 순환대기)은 서술형으로 나올 수 있으니 키워드를 꼭 외우세요.
- 바이브 코딩**:** '안드레이 카파시'와 '개발자 역할의 변화(작성자 $\rightarrow$ 관리자/검증자)'가 핵심 키워드입니다.