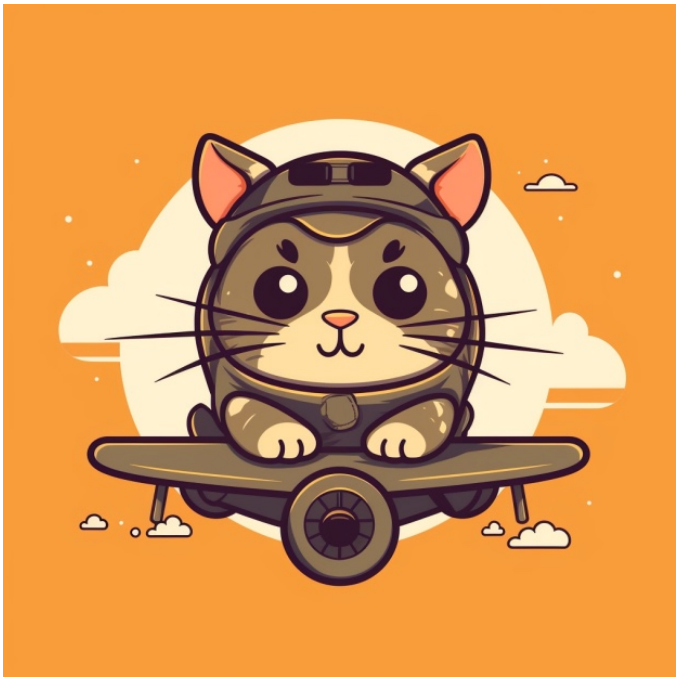
🥷

[취업폭격기 Zeromini 위클리 개념폭격 #22]

📕 과목 : 자료구조론

🔥 참고문제 : 2022년 군무원 7급

🌝 문제 수정 버전 : V 1.0



1.이진 트리의 순회

- 문제: 이진 트리에서 전위(pre-order), 중위(in-order), 후위(post-order) 순회의 차이점에 대해 설명하시오.
- 해설: 이진 트리의 전위 순회는 루트 노드를 먼저 방문한 후 왼쪽 서브트리, 그리고 오른쪽 서브트리 순으로 방문한다. 중위 순회는 왼쪽 서브트리를 먼저 방문한 후 루트 노드, 그리고 오른쪽 서브트리 순으로 방문한다. 후위 순회는 왼쪽 서브트리, 오른쪽 서브트리 순으로 방문한 후 마지막에 루트 노드를 방문한다.

2.해쉬 테이블의 충돌 해결

- 문제: 해쉬 테이블에서 충돌이 발생했을 때, 이를 해결하는 두 가지 방법에 대해 설명하시오.
- 해설: 해쉬 테이블에서 충돌을 해결하는 대표적인 두 가지 방법은 체이닝(Chaining)과 오픈 어드레싱(Open Addressing)이다. 체이닝은 각 해쉬 버킷에 연결 리스트를 사용하여 여러 키-값 쌍을 저장하는 방법이며, 오픈 어드레싱은 빈 버킷을 찾을 때까지 다음 버킷으로 이동하는 방법을 사용한다.

3.**그래프의 표현**

- 문제: 그래프를 표현하는 두 가지 주요 방법에 대해 설명하시오.
- 해설: 그래프를 표현하는 두 가지 주요 방법은 인접 행렬(Adjacency Matrix)과 인접 리스트(Adjacency List)이다. 인접 행렬은 2차원 배열을 사용하여 노드 간의 연결 관계를 표현하며, 인접 리스트는 각 노드에 연결된 노드의 목록을 저장하는 방법을 사용한다.

4.**자료구조의 선택**

- 문제: 검색 연산이 빈번하게 일어나는 경우와 삽입 및 삭제 연산이 빈번하게 일어나는 경우에 적합한 자료구조를 각각 선택하시오.
- 해설: 검색 연산이 빈번하게 일어나는 경우 배열이나 해쉬 테이블이 적합하며, 삽입 및 삭제 연산이 빈번하게 일어나는 경우 연결 리스트나 스택, 큐와 같은 동적 자료구조가 적합하다.

5.**동적 프로그래밍**

- 문제: 동적 프로그래밍의 기본 원칙과 그 특징에 대해 설명하시오.
- 해설: 동적 프로그래밍은 복잡한 문제를 작은 부분 문제로 나누어 해결하는 방법론입니다. 이 방법은 두 가지 핵심 원칙에 기반한다. 첫째, 작은 부분 문제의 해결책을 메모이제이션을 통해 저장하고, 이를 재사용하여 중복 계산을 피한다. 둘째, 주어진 문제를 최적 부분 구조로 나누어 각 부분 문제의 최적 해결책을 결합하여 전체 문제의 최적 해결책을 구한다. 동적 프로그래밍은 효율적인 계산을 위해 이전에 계산한 결과를 재사용하므로 계산 시간을 크게 줄일 수 있다.

6.**B-트리**

- 문제: B-트리의 정의와 그 특징에 대해 설명하시오.
- 해설: B-트리는 균형잡힌 트리 자료구조로, 데이터베이스나 파일 시스템에서 대용량 데이터의 저장 및 검색을 위해 사용된다. B-트리의 모든 노드는 일정한 범위의 자식 노드 수를 가지며, 트리의 높이는 로그 시간에 비례한다. 이러한 특징 덕분에 B-트리는 대량의 데이터를 효율적으로 관리할 수 있다. 또한, B-트리는 노드 내의 키가 정렬된 상태로 유지되며, 삽입, 삭제 시에도 트리의 균형을 유지하기 위한 재구성 작업을 수행한다.

7.**데드락(Deadlock)**

- 문제: 데드락의 발생 조건 네 가지와 이를 해결하는 방법에 대해 설명하시오.
- 해설: 데드락은 여러 프로세스나 스레드가 서로 자원을 기다리며 진행을 멈추는 현상을 말한다. 데드락의 발생 조건은 상호 배제, 점유 및 대기, 비선점, 순환 대기 네 가지이다. 이 조건들 중 하나라도 만족되지 않으면 데드락은 발생하지 않는다. 데드락을 해결하는 방법에는 예방, 회피, 탐지 및 복구, 무시 등이 있다. 예방은 데드락 발생 조건 중 하나를 제거하는 것이며, 회피는 데드락의 가능성을 감지하고 회피하는 것이다. 탐지 및 복구는 데드락을 탐지한 후 적절한 조치를 취하는 것이며, 무시는 데드락이 발생해도 무시하고 시스템을 계속 운영하는 것이다.

8.**시간 복잡도와 공간 복잡도**

- 문제: 시간 복잡도와 공간 복잡도의 차이점과 각각을 평가하는 중요성에 대해 설명하시오.
- 해설: 시간 복잡도는 알고리즘이 실행되는 데 걸리는 시간을 나타내는 지표이며, 공간 복잡도는 알고리즘이 실행되는 데 필요한 메모리 공간을 나타내는 지표이다. 시간 복잡도는 알고리즘의 효율성을 평가하는 데 중요하며, 공간 복잡도는 메모리 사용량을 최소화하여 자원을 효율적으로 사용하는 데 중요하다. 특히, 제한된 메모리 환경에서는 공간 복잡도의 중요성이 더욱 커진다. 따라서, 알고리즘을 설계할 때는 시간과 공간의 트레이드오프를 고려하여 최적의 해결책을 찾아야 한다.

9.**그래프와 트리의 차이**

- 문제: 그래프와 트리의 기본적인 차이점에 대해 설명하시오.
- 해설: 그래프와 트리는 노드와 간선으로 구성된 자료구조이지만, 여러 가지 차이점을 가진다. 트리는 그래프의 특별한 형태로, 사이클이 없는 연결 그래프이다. 트리는 하나의 루트 노드를 가지며, 모든 노드는 유일한 부모 노드를 가진다. 반면, 그래프는 여러 개의 분리된 부분 그래프를 포함할 수 있으며, 사이클을 가질 수 있다. 또한, 트리는 노드 수보다 간선 수가 하나 적지만, 그래프는 이러한 제한이 없다.

10.**스택과 큐의 운영 원칙**

- 문제: 스택과 큐의 데이터 추가 및 제거 원칙에 대해 설명하시오.
- 해설: 스택은 후입선출(LIFO) 원칙에 따라 동작하는 자료구조이다. 데이터는 스택의 상단에 추가되며, 제거될 때도 상단에서 제거된다. 반면, 큐는 선입선출(FIFO) 원칙에 따라 동작한다. 데이터는 큐의 뒤쪽에 추가되며, 제거될 때는 앞쪽에서 제거된다. 이 두 자료구조는 이러한 원칙에 따라 다양한 응용 분야에서 사용된다.

11.**정규 표현식의 활용**

- 문제: 정규 표현식의 주요 활용 분야와 그 중요성에 대해 설명하시오.

- 해설: 정규 표현식은 문자열의 패턴을 표현하고 검색하는 데 사용되는 강력한 도구이다. 주요 활용 분야로는 데이터 검증, 문자열 검색 및 치환, 문법 분석 등이 있다. 정규 표현식을 사용하면 복잡한 문자열 패턴도 간결하게 표현할 수 있어, 텍스트 처리 작업의 효율성과 정확성을 크게 향상시킬 수 있다.

12. **빅 오 표기법의 중요성**

- 문제: 빅 오 표기법의 정의와 그 중요성에 대해 설명하시오.
- 해설: 빅 오 표기법은 알고리즘의 성능을 나타내는 표기법으로, 최악의 경우의 시간 복잡도를 표현한다. 이 표기법은 알고리즘의 효율성을 객관적으로 비교할 수 있게 해주며, 대규모 데이터에 대한 처리 속도 예측에 중요한 역할을 한다. 빅 오 표기법을 통해 알고리즘의 성능을 빠르게 평가하고, 최적의 알고리즘을 선택할 수 있다.

13. **정규 표현식의 활용**

- 문제: 정규 표현식의 주요 활용 분야와 그 중요성에 대해 설명하시오.
- 해설: 정규 표현식은 문자열의 패턴을 표현하고 검색하는 데 사용되는 강력한 도구이다. 주요 활용 분야로는 데이터 검증, 문자열 검색 및 치환, 문법 분석 등이 있다. 정규 표현식을 사용하면 복잡한 문자열 패턴도 간결하게 표현할 수 있어, 텍스트 처리 작업의 효율성과 정확성을 크게 향상시킬 수 있다.

14. **힙 구조의 이해**

- 문제: 힙(Heap)의 기본적인 특성과 최대 힙(Max Heap)과 최소 힙(Min Heap)의 차이점을 설명하시오.
- 해설: 힙은 완전 이진 트리의 일종으로, 각 노드의 키 값이 그 노드의 자식들의 키 값보다 크거나 같은 경우를 최대 힙, 반대로 각 노드의 키 값이 그 노드의 자식들의 키 값보다 작거나 같은 경우를 최소 힙이라고 합니다. 최대 힙에서는 루트 노드가 가장 큰 값을, 최소 힙에서는 루트 노드가 가장 작은 값을 가집니다. 힙은 우선순위 큐를 구현하는 데 주로 사용됩니다.

15. **그래프 탐색 방법**

- 문제: 깊이 우선 탐색(DFS)과 너비 우선 탐색(BFS)의 기본 원리와 차이점을 설명하시오.
- 해설: 깊이 우선 탐색(DFS)은 시작 노드에서 시작하여 깊게 들어가면서 탐색하는 방법입니다. 스택 또는 재귀를 사용하여 구현할 수 있습니다. 반면, 너비 우선 탐색(BFS)은 시작 노드에서 가까운 노드부터 차례대로 탐색하는 방법입니다. 큐를 사용하여 구현합니다. DFS는 경로 탐색에 유용하며, BFS는 최단 경로를 찾는 데 사용됩니다.

16. **허프만 코딩의 원리**

- 문제: 허프만 코딩의 원리와 그 특징에 대해 설명하시오.

- 해설: 허프만 코딩은 손실 없는 데이터 압축 방법 중 하나로, 문자 또는 심볼의 빈도에 따라 가변 길이의 코드를 할당하는 방식입니다. 가장 빈도가 높은 문자에는 짧은 코드를, 빈도가 낮은 문자에는 긴 코드를 할당합니다. 이 방식을 사용하면 데이터를 효율적으로 압축할 수 있습니다. 허프만 트리를 사용하여 코드를 생성하며, 이 트리는 모든 문자를 단말 노드로 가지는 이진 트리입니다.

17.**연결 리스트와 배열의 차이**

- 문제: 연결 리스트의 기본 구조와 배열과의 차이점을 설명하시오.
- 해설: 연결 리스트는 노드들이 포인터를 사용하여 직접 연결되어 있는 선형 자료 구조입니다. 각 노드는 데이터와 다음 노드를 가리키는 포인터로 구성됩니다. 배열과의 주요 차이점은, 연결 리스트는 동적으로 크기를 조절할 수 있으며, 중간에 노드를 삽입하거나 삭제하는 연산이 효율적입니다. 반면, 배열은 연속된 메모리 공간에 데이터를 저장하므로, 중간의 데이터를 삽입하거나 삭제할 때 데이터를 이동해야 하는 단점이 있습니다.

18.**해시 테이블의 작동 원리**

- 문제: 해시 테이블의 작동 원리와 충돌 해결 방법 중 하나를 설명하시오.
- 해설: 해시 테이블은 키를 값에 매핑하는 자료 구조로, 해시 함수를 사용하여 키를 해시 코드로 변환하고, 이 해시 코드를 인덱스로 사용하여 값을 저장하거나 검색합니다. 충돌이 발생할 경우, 여러 가지 방법으로 해결할 수 있는데, 하나의 방법은 체이닝입니다. 체이닝은 같은 해시 코드를 가진 항목들을 연결 리스트로 관리하는 방법입니다.

19.**병합 정렬의 특징**

- 문제: 병합 정렬(Merge Sort)의 기본 원리와 그 특징에 대해 설명하시오.
- 해설: 병합 정렬은 분할 정복 알고리즘의 일종으로, 배열을 반으로 나누어 각각을 정렬한 후, 두 개의 정렬된 배열을 병합하는 방식으로 동작합니다. 이 과정을 재귀적으로 반복하여 전체 배열을 정렬합니다. 병합 정렬의 시간 복잡도는 O(n log n)이며, 안정적인 정렬 알고리즘입니다. 하지만 추가적인 메모리를 필요로 하는 단점이 있습니다.

20.**이진 검색 트리의 특징**

- 문제: 이진 검색 트리(Binary Search Tree, BST)의 기본 원리와 그 특징에 대해 설명하시오.
- 해설: 이진 검색 트리(BST)는 특정 규칙에 따라 데이터를 저장하는 트리 구조입니다. 각 노드는 왼쪽 자식 노드에는 자신보다 작은 값, 오른쪽 자식 노드에는 자신보다 큰 값을 가집니다. 이 규칙 덕분에 BST에서 데이터를 검색할 때, 원하는 데이터를 찾을 때까지 반씩 줄여나가며 검색할 수 있습니다. 이러한 구조는 데이터의 삽입, 삭제, 검색 연산을 로그 시간에 수행할 수 있게 해줍니다.

21.**동적 배열의 작동 원리**

- 문제: 동적 배열(Dynamic Array)의 작동 원리와 그 장점에 대해 설명하시오.
- 해설: 동적 배열은 초기에 정해진 크기를 초과하는 요소가 추가될 때마다 자동으로 크기를 조절하는 배열입니다. 일반적으로 현재 크기의 두 배로 확장하며, 이 과정을 재할당이라고 합니다. 동적 배열의 주요 장점은 크기 제한 없이 데이터를 저장할 수 있으며, 연속된 메모리 공간을 사용하기 때문에 인덱싱이 빠릅니다.

22.**그리디 알고리즘의 원리**

- 문제: 그리디 알고리즘(Greedy Algorithm)의 기본 원리와 그 특징에 대해 설명하시오.
- 해설: 그리디 알고리즘은 매 순간 최적의 선택을 하여 문제의 해결을 추구하는 방법입니다. 이 방법은 간단하고 빠르게 문제를 해결할 수 있지만, 항상 최적의 해를 찾을 수 있는 것은 아닙니다. 그리디 알고리즘은 동전 거스름돈, 작업 스케줄링 등 다양한 문제에서 활용됩니다.

23.**다익스트라 알고리즘의 활용**

- 문제: 다익스트라 알고리즘(Dijkstra's Algorithm)의 원리와 그 활용 분야에 대해 설명하시오.
- 해설: 다익스트라 알고리즘은 시작 노드에서 다른 모든 노드까지의 최단 경로를 찾는 알고리즘입니다. 이 알고리즘은 가중치가 있는 그래프에서 사용되며, 우선순위 큐를 사용하여 효율적으로 작동합니다. 주로 GPS, 네트워크 라우팅 등에서 활용되며, 실제 생활에서도 다양한 분야에서 경로 찾기 문제를 해결하는 데 사용됩니다.

24.**레드-블랙 트리의 특징**

- 문제: 레드-블랙 트리(Red-Black Tree)의 기본 원리와 그 특징에 대해 설명하시오.
- 해설: 레드-블랙 트리는 이진 검색 트리의 확장형으로, 트리의 균형을 유지하기 위한 특별한 규칙을 가지고 있습니다. 노드는 레드 또는 블랙 중 하나의 색상을 가지며, 이 규칙들을 통해 트리의 높이가 로그 시간 내에 유지되도록 합니다. 이로 인해 검색, 삽입, 삭제 연산이 효율적으로 수행됩니다.

25.**플로이드-워셜 알고리즘의 활용**

- 문제: 플로이드-워셜 알고리즘(Floyd-Warshall Algorithm)의 원리와 그 활용 분야에 대해 설명하시오.
- 해설: 플로이드-워셜 알고리즘은 그래프 내의 모든 노드 쌍 간의 최단 경로를 찾는 알고리즘입니다. 이 알고리즘은 3중 반복문을 사용하여 모든 노드 쌍에 대해 최단 경로를 계산합니다. 주로 교통 네트워크 최적화, 도시 간 거리 계산 등에서 활용되며, 큰 그래프에서는 시간 복잡도가 높을 수 있습니다.

26.**트라이 자료구조의 특징**

- 문제: 트라이(Trie) 자료구조의 기본 원리와 그 특징에 대해 설명하시오.
- 해설: 트라이는 문자열을 저장하고 검색하는 데 특화된 트리 기반의 자료구조입니다. 각 노드는 알파벳 하나와 그 자식 노드들의 참조를 저장합니다. 이 구조를 사용하면 문자열 검색 및 삽입 연산을 문자열의 길이에 비례하여 효율적으로 수행할 수 있습니다. 주로 사전 구현이나 자동 완성 기능에서 활용됩니다.

27.**퀵 정렬의 작동 원리**

- 문제: 퀵 정렬(Quick Sort)의 기본 원리와 그 특징에 대해 설명하시오.
- 해설: 퀵 정렬은 분할 정복 전략을 사용하는 정렬 알고리즘입니다. 배열에서 피벗을 선택하고, 피벗보다 작은 요소들은 왼쪽으로, 큰 요소들은 오른쪽으로 분리합니다. 이후 왼쪽과 오른쪽 부분 배열을 재귀적으로 정렬합니다. 퀵 정렬은 평균적으로 빠른 정렬 속도를 보이지만, 이미 정렬된 배열에서는 최악의 성능을 보일 수 있습니다.

28.**백트래킹의 원리**

- 문제: 백트래킹(Backtracking)의 기본 원리와 그 활용 분야에 대해 설명하시오.
- 해설: 백트래킹은 가능한 모든 해를 탐색하는 알고리즘입니다. 해결책의 후보를 구축해 나가다가, 후보가 유망하지 않다고 판단되면 이전 단계로 돌아가 다른 경로를 탐색합니다. 이 방법은 조합, 순열, 미로 찾기, N-퀸 문제 등에서 활용되며, 모든 가능한 해를 탐색하므로 시간 복잡도가 높을 수 있습니다.

29.**NP-완전 문제의 특징**

- 문제: NP-완전(NP-Complete) 문제의 정의와 그 특징에 대해 설명하시오.
- 해설: NP-완전 문제는 NP 클래스에 속하면서, 다른 NP 문제들이 다항 시간 내에 이 문제로 변환될 수 있는 문제를 의미합니다. 즉, NP-완전 문제를 다항 시간 내에 해결할 수 있는 알고리즘이 존재한다면, 모든 NP 문제도 다항 시간 내에 해결될 수 있습니다. 현재까지 NP-완전 문제를 효율적으로 해결하는 알고리즘이 알려져 있지 않습니다.

30.**캐시의 작동 원리**

- 문제: 캐시(Cache)의 기본 원리와 그 중요성에 대해 설명하시오.
- 해설: 캐시는 자주 사용되는 데이터나 연산 결과를 빠르게 접근할 수 있는 고속의 저장소에 저장하는 기술입니다. 주 메모리보다 빠르지만 용량이 작은 저장소를 사용하여, 자주 접근하는 데이터를 빠르게 가져올 수 있게 합니다. 이로 인해 전체 시스템의 성능 향상을 기대할 수 있습니다.

31.**동적 계획법과 분할 정복의 차이**

- 문제: 동적 계획법(Dynamic Programming)과 분할 정복(Divide and Conquer)의 기본 원리와 그 차이점에 대해 설명하시오.

- 해설: 동적 계획법과 분할 정복은 모두 복잡한 문제를 작은 부분 문제로 나누어 해결하는 전략을 사용합니다. 동적 계획법은 중복되는 부분 문제의 해결을 피하기 위해 이전에 계산한 결과를 저장하고 재사용하는 반면, 분할 정복은 문제를 독립적인 부분 문제로 나누어 해결합니다. 두 방법의 주요 차이점은 동적 계획법이 중복된 부분 문제의 해결을 피하기 위해 메모이제이션을 사용한다는 것입니다.

32.**BFS와 DFS의 사용 사례**

- 문제: 너비 우선 탐색(BFS)과 깊이 우선 탐색(DFS)의 주요 사용 사례에 대해 설명하시오.
- 해설: BFS와 DFS는 그래프 탐색 알고리즘입니다. BFS는 시작 노드에서 가장 가까운 노드부터 탐색하며, 최단 경로 문제나 네트워크 라우팅에서 활용됩니다. DFS는 깊게 들어가면서 노드를 탐색하며, 경로 탐색, 퍼즐 문제, 백트래킹 등에서 활용됩니다.

33.**자료구조의 중요성**

- 문제: 자료구조의 중요성과 그 활용 분야에 대해 설명하시오.
- 해설: 자료구조는 데이터를 효율적으로 저장하고 처리하는 방법을 제공합니다. 올바른 자료구조의 선택은 알고리즘의 성능을 크게 향상시킬 수 있습니다. 자료구조는 데이터베이스, 운영 체제, 그래픽, 인공 지능 등 다양한 분야에서 필수적으로 활용되며, 효율적인 자료구조의 사용은 시스템의 전반적인 성능에 큰 영향을 미칩니다.

34.**힙 정렬의 특징**

- 문제: 힙 정렬(Heap Sort)의 기본 원리와 그 특징에 대해 설명하시오.
- 해설: 힙 정렬은 최대 힙 구조를 사용하여 배열을 정렬하는 방법입니다. 힙 구조를 만든 후, 루트 노드를 배열의 마지막 요소와 교환하고 힙 크기를 줄여가며 정렬을 수행합니다. 힙 정렬의 시간 복잡도는 O(n log n)입니다.

35.**이진 검색의 작동 원리**

- 문제: 이진 검색(Binary Search)의 기본 원리와 그 특징에 대해 설명하시오.
- 해설: 이진 검색은 정렬된 배열에서 원하는 값을 로그 시간 내에 찾는 방법입니다. 중간 값과 찾고자 하는 값을 비교하여 검색 범위를 반으로 줄여나가며 값을 찾습니다.

36.**자료구조의 선택 기준**

- 문제: 특정 상황에 적합한 자료구조를 선택하는 기준에 대해 설명하시오.
- 해설: 자료구조의 선택은 데이터의 크기, 연산의 빈도, 메모리 사용량, 연산의 속도 등 여러 요소를 고려하여 결정됩니다. 예를 들어, 검색 연산이 빈번한 경우 해시 테이블이 적합하며, 삽입 및 삭제 연산이 빈번한 경우 연결 리스트가 적합합니다.

37.**자료구조의 중위순회와 전위순회**

- 문제: 이진 트리에 대한 전위(pre-order) 및 중위(in-order) 순회시 방문 순서가 주어졌을 때, 이를 기반으로 이진 트리의 특성을 파악하는 방법에 대해 설명하시오.
- 해설: 전위 순회는 루트 노드를 먼저 방문하고, 왼쪽 서브트리, 그리고 오른쪽 서브트리 순으로 방문합니다. 중위 순회는 왼쪽 서브트리를 먼저 방문하고, 루트 노드, 그리고 오른쪽 서브트리 순으로 방문합니다. 이러한 순회 방식을 통해 트리의 구조와 노드 간의 관계를 파악할 수 있습니다.

38.**덱(deque)의 구현**

- 문제: 덱(deque)을 이중 연결 리스트와 단일 연결 리스트로 각각 구현하였을 때, 덱의 후방에서 요소를 삭제하는 연산의 시간 복잡도 차이에 대해 설명하시오.
- 해설: 이중 연결 리스트를 사용한 덱에서는 후방에서의 요소 삭제 연산이 O(1)의 시간 복잡도를 가집니다. 반면, 단일 연결 리스트를 사용한 덱에서는 후방에서의 요소 삭제 연산이 O(n)의 시간 복잡도를 가질 수 있습니다. 이는 단일 연결 리스트에서는 마지막 노드의 이전 노드에 접근하기 위해 리스트를 순회해야 하기 때문입니다.

39.**퀵 정렬의 특성**

- 문제: 퀵 정렬에 대한 설명 중 피벗 선택 방법에 따른 최악의 시나리오와 안전한 선택 방법에 대해 설명하시오.
- 해설: 퀵 정렬에서 피벗 선택은 정렬 성능에 큰 영향을 미칩니다. 배열의 마지막 원소를 피벗으로 선택하는 경우, 이미 정렬된 배열에서는 최악의 성능을 보입니다. 이러한 최악의 시나리오를 피하기 위해 배열에서 랜덤하게 피벗을 선택하는 것이 안전한 방법으로 간주됩니다.

40.**스택의 연산 순서**

- 문제: 주어진 스택 연산 순서를 바탕으로 출력되는 결과의 순서를 예측하시오.
- 해설: 스택은 후입선출(LIFO)의 원칙을 따르므로, 가장 마지막에 push된 요소가 가장 먼저 pop됩니다. 주어진 연산 순서를 따라가며 push와 pop 연산을 수행하면, 출력되는 결과의 순서를 예측할 수 있습니다.