**프로그램 개요**

**최소 힙 그리고 삭제키를 통해 프림 알고리즘 구현하기**

**최솟힙이란?**

**최솟값을 찾기위한 방법 트리구조**

**텍스트, 스크린샷, 소프트웨어이(가) 표시된 사진

자동 생성된 설명**

**Chat gpt 사용 출처**

**프림알고리즘 구현을 공부하면서 과제에선 인접 리스트로 구현하는데 인접행렬도 구현 할수있습니다. 하지만 인접리스트를 사용하는 이유는 알게되었습니다.**

**인접 리스트와 인접 행렬은 그래프를 표현하는 두 가지 주요 방법입니다. 각 방법은 그래프의 구조와 특성에 따라 선택됩니다. 아래는 인접 리스트와 인접 행렬의 주요 차이점을 설명한 것입니다:**

**1. \*\*데이터 구조\*\*:**

**- \*\*인접 리스트\*\*: 각 정점마다 해당 정점과 직접 연결된 정점들의 리스트를 유지합니다. 간선 정보를 정점마다 연결 리스트로 저장합니다.**

**- \*\*인접 행렬\*\*: 2D 배열을 사용하여 각 쌍의 정점 사이의 연결 관계를 나타냅니다. 행과 열은 각 정점을 나타내며, 행과 열 사이의 값은 해당 두 정점 사이의 간선 유무를 나타냅니다.**

**2. \*\*메모리 사용\*\*:**

**- \*\*인접 리스트\*\*: 희소 그래프(Sparse Graphs)에 적합합니다. 그래프의 간선 수가 적을 때 메모리를 효율적으로 사용할 수 있습니다. 인접 리스트는 간선 수에 비례하는 메모리를 사용합니다.**

**- \*\*인접 행렬\*\*: 밀도가 높은 그래프(Dense Graphs)에 적합합니다. 그래프의 간선 수가 많을 때 메모리를 효율적으로 사용할 수 있습니다. 인접 행렬은 정점 수의 제곱에 비례하는 메모리를 사용합니다.**

**3. \*\*간선 탐색 시간\*\*:**

**- \*\*인접 리스트\*\*: 특정 정점에서 연결된 모든 간선을 찾는 데 O(차수)의 시간이 걸립니다. 여기서 "차수"는 해당 정점에 연결된 간선의 수를 의미합니다.**

**- \*\*인접 행렬\*\*: 두 정점 간 연결 여부를 확인하는 데 O(1)의 시간이 걸립니다.**

**4. \*\*메모리 요구 사항\*\*:**

**- \*\*인접 리스트\*\*: 메모리 사용량은 간선의 수에 의해 결정됩니다.**

**- \*\*인접 행렬\*\*: 메모리 사용량은 정점의 수에 의해 결정됩니다.**

**5. \*\*연산\*\*:**

**- \*\*인접 리스트\*\*: 특정 간선을 찾거나 삭제하기가 상대적으로 빠릅니다.**

**- \*\*인접 행렬\*\*: 간선을 추가하거나 삭제하기가 상대적으로 느립니다.**

**6. \*\*효율성\*\*:**

**- \*\*인접 리스트\*\*: 희소 그래프에서 효율적이며 메모리를 절약할 수 있습니다.**

**- \*\*인접 행렬\*\*: 밀도가 높은 그래프에서 연결 관계를 빠르게 확인할 수 있으나 메모리 소비가 큽니다.**

**따라서, 인접 리스트는 희소 그래프에서 효과적이고 메모리 효율적이며, 인접 행렬은 밀도가 높은 그래프에서 연결 관계 확인이 빠릅니다. 그래프의 특성과 사용 사례에 따라 어떤 표현 방식을 선택할지 결정해야 합니다.// Chat gpt 도움**

**왜 인접리스트를 사용하는지 알게 되었습니다.**

**프로그램 동작설계 설명**

**일단 프림알고리즘는 가중치로 최소하는 최소신장트리를 구현하는 알고리즘 중하나입니다.**

**1. 구조체 정의: 코드는 다양한 구조체를 정의합니다. struct Node는 그래프의 각 정점에서 연결된 다른 정점과 그 가중치를 저장합니다. `struct Graph`는 그래프 자체를 나타내며, 인접 리스트를 사용하여 정점 간 연결 정보를 저장합니다. `struct MinHeap`은 최소 힙을 나타내며, 최소 힙 노드와 관련된 정보를 저장합니다.**

**2. 그래프 생성: create Graph 함수는 주어진 정점 수를 기반으로 그래프를 생성하고 초기화합니다. create Node 함수는 새로운 노드를 생성하며, create Min HeapNode함수는 최소 힙 노드를 생성합니다.**

**3. 최소 힙 생성: create Minheap 함수는 최소 힙을 생성하고 초기화합니다. 이 힙은 프림 알고리즘에서 사용됩니다.**

**4. 간선 추가: `addEdge` 함수를 사용하여 그래프에 간선을 추가합니다. 간선의 정보가 인접 리스트에 저장됩니다.**

**5. 프림 알고리즘 실행: `primMST` 함수가 최소 신장 트리를 찾는 메인 로직을 수행합니다.**

**- 초기에 모든 정점의 키(key)를 무한대로 설정하고, 부모(parent) 배열을 -1로 초기화합니다.**

**- 시작 정점의 키를 0으로 설정하고, 최소 힙에 해당 정보를 추가합니다.**

**- 최소 힙에서 가장 작은 키를 가진 정점을 선택하고, 그 정점과 연결된 간선을 확인합니다.**

**- 새로운 정점과 연결된 간선의 가중치가 해당 정점의 현재 키보다 작으면 키와 부모를 업데이트하고 최소 힙을 재정렬합니다.**

**- 이 과정을 모든 정점을 방문하고, 최소 신장 트리를 형성할 때까지 반복합니다.**

**6. 최소 신장 트리 출력: 최소 신장 트리가 구성되면, 부모 배열을 통해 트리의 간선 정보를 출력합니다.**

**프림 알고리즘은 각 단계에서 현재 MST에 속하지 않는 정점 중에서 가장 가까운 정점을 선택하여 트리를 확장하는 방식으로 동작합니다. 이 과정을 통해 최소 신장 트리를 구성하고, 최종 결과로 MST의 간선 정보를 출력합니다.**

**출력결과 보기**

**텍스트, 폰트, 스크린샷, 블랙이(가) 표시된 사진

자동 생성된 설명**

**결과값 분석하기텍스트, 친필, 화이트보드이(가) 표시된 사진

자동 생성된 설명**

**결과값을 빨간색 펜으로 나타냈습니다.**