과제 #04

데이터사이언스를 위한 컴퓨팅 1 (2022년도 1학기, M3239.005500)

Due: 2022년 05월 09일(월) 23시 59분

1 Graph Repesentation (20점)

C++ Class와 STL (Standard Template Library)을 사용하여 graph를 표현할 것이다. `Graph` class 와 `Vertex` class가 다음과 같은 조건을 만족하도록 `graph.cpp` 파일에 코드를 작성하라.

1.1 Vertex class의 member data

- std::string name_
  Vertex의 이름
- std::vector<Vertex*> adjacency_list_
  Vertex의 adjacency list

1.2 Vertex class의 member function

- Vertex(std::string name)
  Vertex의 constructor, 인자로 vertex의 이름을 입력 받아 저장
- std::string GetName()
  Vertex의 이름을 리턴
- std::vector<Vertex*> GetAdjacencyList()
  Vertex의 adjacency list를 리턴
- void AddToAdjacentList(Vertex* vertex)
  Vertex의 adjacency list에 입력 받은 인자를 추가

1.3 Graph class의 member data

- std::vector<Vertex*> vertices_
  Graph 내의 모든 vertex instance를 저장해두는 리스트

1.4 Graph class의 member function

- Vertex* GenVertex(std::string name)
  Vertex 인스턴스를 생성, vertices_ member data에 생성한 인스턴스를 저장
- void GenEdge(Vertex* start, Vertex* end)
  start vertex의 adjacency list에 end vertex를 추가
- std::vector<Vertex*> GetVertices()
  Graph 내의 모든 vertex 인스턴스를 담고 있는 리스트를 반환

- size_t GetNumVertices()
  Graph 내의 모든 vertex 개수를 리턴

- void PrintGraph()
  Graph 내의 모든 vertex의 이름과 각각의 adjacency list를 출력

모든 코드를 처음부터 작성할 필요는 없다. 뼈대 코드가 주어지며 TODO로 적힌 부분의 힌트를 토대로 프로그램을 완성하라. 실질적으로 구현해야 하는 것는 `GenEdge(...)` 함수와 `AddToAdjacentList(...)` 함수이다. 나머지 내용들은 모두 구현되어 있다. 프로그램 실행 시 아래 화면과 같은 결과를 나타내도록 작성한다.

```
============== Defined Graph ==============
Vertex u, Adjacency List: Vertex v, Vertex x
Vertex v, Adjacency List: Vertex y
Vertex w, Adjacency List: Vertex y, Vertex z
Vertex x, Adjacency List: Vertex v
Vertex y, Adjacency List: Vertex x
Vertex z, Adjacency List: Vertex z
```

2 Breadth-First Search (BFS) (40점)

앞서 완성한 `Graph` class를 사용하여 BFS 알고리즘을 구현할 것이다. 다음과 같은 조건을 만족하는 BFS 알고리즘을 `search.cpp` 파일에 작성하라.

- void BFS(Vertex* start, std::map<Vertex*, unsigned int> &distance);
  첫번째 인자로 BFS를 시작할 vertex 인스턴스를 입력 받음
  두번째 인자에 search한 vertex들의 distance를 저장

뼈대 코드가 주어지며 뼈대 코드에서 TODO로 적힌 부분의 힌트를 토대로 프로그램을 완성하라. 프로그램 실행 시 아래 화면과 같은 결과를 나타내도록 작성한다. 알고리즘을 실행한 결과를 화면에 출력하는 함수는 뼈대코드로 주어진다.

```
============== Defined Graph ==============
Vertex r, Adjacency List: Vertex s, Vertex v
Vertex s, Adjacency List: Vertex r, Vertex w
Vertex t, Adjacency List: Vertex w, Vertex x, Vertex u
Vertex u, Adjacency List: Vertex t, Vertex x, Vertex y
Vertex v, Adjacency List: Vertex r
Vertex w, Adjacency List: Vertex s, Vertex t, Vertex x
Vertex x, Adjacency List: Vertex t, Vertex u, Vertex w, Vertex y
Vertex y, Adjacency List: Vertex u, Vertex x
=============== BFS Result ===============
Vertex r, Distance: 1
Vertex s, Distance: 0
Vertex t, Distance: 2
Vertex u, Distance: 3
Vertex v, Distance: 2
Vertex w, Distance: 1
Vertex x, Distance: 2
Vertex y, Distance: 3
```

3 Depth-First Search (DFS) (40점)

앞서 완성한 `Graph` class를 사용하여 DFS 알고리즘을 구현할 것이다. 다음과 같은 조건을 만족하는 DFS 알고리즘을 `search.cpp` 파일에 작성하라.

- void DFS(Vertex* vertex, unsigned int &timestamp, std::map<Vertex*, unsigned int> &discovery_time, std::map<Vertex*, unsigned int> &finishing_time);
  첫번째 인자로 DFS를 시작할 vertex 인스턴스를 입력 받음
  두번째 인자로 discovery time과 finishing time 기록을 위한 시작 time을 입력 받음
  세번째 인자에 search한 vertex들의 discovery time을 저장
  네번째 인자에 search한 vertex들의 finishing time을 저장

뼈대 코드가 주어지며 뼈대 코드에서 TODO로 적힌 부분의 힌트를 토대로 프로그램을 완성하라. 프로그램 실행 시 아래 화면과 같은 결과를 나타내도록 작성한다. 알고리즘을 실행한 결과를 화면에 출력하는 함수는 뼈대코드로 주어진다.

```
============== Defined Graph ==============
Vertex u, Adjacency List: Vertex v, Vertex x
Vertex v, Adjacency List: Vertex y
Vertex w, Adjacency List: Vertex y, Vertex z
Vertex x, Adjacency List: Vertex v
Vertex y, Adjacency List: Vertex x
Vertex z, Adjacency List: Vertex z
=============== DFS Result ===============
Vertex u, Discovery time: 1, Finishing time: 8
Vertex v, Discovery time: 2, Finishing time: 7
Vertex w, Discovery time: 9, Finishing time: 12
Vertex x, Discovery time: 4, Finishing time: 5
Vertex y, Discovery time: 3, Finishing time: 6
Vertex z, Discovery time: 10, Finishing time: 11
```

4 참고 사항

이번 과제에서는 STL의 vector, map, queue를 사용한다. 뼈대코드의 TODO에 적힌 힌트에 맞게 STL에서 제공하는 함수를 호출하면 되며, 필요한 함수 리스트는 아래의 링크를 참고한다.

- https://www.cplusplus.com/reference/vector/vector/
- https://www.cplusplus.com/reference/map/map/
- https://www.cplusplus.com/reference/queue/queue

이외에도 STL vector, map, queue로 검색하면 많은 사용 예제를 찾을 수 있으니 참고하여 프로그래밍 한다.

5 Submission Instruction

- `graph.cpp`, `search.cpp`를 한 파일로 압축하여 ETL에 제출한다.
- `main.cpp`, `graph.h`, `search.h`, `Makefile`은 수정할 수 없으며, 제출하지 않도록 한다.
- 첨부 파일명은 계정이름_HW04.zip으로 한다. (*e.g.*, `cfds999_HW04.zip`)
- 본인의 계정을 모를 경우 ETL 자료실에 업로드 되어있는 게시글에서 확인한다. 게시글에 본인의 이름이 적혀있지 않은 경우 조교에게 이메일로 문의한다.

- Grace day를 사용하려면 본인이 과제를 제출한 날에 조교에게 메일(`cfdsta@aces.snu.ac.kr`)로 알려야 한다. 메일 없이 제출만 한 경우 다음 과제를 위해 아낀 것으로 판단, 미제출 처리된다. 또한, grace day 사용 시에도 과제 제출은 이메일이 아닌 ETL을 통해 해야한다.
- 채점은 프로그램에 의해 기계적으로 처리되므로 위 사항을 지키지 않은 경우 채점 대상에서 누락되거나 감점을 받을 수 있다. 자세한 감점 기준은 ETL 공지사항을 참조한다.