Расчетно-графическая работа

по дисциплине «Алгоритмы и структуры данных»
направление 09.03.01 Информатика и вычислительная техника,
направление 09.03.04 Программная инженерия
3 курс, 5 семестр

Цель РГР – освоение технологии разработки комплексного программного обеспечения для решения задач в различных прикладных областях.

Задание 1

1) Спроектировать и реализовать шаблонный класс для коллекции «Простой граф» и использовать коллекцию для решения задач для неориентированных, ориентированных и взвешенных графов.

Разработать АТД «Простой граф».

Интерфейс АТД «Простой граф» включает операции:

Конструктор ( ) по умолчанию: создает пустой L - граф с нулевым числом вершин и ребер,
Конструктор (V, D, F) создает граф с **V** вершинами, без ребер, типа **D** (ориентированный / неориентированный), формы представления **F** (L- граф/M-граф),
Конструктор (V, E, D, F) создает граф с **V** вершинами, с **E** случайными ребрами, типа **D** (ориентированный / неориентированный), формы представления **F** (L- граф/M-граф),
Конструктор (G) - конструктор копирования создает объект – копию графа **G,**
Деструктор ( ) уничтожает внутренние структуры графа,
V ( ) - возвращает число вершин в графе,
E ( ) - возвращает число ребер в графе**,**
Directed ( ) - возвращает тип графа (ориентированный / неориентированный)
Dense ( ) - возвращает форму представления графа (L- граф / M- граф),
K ( ) - возвращает коэффициент насыщенности графа,
ToListGraph ( ) преобразует граф к L- графу,
ToMatrixGraph ( ) преобразует граф к M- графу,
InsertV ( ) добавляет вершину к графу и возвращает адрес дескриптора вновь созданной вершины,
DeleteV (*v*) - удаляет вершину из графа, заданную адресом дескриптора ***v*,**
InsertE(*v1, v2*) - добавляет ребро (***v1, v2***) к графу, соединяющую вершины, заданные адресами дескрипторов ***v1*** и ***v2,*** и возвращает адрес дескриптора вновь созданного ребра - ***e***,
DeleteE (*v1, v2*) удаляет ребро, соединяющее вершины, заданные адресами дескрипторов ***v1*** и ***v2,***
GetEdge (*v1, v2*) возвращает адрес дескриптора ребра соединяющего вершины, заданные дескрипторами ***v1*** и ***v2*,**

Разработать ассоциированные с графом типы:

АТД «Дескриптор вершины графа»

Дескриптор вершины содержит поля:
***name* –** имя вершины,
***data* –** данные, связанные с вершиной,
***index* –** индекс вершины в структуре графа или -1,

Интерфейс АТД «Дескриптор вершины графа» включает операции:
Конструктор ():поле ***name*** не определено, поле ***data*** не определено,
Конструктор (***name, data***): ***name*** - имя вершины, ***data*** - данные, связанные с вершиной,
GetName ( ) - возвращает имя вершины,
GetData ( ) - возвращает данные, связанные с вершиной,
SetName (name) – задает имя вершины,
SetData (*data*) – записывает данные ***data*** в дескриптор вершины.

АТД «Дескриптор ребра графа»

Дескриптор ребра содержит поля:
v1 - дескриптор вершины, из которой исходит ребро,
v2 - дескриптор вершины, в которую входит ребро,
w - вес ребра,
data - данные, связанные с ребром,

Интерфейс АТД «Дескриптор ребра графа» включает операции:
Конструктор (***v1, v2***): ***v1*** - дескриптор вершины, из которой исходит ребро, ***v2*** - дескриптор вершины, в которую входит ребро,
Конструктор (***v1, v2, w***): ***v1*** - дескриптор вершины, из которой исходит ребро, ***v2*** - дескриптор вершины, в которую входит ребро, ***w*** - вес ребра,
Конструктор (***v1, v2, w, data***): ***v1*** - дескриптор вершины, из которой исходит ребро, ***v2*** - дескриптор вершины, в которую входит ребро, ***w*** - вес ребра, ***data*** - данные, связанные с ребром
v1( ) - возвращает дескриптор вершины, из которой исходит ребро,
v2( ) - возвращает дескриптор вершины, в которую входит ребро,
GetW ( ) - возвращает вес ребра,
SetW (*w*) - изменение веса ребра,
GetData ( ) - возвращает данные, связанные с ребром,
SetData (***data***) - изменение данных, связанных с ребром.

АТД «Итератор вершин графа»

Интерфейс АТД «Итератор вершин графа» включает операции:
Конструктор ( ) - создает итератор вершин графа,
beg ( ) - возвращает итератор, установленный на первую вершину графа,
end ( ) - возвращает итератор, соответствующий окончанию переходов итератора,
operator ++ - переход к следующей вершине графа,
operator * - возвращает дескриптор вершины графа, на которую указывает итератор.

АТД «Итератор ребер графа»

Интерфейс АТД «Итератор ребер графа» включает операции:
Конструктор ( ) - создает итератор ребер графа,
beg ( ) - возвращает итератор, установленный на первое ребро графа,
end ( ) - возвращает итератор, соответствующий окончанию переходов итератора,
operator ++ - переход к следующему ребру графа,

operator * - возвращает дескриптор ребра графа, на которое указывает итератор.

АТД «Итератор исходящих ребер вершины»

Интерфейс АТД «Итератор исходящих ребер вершины» включает операции:
Конструктор (***v***) - создает итератор исходящих ребер графа для вершины, заданной дескриптором ***v,***
beg ( ) - возвращает итератор, установленный на первое исходящее ребро вершины,
end ( ) - возвращает итератор, соответствующий окончанию переходов итератора,
operator ++ - переход к следующему исходящему ребру,
operator * - возвращает дескриптор исходящего ребра вершины, на которое указывает итератор.

Задание 2

Спроектировать и реализовать шаблонный класс для ***АТД «Задача для невзвешенного графа»*** в соответствии с вариантом и использовать для решения задачи на неориентированном или ориентированном графе. Для программирования графа использовать разработанный в задании 1 АТД «Простой граф».

Интерфейс ***АТД «Задача для невзвешенного графа»*** включает операции:
Конструктор (***g***) - создает объект задачи 1, ассоциированный с графом ***g***, и выполняет решение задачи для графа ***g***,
Конструктор (*T*) - конструктор копирования создает копию объекта – задачи ***T,***
Деструктор ( ) - уничтожает внутренние структуры объекта задачи,
Set (***g***) – связывает объект задачи с графом ***g*** и выполняет решение задачи 1 для графа ***g***,
Restart ( ) – повторно выполняет решение задачи 1 для графа ***g***,
Result ( ) – возвращает результат решения задачи 1

Варианты задачи:

1) формирования двудольного неориентированного графа с раскраской в два цвета,
2) определение кратчайших по числу ребер путей между всеми парами вершин орграфа,
3) определение гамильтонова цикла в орграфе,
4) определение диаметра связного неориентированного графа,
5) определение последовательности вершин в результате прямой топологической сортировки ациклического орграфа DAG (сортировка - на основе очереди истоков),
6) построение остовного дерева орграфа в пределах заданной высоты за счёт добавления минимального числа рёбер,
7) определение последовательности вершин на основе топологической сортировки ациклического орграфа DAG на основе поиска в глубину,
8) поиск всех простых циклов, включающих заданную вершину орграфа,
9) формирование реберно-связного неориентированного графа,
10) поиск простого цикла заданной длины, включающего заданную вершину,
11) построение редуцированного графа сильносвязных компонент,
12) классификация всех рёбер относительно заданной вершины орграфа,
13) определение вершин, отстоящих на расстоянии ***d*** от заданной вершины (***d*** – число рёбер),
14) формирование двусвязного неориентированного графа,
15) определение эйлерова цикла в неориентированном графе,
16) формирование списка ребер неориентированного графа в порядке двухпроходного эйлерова цикла.

Задание 3

Спроектировать и реализовать шаблонный класс для ***АТД «Задача для взвешенного графа»»*** в соответствии с вариантом и использовать для решения задачи на взвешенном графе. Для программирования графа использовать разработанный в задании 1 АТД «Простой граф».

Интерфейс ***АТД «Задача для взвешенного графа»»*** включает операции:
Конструктор (g) - создает объект задачи 3, ассоциированный с графом g, и выполняет решение задачи для графа g,
Конструктор (*T*) - конструктор копирования создает копию объекта – задачи ***T,***
Деструктор ( ) - уничтожает внутренние структуры объекта задачи,
Set (g) – связывает объект задачи с графом g и выполняет решение задачи 3 для графа g,
Restart ( ) – повторно выполняет решение задачи 3 для графа g,
Result( ) – возвращает результат решения задачи 3

Варианты задачи:

1) определение диаметра и пути, соответствующего диаметру, на основе алгоритма Флойда,
2) приведение отрицательной весовой функции орграфа к положительной методом Джонсона,
3) определение входящих в заданную вершину кратчайших путей на основе алгоритма Дейкстры,
4) определение центра взвешенного орграфа на основе алгоритма Беллмана – Форда.
5) разбиение вершин взвешенного неориентированного графа на кластеры, объединяющие вершины ребрами с длиной, большей d. Для разбиения использовать алгоритм Крускалла,
6) определение периферии взвешенного орграфа на основе алгоритма Дейкстры,
7) определение списка вершин, которые лежат в пределах заданного расстояния d от заданной вершины взвешенного орграфа с отрицательной весовой функции на основе алгоритма Беллмана-Форда,
8) определение первого по величине (по количеству вершин) кластера в неориентированном взвешенном графе, вершины которого объединены ребрами с длиной, большей d. Для определения кластера использовать алгоритм Прима,
9) определение периферии взвешенного орграфа на основе алгоритма Флойда,
10) нахождения ребра (ребер), устранение которых вызывает максимальное возрастание кратчайшего пути из вершины u в вершину v во взвешенном орграфе на основе алгоритма Дейкстры,
11) определения кратчайших путей для всех пар вершин взвешенного орграфа на основе алгоритма Беллмана-Форда,
12) построение минимального остовного дерева относительно заданной взвешенного неориентированного графа методом Прима,
13) определение радиуса и соответствующего радиусу пути взвешенного орграфа на основе алгоритма Дейкстры,
14) определение центра взвешенного орграфа на основе алгоритма Флойда,
15) определение эксцентриситета заданной вершины взвешенного орграфа с отрицательной весовой функции на основе алгоритма Беллмана-Форда,
16) формирование остовного дерева взвешенного неориентированного графа с суммарным весом ребер, ближайшим к заданному весу d. Для формирования дерева использовать алгоритм Крускалла,

Методические указания.

1) Граф реализуется в виде класса «Простой граф» и ассоциированных классов «Дескриптор вершины графа», «Дескриптор ребра графа», «Итератор вершин графа», «Итератор ребер графа», «Итератор исходящих ребер вершины.

2) Объект «Дескриптор вершины графа» реализуется в виде шаблонного класса. Параметрами шаблона являются тип, задающий имя вершины, и тип данных, связанных с вершиной.

3) Объект «Дескриптор ребра графа» реализуется в виде шаблонного класса. Параметрами шаблона являются тип «Дескриптор вершины графа», тип, задающий вес ребра, и тип данных, связанных с ребром.

4) Объект «Простой граф» реализуется в виде шаблонного класса. Параметрами шаблона являются типы «Дескриптор вершины графа» и «Дескриптор ребра графа».

5) Ассоциированные классы («Итератор вершин графа», «Итератор ребер графа», «Итератор исходящих ребер вершины») реализовать, как вложенные в класс «Простой граф».

6) Для реализации различных форм представления графа используются вспомогательные классы «L – граф» и «M – граф». Рекомендуется объединить эти классы в иерархию с использованием наследования и полиморфизма.

7) Объект «Простой граф» содержит вектор дескрипторов вершин, вставленных в граф.

8) Объект «Простой граф» содержит структуру графа в виде списков смежности (L-граф) или матрицы смежности(M-граф). Элементы списков смежности или матрицы смежности содержат дескрипторы ребер, вставленных в граф.

9) Для согласования систем идентификации вершин в прикладной программе (имена вершин) и в объекте «Простой граф» (адреса дескрипторов вершин) рекомендуется использовать объект «Отображение имени на адрес дескриптора». Объект реализуется в виде шаблонного класса. Параметрами шаблона являются тип имени вершины графа и указатель на «Дескриптор вершины графа». Такой объект обеспечивает прямой доступ по имени вершины к адресу дескриптора вершины. Объект - отображение можно реализовать на базе ассоциативной структуры (сбалансированное дерево, хеш-таблица), объекта библиотеки STL - map или hash_map, где ключом является имя вершины и т.п.

10) Для реализации задач 1 - 2, заданных в варианте задания, разработать классы – клиенты для АТД «Простой граф».

11) Объект задачи связан с объектом графа, для которого решается задача, с помощью указателя на объект «Простой граф».

12) Объект задачи для решения задачи использует интерфейс объекта «Простой граф» и интерфейс ассоциированных с графом объектов («Дескриптор вершины графа», «Дескриптор ребра графа», «Итератор вершин графа», «Итератор ребер графа», «Итератор исходящих ребер вершины»).

13) Объект задачи содержит все структуры, необходимые для решения задачи, сохранения результатов решения задачи.

14) Решение задачи происходит в момент создания объекта задачи, в момент вызова методов **Set (g)** (связывание объекта задачи с другим графом g), вызова метода **Restart ( )** (повторное решение задачи для графа).

15) Метод **Result ( )** формирует результат решения задачи для клиентской программы в удобной форме, в соответствии с целью решаемой задачи.

16) Разработать программу визуализации структуры графа (**V ≤ 20**). Программа должна обеспечивать визуальный просмотр структуры графа, результатов работы всех операций АТД «Простой граф» и ассоциированных объектов («Дескриптор вершины графа», «Дескриптор ребра графа», «Итератор вершин графа», «Итератор ребер

графа», «Итератор исходящих ребер вершины»), операций и результатов объектов задач. Должна обеспечиваться возможность модификации структуры графа (вставка или удаление вершин, ребер, изменение параметров, связанных с вершинами, ребрами).

Содержание отчета

1) Оглавление,
2) Общее задание,
3) Диаграмма взаимосвязи объектов, реализующих АТД «Простой граф», вспомогательных объектов, АТД задач 1 – 2.
4) Формат АТД «Простой граф»,
5) Клиентское определение класса «Простой граф»,
6) Формат АТД «Дескриптор вершины»,
7) Клиентское определение класса «Дескриптор вершины»,
8) Формат АТД «Дескриптор ребра»,
9) Клиентское определение класса «Дескриптор ребра»,
10) Формат АТД «Итератор вершин»,
11) Клиентское определение класса «Итератор вершин»,
12) Формат АТД «Итератор ребер»,
13) Клиентское определение класса «Итератор ребер»,
14) Формат АТД «Итератор исходящих ребер»,
15) Клиентское определение класса «Итератор исходящих ребер»,
16) Формат АТД «Итератор входящих ребер»,
17) Клиентское определение класса «Итератор входящих ребер»,
18) Формат АТД «Задача 1»,
19) Клиентское определение класса «Задача 1»,
20) Формат АТД «Задача 2»,
21) Краткое описание алгоритма, теоретическая оценка трудоемкости задачи 1,
22) Краткое описание алгоритма, теоретическая оценка трудоемкости задачи 2,
23) Заключение,
24) Список использованных источников,
25) Приложения:
    - Тексты реализации всех АТД.

Список рекомендуемой литературы

1. Альфред Ахо, Джон Э. Хопкрофт, Д. Ульман Структуры данных и алгоритмы. - М. - СПб – Киев: Вильямс, 2000 г. – 384 с.
2. Ананий Левитин Алгоритмы. Введение в разработку и анализ. – М. – СПб – Киев: Вильямс, 2006 г. – 576 с.
3. Фрэнк М. Каррано, Джанет Дж. Причард. Абстракция данных и решение задач на C++. Стены и зеркала. - М. - СПб – Киев: Вильямс, 2003 г. – 848 с.
4. Коллинз Уильям Дж. Структуры данных и стандартная библиотека шаблонов. – М.: ООО «Бином – Пресс», 2004 г. – 624 с.
5. Т. Кормен, Ч. Лейзерсон, Р. Ривест Алгоритмы. Анализ и построение. - М: БИНОМ, 2000 г. – 960 с.
6. Джерими Сик, Лай-Кван Ли, Эндрю Ламсдэйн. C++ Boost Graph Library. Библиотека программиста. – СПб: Питер. 2006 г. – 304 с.
7. Роберт Сэджвик. Фундаментальные алгоритмы на C++. Часть 5. Алгоритмы на графах - М: DiaSoft, 2002 г. – 496 с.