Библиотека graph.h

Описание реализованного типа данных Graph

Библиотека предоставляет средства для работы со взвешенными ориентированными графами с целыми неотрицательными весами ребер (если ребра не существует, его вес считается равным нулю) и с целыми неотрицательными ключами вершин. В типе данных Graph можно размещать матрицу смежности графа, массив из его вершин и их количество, в свою очередь каждая вершина реализована с помощью структурного типа Node - в нем размещается информация о вершине: ее ключ, порядковый номер и т.д.

Основные функции

- Функция init_graph

```
1 Graph init_graph(void);
```

осуществляет инициализацию графа без начального значения.
Входные данные отсутствуют.
Возвращает значение типа Graph.
Обработка ошибок: при ошибке выделения памяти печатается сообщение об ошибке в stderr и программа завершается с кодом 1.

- Функция initf_graph

```
1 Graph initf_graph(const char *name);
```

осуществляет инициализацию графа с начальным значением, считываемым из файла с именем name.
Входные данные: имя файла для чтения.
Возвращает значение типа Graph.
Обработка ошибок: если не удалось открыть файл, печатается сообщение об ошибке в stderr и программа завершается с кодом 1; аналогично функции init_graph обрабатывается ошибка при выделении памяти; также при неверном формате ввода графа печатается сообщение об ошибке и программа также завершается с кодом 1.
Спецификация: вводимый файл должен быть текстовым, его строки должны иметь следующий вид:
4 -> 2 {10}
5
2 -> 5
в первом случае считается, что создаются вершины графа с ключами 4 и 2, а также ребро из вершины с ключом 4 в вершину с ключом 2, вес ребра равен 10 (числу в фигурных скобках); во второй строчке создается только вершина с ключом 5 (изолированная); в третьей строчке не указан вес ребра, в этом случае он считается равным единице.
Также в строчках допускается любое количество пробелов и табуляций, при этом строки 4 2 -> 5 и 42 -> 5 считаются одинаковыми.

- Функция initstd_graph

```
1 Graph initstd_graph(void);
```

осуществляет инициализацию графа с начальным значением, считываемым с stdin. *Входных параметров* не имеет, *возвращает* значение типа Graph.
Обработка ошибок и *спецификация* полностью аналогичны функции initf_graph.

- Функция increase_weight

```
1 int increase_weight(Graph *graph, int key1, int key2, int n);
```

увеличивает вес ребра из вершины key1 в вершину key2 (здесь и далее key1 и key2 - ключи этих двух вершин) на n (или создает ребро веса n, если его не было).
Входные параметры: graph - указатель на переменную типа Graph, key1 и key2 - ключи вершин, n - число, определяющее увеличение веса ребра.
Выходные данные: код ошибки (0, если ошибок нет; 1 в противном случае).
Обработка ошибок: возвращает 1, если граф пуст или не инициализирован, а также если хотя бы одна из указанных вершин не существует или если n $<$ 0.

- Функция decrease_weight

```
1 int decrease_weight(Graph *graph, int key1, int key2, int n);
```

уменьшает вес ребра из вершины key1 в вершину key2 на n (или удаляет его, если n $>=$ текущего веса ребра).
Входные параметры: graph - указатель на переменную типа Graph, key1 и key2 - ключи вершин, n - число, определяющее увеличение веса ребра.
Выходные данные: код ошибки (0, если ошибок нет; 1 в противном случае).
Обработка ошибок: возвращает 1, если граф пуст или не инициализирован, а также если хотя бы одна из указанных вершин не существует или если n $<$ -1.
Спецификация: при вводе n = -1 вес ребра становится равным нулю, то есть оно удаляется.

- Функция add_node

```
1 int add_node(Graph *graph, int key);
```

создает в графе вершину с ключом key.
Входные парамерты graph - указатель на граф, имеет тип Graph *, key - ключ добавляемой вершины (целочисленный).
Выходные данные: код ошибки (0, если ошибок нет).
Обработка ошибок: в случае попытки добавить вершину с отрицательным ключом или в неинициализированный граф печатается сообщение об ошибке и программа завершается с кодом 1; аналогичные действия функция принимает при возникновении ошибки при выделении памяти.

- Функция delete_node

```
1 int delete_node(Graph *graph, int key);
```

удаляет вершину с ключом key из графа.
Входные параметры: graph - указатель на переменную типа Graph, key - ключ удаляемой вершины.
Выходные данные: код ошибки (0, если ошибок нет, иначе -1).
Обработка ошибок: если граф пустой или неинициализированный, или если вершины с таким ключом не существует, функция возвращает -1.

- Функция delete_graph

```
1 int delete_graph(Graph *graph);
```

удаляет граф, на который указывает указатель graph, то есть освобождает всю выделенную под этот граф память.
Входные параметры: graph - указатель на граф.
Выходные данные: код ошибки (0 в случае их отсутствия).

- Функция get_degree

```
1 int get_degree(Graph graph, int key);
```

возвращает степень вершины с ключом key.
Входные данные: graph - переменная с графом, key - ключ вершины.
Выходные данные: степень вершины с ключом key или код ошибки.
Обработка ошибок: функция возвращает -1, если граф пуст или если в нем не существует вершины с ключом key.

- Функция get_num_nodes

```
1 int get_num_nodes(Graph graph);
```

возвращает число вершин графа graph.
Входные данные: граф graph.
Выходные данные: число вершин графа.
Обработка ошибок: если граф не существует, число вершин считается равным нулю.

- Функция get_weight

```
1 int get_weight(Graph graph, int key1, int key2);
```

возвращает вес ребра из вершины key1 в вершину key2.
Входные данные: graph - переменная с графом, key1 и key2 - ключи двух вершин.
Выходные данные: вес ребра или код ошибки.
Обработка ошибок: функция возвращает значение -1, если одна из вершин не существует, значение -2, если граф пуст или не существует.

- Функция print_graph

```
1 int print_graph(Graph graph);
```

печатает граф, а точнее его матрицу смежности и информацию о вершинах, расположенный в переменной с именем graph.
Входные данные: graph - переменная с графом.
Выходные данные: код ошибки (0, если ошибок нет, иначе 1).
Обработка ошибок: возвращает 1 и печатает сообщение об ошибке при попытке вывести пустой или неинициализированный граф.

- Функция shortest_path

```
1 int shortest_path(Graph graph, int key1, int key2);
```

находит длину кратчайшего пути из вершины key1 в вершину key2.
Входные данные: graph - переменная с графом типа Graph, key1 и key2 - ключи вершин.

Выходные данные: длина пути или код ошибки.
Обработка ошибок: ошибка с кодом -1, если такого кратчайшего пути не существует, -2 если граф пуст или не существует хотя бы одной из вершин.

- Функция median

```
1 int median(Graph graph);
```

находит медиану графа, то есть вершину, имеющую минимальную сумму кратчайших расстояний до всех остальных вершин.
Входные параметры: graph - переменная с графом типа Graph.
Выходные данные: ключ найденной вершины или код ошибки.
Обработка ошибок: ошибка с кодом -1, если такой вершины не существует, -2 если граф пуст.