Итеративные и рекурсивные алгоритмы

Контрольные вопросы

1. Каковы особенности итеративного и рекурсивного алгоритма?

Рекурсия — это такой способ организации обработки данных, при котором программа вызывает сама себя непосредственно, либо с помощью других программ. Под рекурсией понимают способ задания функции через саму себя, например, способ задания факториала в виде N! = (n – 1)!*n.

Итерация — способ организации обработки данных, при котором определенные действия повторяются многократно, не приводя при этом к рекурсивным вызовам программ. Под итерацией понимают результат многократно повторяемой какой-либо операции, например, представление факториала в виде n! = 1*2*3... *n

2. В каких случаях целесообразно использовать рекурсивный или итеративный алгоритм? Приведите примеры итерации и рекурсии.

Эффективным средством программирования для некоторого класса задач является рекурсия. С ее помощью можно решать сложные задачи численного анализа, комбинаторики, алгоритмов трансляции, операций над списковыми структурами и т.д. Программы в этом случае имеют небольшие объемы по сравнению с итерацией и требуют меньше времени на отладку.

Рекурсивный алгоритм:

Когда задача естественно разбивается на более простые подзадачи того же типа. Например, алгоритмы обхода графов, факториал числа, вычисление чисел Фибоначчи.

Когда решение задачи требует обращения к ранее вычисленным результатам (мемоизация). Например, динамическое программирование.

Итеративный алгоритм:

Когда требуется более эффективное использование памяти. Рекурсия может привести к переполнению стека при больших входных данных.

Когда требуется большая скорость работы. Итеративные алгоритмы часто бывают более эффективными по времени исполнения.

Когда нет глубокой вложенности вызовов или необходимости сохранения стека вызовов. Например, итеративный обход списков.

3. Все ли языки программирования дают возможность рекурсивного вызова процедур?

Нет, не все языки программирования предоставляют равные возможности для реализации рекурсивных вызовов процедур. Некоторые языки могут иметь ограничения или не поддерживать рекурсию вообще. Например:

Некоторые ограниченные микроконтроллерные языки программирования не поддерживают выполнение рекурсивных функций из-за ограничений в памяти и стеке.

Некоторые функциональные языки программирования могут давать большие преимущества в работе с рекурсией, так как рекурсия является естественной для них.

Некоторые языки программирования, особенно те, которые ориентированы на процедурное программирование и вычислительную эффективность, могут иметь ограничения на глубину рекурсии во избежание переполнения стека.

Однако, в целом, большинство современных языков программирования, таких как Java, Python, C++, JavaScript, поддерживают рекурсию и позволяют реализовывать рекурсивные вызовы процедур. Важно помнить о возможных

ограничениях по глубине рекурсии в некоторых языках и конкретных средах выполнения.

4. Приведите пример рекурсивной структуры данных.

К рекурсивным процедурам относятся структуры строчные (стек, очередь, дек) и списковые. Список - набор элементов, расположенных в определенном порядке. Список очередности - список, в котором последний поступающий элемент добавляется к нижней части списка. Список с использованием указателей - список, в котором каждый элемент содержит указатель на следующий элемент списка.

Однонаправленный и двунаправленный списки - линейный список, в котором все исключения и добавления происходят в любом месте списка. Однонаправленный список отличается от двунаправленного списка только связью. Иными словами, в однонаправленном списке можно перемещаться только в одном направлении (из начала в конец), а двунаправленном - в любом. В однонаправленном списке структура добавления и удаления такая же, только связь между элементами односторонняя.

5. Что такое указатели и динамические переменные в языке Турбо Паскаль?

Рассмотрим понятия списка, указателя и динамической переменной языка Турбо Паскаль. Информационную часть можно описать как INTEGER (целый тип), REAL (действительный), CHAR (символьный) и т.д. Для отображения указателя (горизонтальной стрелки) в языке Турбо Паскаль введен особый тип данных, который называется указателем. Для его описания нет ключевого слова, вместо него используется символ ^. Структуру данных, рассмотренную в виде списка, можно представить на языке Турбо Паскаль следующим образом:

```
TYPE LINK = ^ELEMENT;
   ELEMENT =RECORD;
     INFORM: CHAR;
     NEXT: LINK;
END;
```

Здесь LINK - указатель, указывает на ELEMENT. Структура элемента ELEMENT отражена в виде записи, состоящей из двух частей: INFORM и NEXT. Следует обратить внимание на характер структуры данных. В начале описания тип данных LINK указывает на ELEMENT, у которого, в свою очередь, одна из составляющих NEXT является типом указателя LINK, а LINK указывает на ELEMENT. Получается замкнутый круг. Такая структура данных называется рекурсивной.

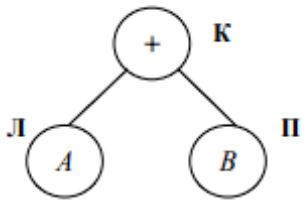
6. Укажите виды обхода бинарных деревьев.

Наиболее распространены три способа обхода узлов дерева, выше получили следующие названия:

1) обход в направлении слева направо (обратный порядок, инфиксная запись);

2) обход сверху вниз (прямой порядок, префиксная запись);

3) обход снизу вверх (концевой порядок, постфиксная запись).



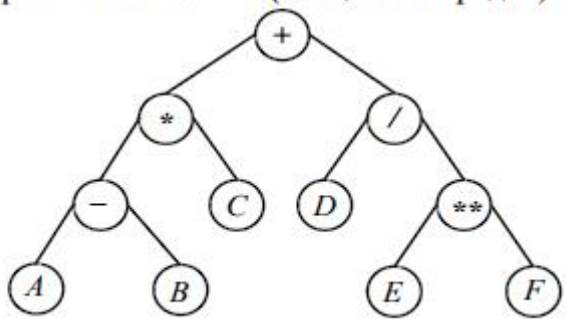
КЛП (Корень - Левая - Правая)
Прямой обход: $+AB$ (префиксная)

ЛКП (Левая - Корень - Правая)
Обратный обход: $A+B$ (инфиксная)

ЛПК (Левая - Правая - Корень)
Концевой обход: $AB+$ (постфиксная)

В результате обхода дерева (рис.6), каждым из трех способов порождаются следующие последовательности прохождения узлов:

- слева направо: $A + B*C - D$ (обратный порядок);
- сверху вниз: $* + AB - CD$ (прямой порядок);
- снизу вверх: $AB + CD -*$ (концевой порядок).



7. Приведите пример рекурсивной структуры данных.

К рекурсивным процедурам относятся структуры строчные (стек, очередь, дек) и списковые. Список - набор элементов, расположенных в определенном порядке. Список очередности - список, в котором последний поступающий элемент добавляется к нижней части списка. Список с использованием указателей - список, в котором каждый элемент содержит указатель на следующий элемент списка.

Однонаправленный и двунаправленный списки - линейный список, в котором все исключения и добавления происходят в любом месте списка. Однонаправленный список отличается от двунаправленного списка только связью. Иными словами, в однонаправленном списке можно перемещаться только в одном направлении (из начала в конец), а двунаправленном - в любом. В однонаправленном списке структура добавления и удаления такая же, только связь между элементами односторонняя.

8. Что такое указатели и динамические переменные в алгоритмических языках?

В алгоритмических языках программирования, таких как C и C++, указатели и динамические переменные играют важную роль при работе с памятью и

структурами данных. Давайте рассмотрим определения и особенности указателей и динамических переменных:

Указатели:

Указатель - это переменная, содержащая адрес ячейки памяти, где хранится значение другой переменной или объекта.

- Указатели позволяют манипулировать данными, ссылаясь на их памятьное расположение, а не на сами значения.

- Позволяют обращаться к динамически выделенной памяти и работать с динамическими структурами данных, такими как связанные списки, деревья, графы и т.д.

Динамические переменные:

- Динамическая переменная - это переменная, память для которой выделяется во время выполнения программы, в отличие от статических переменных, для которых память выделяется при компиляции.

- Часто динамические переменные хранятся в куче (heap) - области памяти с динамическим выделением и освобождением памяти.

- Позволяют программе эффективно управлять памятью и создавать сложные структуры данных во время выполнения.

Алгоритмы построения остовного (покрывающего) дерева сети

Контрольные вопросы

1. **Что понимается под остовным деревом?**

Остовное дерево графа G — ациклический связный подграф связного неориентированного графа, в который входят все его вершины.

2. **Каковы особенности методов Крускала и Прима?**

Основная идея алгоритма Крускала заключается в сортировке ребер на основе их веса. После этого мы начинаем отбирать ребра одно за другим, исходя из меньшего веса. В случае, если мы берем ребро, и это приводит к формированию цикла, то это ребро не включается. В противном случае ребро включается.

По сути, алгоритм Прима представляет собой модифицированную версию алгоритма Дейкстры. Сначала мы выбираем узел для начала и добавляем всех его соседей в очередь приоритетов.

После этого мы выполняем несколько шагов. На каждом шаге мы извлекаем узел, до которого нам удалось добраться, используя ребро с наименьшим весом. Следовательно, очередь приоритетов должна содержать узел и вес ребра, благодаря которому мы достигли этого узла. Кроме того, она должна сортировать узлы внутри него на основе переданного веса.

3. **В чем состоит методика анализа сложности алгоритмов построения остовного дерева графа?**

Методика анализа сложности алгоритмов построения остовного дерева графа направлена на оценку производительности различных алгоритмов и их эффективности в поиске минимального остовного дерева для графа. Вот основные шаги этой методики:

1. Определение входных данных: Изучение характеристик входного графа, таких как количество вершин (узлов), количество ребер, связность и веса ребер.

2. Определение целевой функции: Целью анализа является определение минимального остовного дерева графа - подграфа с минимальной суммой весов входящих в него ребер.

3. Выбор алгоритма построения остовного дерева: Рассмотрение различных алгоритмов, таких как алгоритм Прима, алгоритм Краскала, алгоритм Борувки и другие, с учетом их сложности и особенностей.

4. Оценка временной сложности: Анализ алгоритмов на возможность выполнения за разумное время в зависимости от размеров входных данных. Это включает оценку количества операций, которые алгоритм должен выполнить.

5. Оценка памяти и пространственной сложности: Изучение использования памяти алгоритмом для хранения данных и промежуточных результатов. Это важно для определения требований по памяти и оптимизации использования ресурсов.

6. Изучение зависимости от характеристик графа: Анализ поведения алгоритма при разных типах графов, таких как плотные или разреженные, со случайным или упорядоченным распределением весов ребер.

7. Сравнение алгоритмов: Сравнение различных алгоритмов построения остовного дерева графа с целью выбора наиболее эффективного в конкретной ситуации.

Используя эту методику, можно провести анализ и выбрать подходящий алгоритм построения остовного дерева графа в зависимости от поставленных

задач, требований к производительности и характеристик графов, с которыми необходимо работать.

4. **Определить, является ли связным заданный граф.**

Неориентированный граф называется связным, если для любой пары вершин xi,xj⍰X существует хотя бы один маршрут, их соединяющий

5. **Найти все вершины графа, к которым существует путь заданной длины от выделенной вершины графа.**

алгоритму Дейкстры?

6. **Найти все вершины графа, достижимые из заданной.**

7. **Подсчитать количество компонент связности заданного графа.**

Компонентой связности неориентированного графа называется подмножество вершин, достижимых из какой-то заданной вершины. Как следствие неориентированности, все вершины компоненты связности достижимы друг из друга.

8. **Найти диаметр графа, т.е. максимум расстояний между всевозможными парами его вершин.**

Диаметром δ связного графа G называется максимальное возможное расстояние между любыми двумя его вершинами.

алгоритму Дейкстры?

9. **Найти такую нумерацию вершин орграфа, при которой всякая дуга ведет от вершины с меньшим номером к вершине с большим номером.**