Элементарные алгоритмы

Алгоритмы сортировки

Гномья сортировка.

Идея алгоритма – расстановка цветочных горшков по росту в качестве украшения двора дома. (Видели в фильмах?). Считается, что это делают гномы.

Горшок сравнивается с соседним. Если они стоят правильно – переход к следующей паре. Если нет – меняем их местами и возвращаемся назад, сравнивая с предыдущим.

Алгоритмы сортировки

Гномья сортировка:

```
i=0
ПОКА (i<n)
    ЕСЛИ (i==0) ИЛИ (A[i-1]<=A[i]) ТО
        i=i+1
    ИНАЧЕ
        врем=A[i]
        A[i]=A[j]
        A[j]=врем
        i=i-1
```

Алгоритмы сортировки

Недостатками этих алгоритмов является большое количество перестановок, если исходный массив изначально отсортирован неправильно или близок к этому. Элемент из конца массива будет передвигаться в начало по одному шагу за раз.

Для борьбы с большим количеством перестановок был разработан алгоритм «расческа», в котором шаг сравнения больше единицы. Как при расчесывании волос сначала берется гребень с широким шагом, а потом с более мелким. Так и в этом алгоритме сначала сравниваются элементы на большем расстоянии с последующем уменьшением шага сравнения до единицы.

Важнейший параметр алгоритма – фактор уменьшения.

Оптимальным считается значение 1,247 полученное из золотого числа по формуле: 1 / (1-e^(-φ)),

где e - экспонента; φ - "золотое" число.

Алгоритмы сортировки

Сортировка расческой:

```
шаг=n
прзн_перест=ИСТИНА //признак перестановки
ПОКА (шаг>1)ИЛИ (прзн_перест)
    ЕСЛИ (шаг>1) ТО
        шаг=шаг/1,247 // с отбрасыванием дробной части
    прзн_перест=ЛОЖЬ
    i=0
    ПОКА (i+шаг<n)
        ЕСЛИ (A[i]>A[i+шаг] ТО
            врем=A[i]
            A[i]=A[i+шаг]
            A[i+шаг]=врем
            прзн_перест=ИСТИНА
        i=i+1
```

Алгоритмы сортировки

Быстрая сортировка, она же сортировка Хоара, quicksort, qSort – алгоритм, разработанный английским информатиком Чарльзом Хоаром во время его работы в МГУ в 1960 году.

Является одним из самых быстрых универсальных алгоритмов, хотя является улучшением сортировки пузырьком. Основан на стратегии «разделяй и властвуй».

Общий принцип:

В массиве выбирается опорное значение. Стратегия выбора в общем случае не важна, хотя может сократить время сортировки.

Массив переупорядочивается так, чтобы слева от опорного элементы были элементы массива со значением меньше или равным опорному, а справа – больше.

Для каждой части массива операция разделения повторяется снова.

Алгоритмы сортировки

Обобщенный алгоритм быстрой сортировки:

1. Выбирается опорный элемент.

2. Производится разделение массива:

2.1. Выбираются два индекса l и r в которые заносятся значения левой и правой границ массива.

2.2. Индекс l увеличивается пока l-ый элемент не будет больше или равен опорному.

2.3. Индекс r уменьшается пока r-ый элемент не будет меньше или равен опорному.

2.4. Если l равен r, то операция разделения закончена. Иначе возвращаемся к шагу 2.2.

3. Рекурсивно упорядочиваются полученные в результате разделения подмассивы. База рекурсии – пустой массив или массив из одного элемента.

Рекурсия

Рекурсией называется вызов функцией самой себя с некоторым изменением входных параметров.

Пример рекурсии – вычисление факториала:

6!=6*5!

5!=5*4!

.....

1!=1

Главным условием для использования рекурсии в функциях является наличие базы рекурсии – то есть значения, которое зависит только от входного параметра и позволяет выйти из рекурсии.

В примере база – это факториал 1.

Алгоритмы в духе «У попа была собака...» использовать нельзя!

Рекурсия

Еще одним ограничивающим фактором применения простой рекурсии является объем специальной области памяти, где хранится контекст и адреса возврата при вызове функций. При большой глубине рекурсии эта память может быть переполнена, что приведет к ошибке выполнения программы.

Иными словами 1000000000! теоретически вычислимая рекурсия, так как имеет базу, но глубина рекурсии может превысить допустимые ограничения, что приведет к невозможности получения результата.

Рекурсия

Тем не менее, рекурсия может быть бесконечной. Это так называемая **хвостовая рекурсия.** Она поддерживается в некоторых языках программирования. Для ее работы нужно чтобы рекурсивный вызов был всегда последним в теле рекурсивной функции, и чтобы компилятор или интерпретатор языка умели обнаруживать подобный вызов. В таком случае нет нужды запоминать адрес возврата и память не переполняется.

Хвостовая рекурсия любимый прием функциональных языков программирования (Haskell, Erlang и др.), в основе которых лежит понимание любой программы как математической функции.

Алгоритмы сортировки

Обезьянья сортировка.

Также известна как случайная сортировка.

Элементы массива переставляются случайным образом.

Если они оказались отсортированы, то алгоритм завершился. Если нет, то он начинается сначала.

Пример неэффективного алгоритма. Использовать не надо.

Проверка вводимых данных

Проверка вводимых данных

1. Операции с данными.

Выражение В=А+Б потенциально не содержит проблем, а выражение В=А/Б может привести к появлению ошибки.

А будет ли возможна ошибка если написать:

ЕСЛИ (Б!=0) ТО

В=А/Б

Г=Д/В – а теперь?

Кроме того, что А может быть 0 и второе выражение станет вычислить невозможно, ошибка может возникнуть и при А=1 и Б=3, так как для целого В значение 0,(3) является «машинным нулем».

Аналогичные ограничения есть у четных корней, прямых и обратных тригонометрических функций.

Проверка вводимых данных

1. Операции с данными.

Все операнды функций, имеющих ограничения на область допустимых значений (ОДЗ), должны в обязательном порядке проверяться на соответствие ОДЗ.

Проверка вводимых данных

2. Операции с массивами и памятью.

Память данных и команд в современных ЭВМ в большинстве случаем не разделена и слабо защищена от попадания в чужую область памяти.

Так, объявив массив на 5 элементов можно попытаться обратиться к 6, 10, -5 элементам попав в чужую область памяти и исказив как данные, так и программные коды.

Аналогичные ошибки бывают при копировании данных:

Запрос пользователю: Введите имя

Скопировать в строку введенное имя

При этом размер копируемой области памяти в большинстве случаев определяется длинной введенной строки, без учета размера строки получателя, что приведет к переполнению, если пользователь ввел очень большое имя.

Проверка вводимых данных

2. Операции с массивами и памятью.

При любых операциях с массивами и памятью необходимо контролировать выход индексных переменных за границы.

При копировании блоков памяти необходимо проверять как размер копируемого блока, так и размер области получателя выбирая минимальное из них или отказываясь от копирования при превышении допустимого объема.

Проверка вводимых данных

3. Знаковые и беззнаковые переменные.

Как уже упоминалось знаковые и беззнаковые переменные это разная интерпретация одних и тех же значений разрядной сетки числа.

Например:

```
ЕСЛИ n<10
   i=0
   ПОКА (i<n)
   .....
   i=i+1
```

Этот безобидный код ограничивает количество выполняемых циклов 10.

Но если мы введем n=-1 и при этом n будет беззнаковым или беззнаковым будет i, то при однобайтовом n станет 255, при двухбайтовом – 65535, при четырехбайтовом > 4 млрд.

Проверка вводимых данных

3. Знаковые и беззнаковые переменные.

Нельзя использовать в одном выражении знаковые и беззнаковые переменные, так как это может привести к непредсказуемой их интерпретации и ошибкам, как следствие того, что знаковое число воспринимается как большое беззнаковое и наоборот.

Перед вычислениями знаковые и беззнаковые числа должны быть принудительно приведены к одной форме.

Домашнее задание

Разработать и реализовать программу, которая будет сортировать массив алгоритмом быстрой сортировки.