Полное построение алгоритма ч 2.

Задача коммивояжера

Реализация алгоритма.

На этом этапе следует ответить на вопросы:

1. Каковы основные переменные?
2. Каких они типов?
3. Сколько нужно массивов, и какой размерности?
4. Имеет ли смысл пользоваться связными списками?
5. Какие нужны подпрограммы (возможно, уже записанные в памяти)
6. Каким языком программирования пользоваться.

Пункты 1-4 - построение структур данных.

Пункты 5-6 – непосредственное использование языка программирования.

Конкретная реализация может существенно влиять на требования к памяти и на скорость работы алгоритма.

Реализация алгоритма.

Другой аспект построения программной реализации - это программирование "сверху - вниз".

Необходимо разбить задачу на элементарные шаги (процедуры), т.е.

преобразовать алгоритм в такую последовательность все более конкретизированных алгоритмов, что окончательный вариант будет представлять собой программу

Реализация алгоритма.

1. Процедура генерации всех возможных перестановок.
2. Процедура вычисления стоимости каждого полученного пути.
3. Процедура сравнения различных путей и выбора минимального.

Реализация алгоритма.

На первом этапе пункт 1 может быть осуществлен вручную, с помощью ввода данных с клавиатуры.

Необходимо определить, что будет на входе и на выходе каждой процедуры.

1. Для генерации перестановок:
   - Вход: количество городов (К)
   - Выход: массив всех перестановок

(от 1 до К, матрица всех возможных путей).

Реализация алгоритма.

2. Процедура вычисления стоимости каждого полученного пути.

   Вход:

   Выход:

   Описать назначение и структуру данных

3. Процедура сравнения различных путей и выбора минимального

Алгоритм формирования перестановок «вручную»

Анализ алгоритма и его сложности

- В начале проводится оценка ресурсов:
- Как будет использовать алгоритм ресурсы машины, например, память (получение оценок или границ для объема памяти).
- Полезно оценить время работы до отладки и программирования.
- Необходимо иметь абсолютный (количественный) критерий для сравнения двух алгоритмов, претендующих на решение одной и той же задачи. Более сложный алгоритм должен быть улучшен или отброшен
- *Когда можно считать решение задачи оптимальным?* Когда алгоритм настолько хорош, что его невозможно значительно улучшить.

Анализ алгоритма и его сложности

Пусть A - алгоритм для решения некоторого класса задач.

N - размерность отдельной задачи из этого класса.

Может быть:

- просто скаляр, равный числу вершин графа;
- размер массива или длина вводимой
- последовательности.

Анализ алгоритма и его сложности

- Пусть $f_A(n)$ - рабочая функция, дающая верхнюю границу для максимального числа основных операций (сложение, сравнение), которые должны быть выполнены алгоритмом A для решения задачи размерности n.
- Критерий оценки качества алгоритма A основан на времени работы в худшем случае:
  1. Алгоритм A - *полиномиальный,* если $f_A(n)$ растет быстрее, чем полином от n.
  2. В противном случае алгоритм A называется *экспоненциальным* (exp)
- Последовательные или параллельные машины более или менее способны воспринимать полиномиальные алгоритмы для задач большой размерности, а на экспоненциальных задачах они довольно быстро "задыхаются".

Анализ алгоритма и его сложности

- **Введем обозначения:**
- Функцию f(n) обозначим как O[g(n)] и будем говорить, что она порядка g(n) для больших n, если

lim f(n)/g(n)=const≠0

- Функцию f(n) обозначим как o[z(n)] и будем говорить, что она порядка z(n) для больших n, если

lim f(n)/z(n)=0

- Если f(n)=O[g(n)], то эти две функции возрастают с одинаковой скоростью при n→∞, то есть эти два алгоритма одного класса, они одинаково растут.
- Если f(n)=o[z(n)], то z(n) растет горазда быстрее, чем f(n).

Анализ алгоритма и его сложности

Примеры:

- Полином $f(n)=2n^5+6n^4+6n^2+18$ есть $O(n^5)$
- Функция $f(n)=2^n$ есть $o(n!)$, так как $2^n/n! \to 0$
- $f(n)=O(2^{n+1})$
- $f(n)=o(5^{n+1})$
- $f(n)=1000\sqrt{n}$ $f(n)=O(n)$

Анализ алгоритма и его сложности

- **Итак,** алгоритм A полиномиальный, если $f_A(n)=O(P_k(n))$ или $f_A(n)=o(P_k(n))$, где $P_k(n)$- некоторый многочлен от переменной n произвольной фиксированной степени k.

- В противном случае алгоритм является экспоненциальным.

Анализ алгоритма и его сложности

"Задача коммивояжера"

- Размерность задачи - n.
- Оценка времени работы алгоритма O(n!), так как количество перестановок первых n-1 положительных целых чисел (n-1)!,
- т.е., эта часть алгоритма потребует O(n-1!) шагов. В каждой перестановке можно найти путь и его стоимость за O(n) шагов (т. к. n сумм.)
- $f_A(n)=O[n \cdot (n-1)!]=O(n!)$ - верхняя граница для общего времени работы.

Анализ алгоритма и его сложности

- Пусть размерность n=20
- время выполнения одной операции:

(сравнение, сложение, поиск элемента матрицы) - 10^{-7} сек.

- Тогда $20! \approx 2 \cdot 10^{18}$

(по формуле Стирлинга $n! = 2 \cdot 10^{n-2}$)

- $C \cdot 2 \cdot 10^{18} \cdot 10^{-7} = C \cdot 2 \cdot 10^{11}$ (70 веков),

где C - константа.

Замечание: Умные люди все это вычисляют на стадии разработки алгоритма, а не после того, как запрограммируют.

Проверка программы

Эксплуатации программы предшествует её отладка.

- **Отладка программы** - экспериментальное подтверждение того факта, что программа делает именно то, что должна.
- **Проверка вручную:** рассматривается задача небольшой размерности и все просчитывается на бумаге, если есть сравнение - пример на каждую ветвь (таблица истинности).
- **Тестирование каждого участка программы**, т.е., множество выводов всех возможных крайних случаев

если все случаи проверить практически невозможно, то проверить только те, которые встретятся с наибольшей вероятностью

Проверка программы

- **Особенности ОС**, которые могли не учесть. (Пример с фирмой MS).
- **Проверка качества алгоритма**.
  - Какие были сделаны допущения.
  - Учесть все возможные варианты.
  - Работает ли алгоритм лучше в среднем, чем в худшем случае. (→п.6)
- **Тестирование для определенных вычислительных ограничений.**
- **Анализ среднего функционирования.**

Проверка программы

- Многие программы для некоторых входных данных работают хорошо, а для других плохо.
- Характеристика работы алгоритма должна меняться плавно от хорошей к плохой при переходе от входных данных, на которых алгоритм работает хорошо, к входным данным на которых это не так.

"Задача коммивояжера"

- При $n \leq 6$ работает хорошо.
- При $6 \leq n \leq 15$ плохо.
- При $n \geq 15$ просто ужасно.

Проверка программы

- Из формулировки задачи вытекает необходимость проверки работы программы по крайней мере на двух тестах.
- Пусть, например, в задаче требуется подсчитать количество окружностей, каждая из которых проходит хотя бы через три различные точки из заданного множества, в котором не меньше трех точек.

Тогда в качестве тестов заведомо необходимо взять:

- множество точек, лежащих на одной прямой (с ожидаемым сообщением об отсутствии искомых окружностей),
- множество, в котором не все точки лежат на одной прямой

в этом случае тест должен содержать ответ -- сколько требуемых окружностей должна обнаружить программа с учетом приближенности вычислений, о которых говорилось ранее).

Проверка програмы

- Далее, всякий раз, когда в алгоритме, решающем задачу, происходит разветвление, набор тестов необходимо пополнить так, чтобы иметь возможность пройти каждую из ветвей.
- Аналогично, если встречается оператор цикла с условием продолжения, то в наборе должен появиться тест, на котором тело цикла не выполняется ни разу, а также тест, на котором тело цикла выполняется хотя бы один раз

Пример тестирования

- Пусть требуется построить программу, которая печатает сообщение

N--ПРОСТОЕ, если натуральное число N является простым, и сообщение

N--СОСТАВНОЕ в противном случае.

Составление документации:

- Описание алгоритма на языке, понятном для человека, не связанного с предметной областью
- Описание исходных и выходных данных
- Описание программы (алгоритма)
- Руководство по вводу либо корректировке данных
- Особенности функционирования программы (особые случаи, ограничения)
- Контрольный пример (примеры расчетов)

Описание алгоритма и данных

- Самое главное - оформлять в том виде, в котором хотелось бы читать.
- Следует учесть, что ваше описание должны понять люди, не владеющие предметной областью.
- Описать план алгоритма «сверху – вниз».
- Описать форматы данных и требования к вводу - выводу.

Описание алгоритма

- При составление больших программ (систем) возникает необходимость разбивать задачу на подзадачи, чтобы над каждой могла работать отдельная группа людей.
- Разные группы должны контактировать между собой, так как выход из одной задачи это вход в другую.
- Основная ошибка - что-то неправильно описали.

Особенности функционирования

- Указать условия функционирования и ограничения. Указать также, в каких случаях программа работает, а в каких не работает или работает плохо.
- Привести доказательство правильности функционирования алгоритма.
- Приложить описание тестовых примеров и результаты тестирования.
- Описать порядок настройки программы на конкретные условия функционирования.

Задание к практической работе: Решение задачи коммивояжера

1. Программирование исчерпывающего алгоритма для задачи коммивояжера.

2. Дополнить задачу коммивояжера (исчерпывающий алгоритм) процедурой генератора перестановок

3. Докажите, что если матрица стоимостей в задаче коммивояжера с ***n*** городами симметрична, то число разных по стоимости путей (гамильтоновых циклов) равно ***(n-1)!/2***

Генератор перестановок

Описание алгоритма