Алгоритмы на строках

Махортов Сергей Дмитриевич
ВГУ, ФКН, кафедра ПиИТ
http://www.cs.vsu.ru/msd

Алгоритмы на строках

- Обработка текстов
- Сжатие данных
- Криптография
- Распознавание речи
- Вычислительная геометрия
- Молекулярная биология

Алгоритмы на строках

Новые области:

- Компьютерное зрение
- Информационный поиск в глобальных сетях
- Вычислительная геномика

Особенность:

- *Эффективная* обработка *гигантских* символьных последовательностей

Поиск вхождений образца

- Текст T длины n
- Образец P длины m
- Найти позиции всех вхождений P в T

Наивный алгоритм

ATATGAACTGAGATCAAT

ATCA

Наивный алгоритм

ATATGAACTGAGATCAAT

ATCA

Наивный алгоритм

ATATGAACTGAGATCAAT

  ATCA

Наивный алгоритм

ATATGAACTGAGATCAAT

ATCA

Наивный алгоритм

ATATGAACTGAGATCAAT

ATCA

Примеры оценок

- Наивный алгоритм: $O(n \times m)$
- Кнута-Морриса-Пратта: $O(n + m)$
- Суффиксное дерево (П. Вайнер):

    построение дерева – $O(n)$

    вычисление вхождений – $O(m$ + число_вхождений)

Разница существенна при больших n и m. Например, при исследовании ДНК возможны значения

$n = 3000000$, $m=300$.

Данный курс

Основные разделы:

- основы теории сложности алгоритмов
- постановка и решение известных задач обработки строк
- обсуждение некоторых нерешенных задач
- обзор применений в конкретных предметных областях

Значение:

- знакомство с современным разделом компьютерных наук
- стимулирование новых исследований
- итого: устранение пробела в университетском образовании

Основные понятия: алгоритм

- *Алгоритм* – формальная процедура, получающая *исходные данные – вход*, и выдающая результат на *выход*
- *Псевокод* – средство записи алгоритма
- Например, подмножество языка программирования без использования технических деталей – описания типов, обработки исключений и т.п.

Основные понятия: строка

- *Алфавит* – множество различимых элементов (*символов*)
- *Строка S* – конечная последовательность символов алфавита
- *n* = |*S*| – число символов (длина) строки *S*
- *Пустая строка* ε – не содержит символов
- Интерпретация: массив символов *S*[0..*n*-1]
- Две *строки* на общем алфавите *равны*, если имеют равные длины и посимвольно совпадают

Основные понятия: подстрока

- *Подстрока* – это строка, полученная как непрерывная подпоследовательность исходной строки
- Подстрока $S[i..j]$ строки S начинается в позиции i и заканчивается в позиции j
- $S[0..i]$ называется *префиксом*, а $S[j..n\text{-}1]$ – *суффиксом* строки S (более кратко – $S[..i]$ или $S[j..]$)
- При $i > j$ строка $S[i..j]$ пуста
- Длина непустой строки $S[i..j]$ равна $j – i + 1$
- Подстрока называется *собственной*, если она не равна S.
- Пустая строка – собственная подстрока любой непустой строки в любой позиции

Наивный алгоритм поиска вхождений

```
Naive-String-Match (T, P)
{
  n = strlen(T); m = strlen(P); // Функция strlen(S) вычисляет |S|
  for (i = 0; i <= n – m; ++i)
  {
    j = 0;
    while (j < m && P[j] == T[i + j]) ++j;
    if (j == m) printf ("Найдено вхождение в позиции %d\n", i);
  }
}
```

Принципы оценки алгоритмов

- Выбор модели вычислений – *машина с произвольным доступом без параллельных вычислений*
- *Время работы* алгоритма – число элементарных шагов, которые он выполняет
- Может также анализироваться объем используемой памяти
- Результат сравнительного анализа алгоритмов существенно не зависит от выбора (адекватной) вычислительной модели
- *Вычислительная сложность* алгоритма – функция, выражающая зависимость времени работы от размера входа
- Способ измерения размера входа зависит от конкретной задачи. Часто размером считают число элементов данных (n).
- Иногда размер входа измеряется не одним числом, а несколькими (например, при поиске образца – n, m)

Оценки алгоритмов – упрощения

- Существенна не формула функции, а ее асимптотические свойства при возрастании аргумента
- Соответствующие упрощения: не учитываются величины меньших порядков и постоянные коэффициенты
- Пример: для времени работы получено выражение вида $an^2 + bn + c$, где n – размер входа; $a > 0$, b, c – константы, не зависящие от n
- Такая сложность пропорциональна n^2, то есть равна $\Theta(n^2)$
- Теоретически два алгоритма, имеющие одинаковые оценки сложности, равноценны
- Алгоритм с меньшим порядком роста времени работы как правило предпочтительнее

Оценки в худшем, лучшем и среднем случаях

- Для многих алгоритмов важен не только размер входа, но и значения его элементов
- В зависимости от них рассматривают оценки сложности в *худшем* и *лучшем* случае
- Пример – поиск первого вхождения образца: в худшем случае сложность наивного алгоритма квадратична, в лучшем – *линейна* по m
- Пример – поиск всех вхождений образца: в худшем случае сложность наивного алгоритма квадратична, в лучшем – *линейна* по n
- *Среднее* время работы алгоритмов – вероятностные методы
- *Ожидаемое время* работы алгоритма = его *математическое ожидание*

Оценка в худшем случае – приоритетна

- Если есть оценка в худшем случае, то для *любого* входа гарантировано выполнение алгоритма, причем с известной оценкой сверху
- На практике «плохие» входы встречаются часто. Например, при поиске многократно присутствующей подстроки в тексте
- Время работы в среднем нередко близко ко времени работы в худшем случае, то есть имеет ту же асимптотическую оценку

Асимптотические оценки

- При анализе алгоритмов оценивают асимптотику времени работы при стремлении размера входа к бесконечности
- В оценках отбрасывают члены меньших порядков и пренебрегают коэффициентом при оставшемся старшем слагаемом
- Если у некоторого алгоритма скорость роста меньше, то в большинстве случаев он эффективнее
- Исключение – достаточно короткие входы, для которых разница во времени работы ничтожна
- Рассматриваем одномерный случай – размер входа задается одной переменной; понятия легко переносятся и на большее число измерений

Θ - обозначение

- Показывает асимптотическую оценку вычислительной сложности алгоритмов
- Например, для времени $T(n)$ работы некоторого алгоритма на входах длины n, равенство $T(n) = \Theta(n^2)$ интерпретируется так:
$$\exists c_1, c_2 > 0, n_0 > 0 : 0 \leq c_1 n^2 \leq T(n) \leq c_2 n^2 \ (\forall n > n_0)$$
- Вообще для функций $f(n)$, $g(n)$ запись $f(n) = \Theta(g(n))$ означает:
$$\exists c_1, c_2 > 0, n_0 > 0 : 0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n) \ (\forall n > n_0)$$
- Говорят, что $g(n)$ асимптотически точно оценивает функцию $f(n)$
- В этом случае, очевидно, верно и обратное
- Важный частный случай: $\Theta(1)$ - ограниченная функция, отделенная от нуля положительной константой при достаточно больших значениях аргумента

O- и Ω- обозначения

- Запись $f(n) = \Theta(g(n))$ включает две оценки (верхнюю и нижнюю), которые можно сформулировать раздельно
- O – оценка сверху, Ω – оценка снизу
- Пишут $f(n) = O(g(n))$, если для $f(n)$ имеет место верхняя оценка:
$$\exists c > 0, n_0 > 0 : 0 \leq f(n) \leq cg(n) \ (\forall n > n_0)$$
- Равенство $f(n) = \Omega(g(n))$ означает, что для $f(n)$ есть нижняя оценка:
$$\exists c > 0, n_0 > 0 : 0 \leq cg(n) \leq f(n) \ (\forall n > n_0)$$
- Нетрудно показать, что $f(n) = \Theta(g(n)) \Leftrightarrow f(n) = O(g(n)), f(n) = \Omega(g(n))$
- Кроме того, $f(n) = O(g(n)) \Leftrightarrow g(n) = \Omega(f(n))$

o- и ω- обозначения

- Запись $f(n) = O(g(n))$ означает, что с ростом n частное $f(n)/g(n)$ ограничено. Если к тому же $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$, то $f(n) = o(g(n))$.
- Например, $2n = o(n^2)$, но $2n^2 \neq o(n^2)$
- Аналогично, если $\lim_{n\to\infty} \frac{g(n)}{f(n)} = 0$, то $f(n) = \omega(g(n))$
- Например, $n^2/2 = \omega(n)$, но $n^2/2 \neq \omega(n^2)$
- Очевидно, что $f(n) = o(g(n)) \Leftrightarrow g(n) = \omega(f(n))$

Свойства асимптотических функций

Выше были введены бинарные отношения. Их свойства:

- Транзитивность (*O, Ω, o, ω – аналогично*)

  $f(n) = \Theta(g(n)),\ g(n) = \Theta(h(n)) \Rightarrow f(n) = \Theta(h(n))$

- Рефлексивность

  $f(n) = \Theta(f(n)),\ \ f(n) = O(f(n)),\ f(n) = \Omega(f(n))$

- Симметричность

  $f(n) = \Theta(g(n)) \Leftrightarrow g(n) = \Theta(f(n))$

- Обращение

  $f(n) = O(g(n)) \Leftrightarrow g(n) = \Omega(f(n))$

  $f(n) = o(g(n)) \Leftrightarrow g(n) = \omega(f(n))$

Аналогия с числами

- Частичная аналогия с отношениями порядка на множестве чисел (например, целых):

  $f(n) = O(g(n)) \sim a \le b$

  $f(n) = \Omega(g(n)) \sim a \ge b$

  $f(n) = \Theta(g(n)) \sim a = b$

  $f(n) = o(g(n)) \sim a < b$

  $f(n) = \omega(g(n)) \sim a > b$

- Аналогия неполная. Например, в отличие от чисел, существуют несравнимые функции: $f(n) = n$ и $g(n) = n^{1+\sin(n)}$

Многомерный случай

- Асимптотические обозначения и их свойства легко переносятся на большее число измерений
- Запись $f(m, n) = \Theta(g(m, n))$ означает следующее:

$$\exists c_1, c_2 > 0, m_0, n_0 > 0 : 0 \leq c_1 g(m,n) \leq f(m,n) \leq c_2 g(m,n)$$
$$(\forall m > m_0, n > n_0)$$

- Аналогично – O, Ω, o, ω
- Вычислительная сложность наивного алгоритма поиска всех вхождений образца в текст выражается оценками $O(mn)$, $\Omega(n)$.