Дисциплина

«РАЗРАБОТКА ЭФФЕКТИВНЫХ АЛГОРИТМОВ»

Лекция 8

МЕТОД МАТЕМАТИЧЕСКИХ ОЖИДАНИЙ И АЛГОРИТМЫ ПОИСКА МАКСИМУМА И СОРТИРОВКИ ВСТАВКАМИ

Лектор: Михаил Васильевич Ульянов,

muljanov@mail.ru, **8 916 589 94 04**

Введение в теорию вероятностей

1. Почему математический маятник не есть предмет теории вероятностей? Ответ — у нас есть математическая модель, обладающая необходимой предсказательной силой. При отсутствии модели или точных начальных условий приходится рассматривать процесс как «черный ящик». Мы в силах только наблюдать за реализациями и их частотами — ТВ.
2. Изложение основ ТВ на основе теории меры принадлежит А.Н. Колмогорову — Kolmogorov A. N. Grundbegriffe der Wahrscheinlichkeitrechnung, in Ergebnisse der Mathematik. — Berlin, 1933.
3. Вероятностная модель есть совокупность вероятностного пространства и вероятностной меры (для континуального случая необходима сигма-алгебра на вероятностном пространстве) для дискретного случая:

$$M = \langle \Omega, P(.) \rangle$$

4. Ω — вероятностное пространство (полная группа событий). Нормировка вероятностной меры $P(\Omega) = 1$, и вероятностная мера аддитивна на пересекающихся подмножествах Ω.
5. Случайная величина — измеримая на множестве действительных чисел характеристика случайного события, т.е. $X(\omega) = r$, при этом X есть функция из вероятностного пространства Ω в множество действительных чисел R. Т.е. каждому элементарному случайному событию поставлено в соответствие число — вероятность.
6. Комогоров четко отделил области исследования теории вероятностей и математической статистики. Вероятностная модель не нуждается в доказательствах — это прерогатива исследователя, т.е. вероятности назначаются! Матстат на основе реальных экспериментов либо подтверждает, либо опровергает вероятностную модель!

Метод математических ожиданий для трудоемкости в среднем

Метод основан на красивой теореме теории вероятностей:

Если случайная величина $Y = \sum X_i$, и случайные величины X_i имеют математическое ожидание, то $E(Y) = E(\sum X_i) = \sum E(X_i)$. Высоким стилем — операторы суммы и математического ожидания коммутируют!

Рассматривая трудоемкость как случайную величину на множестве входов фиксированной длины, мы можем говорить, что трудоемкость в среднем есть математическое ожидание:

$$E(f(n)) = \sum p(D) f(D)$$

Но, достаточно часто таким способом не удаётся получить трудоемкость в среднем — это связано с трудностями определения вероятностей классов. Метод предполагает представление случайной величины в виде суммы случайных величин, для которых мат ожидание вычисляется достаточно просто.

Анализ алгоритма поиска максимума

Запись алгоритма имеет вид:

```
Max_A0 (S, n; Max, iMax)
  Max ← S[1]                    2
  iMax←1                        1
  For i ← 2 to n                1+3(n-1)
    If Max < S[i]               2(n-1)
      then
        Max ← S[i]              2m
        iMax←i                  1m
  end For
Return (Max, iMax)
End.
```

Входом данного алгоритма является массив из n элементов — соответствующая задача имеет тип L_n, а трудоемкость алгоритма зависит как от количества чисел в массиве, так и от порядка их расположения — алгоритм является количественно-параметрическим (класс NPR, подкласс $NPRS$).

Полный анализ для фиксированной размерности множества исходных данных предполагает получение функций трудоемкости для худшего, лучшего и среднего случаев. Для данного алгоритма фрагментом, определяющим параметрическую зависимость трудоемкости, является блок **`then`**, содержащий строки:

`Max` ← `S[i]` и `iMax`←`i`.

Обозначим общее количество выполнений этого блока, задаваемое алгоритмом **`Max_A0`**, через m, заметим, что сам блок содержит три операции.

Худший случай. Значение m будет максимально, а именно $m = n - 1$, когда на каждом проходе цикла выполняется переприсваивание текущего максимума — это ситуация, когда элементы исходного массива различны и отсортированы по возрастанию. В этом случае трудоемкость алгоритма равна:

$$f_A\hat{}(n) = 2 + 1 + 1 + (n-1)(3+2) + (n-1)(3) = 8n - 4 = \Theta(n).$$

Лучший случай. Значение m будет минимально, и равно нулю ($m = 0$), в том случае, если первый элемент массива является максимальным. Трудоемкость алгоритма в этом случае равна:

$$f_A^{\vee}(n) = 2 + 1 + 1 + (n-1)(3+2) = 5n - 1 = \Theta(n)$$

Трудоемкость в среднем. Для анализа трудоемкости этого алгоритма в среднем случае необходимо обосновать значение вероятности, с которой происходит переприсваивание максимума в строке `Max ← S[i]`. Это обоснование может опираться на следующие рассуждения, основанные на равновероятном распределении положения текущего максимума. Алгоритм поиска максимума последовательно перебирает элементы массива, сравнивая текущий элемент массива с текущим значением максимума. На очередном шаге, когда просматривается k -ый элемент массива, переприсваивание максимума произойдет, если в подмассиве из первых k элементов максимальным элементом

является последний. В случае если элементы исходного массива подчиняются равномерному распределению, вероятность того, что максимальный из k элементов расположен в определенной, например, последней позиции, равна $1/k$. Обозначим среднее значение m через $\overline{m}$. Тогда в массиве, содержащем n элементов, значение $\overline{m}$, а именно оно нас и интересует, определяется формулой:

$$\overline{m} = \sum_{k=2}^{n} \frac{1}{k} = H_n - 1 \approx ln\, n + \gamma - 1,\ \gamma \approx 0{,}57,$$

где H_n — n -ое гармоническое число, а γ — постоянная Эйлера. Суммирование начинается с двойки, поскольку первое присваивание максимума уже сделано до входа в цикл. Асимптотическое поведение H_n при $n \to \infty$ имеет вид.

$$H_n = ln\, n + \gamma - \frac{1}{2n} - \frac{1}{12n^2} + O(n^{-3}).$$

Таким образом, если исходный массив состоит из различных чисел, а генеральная совокупность представляет собой $n!$ различных перестановок чисел массива, при условии, что появление каждой такой перестановки на входе

алгоритма равновероятно, то значение $\overline{m}$ — математического ожидания количества операций присваивания максимума в основном цикле алгоритма для массива из n элементов, равно $H_n - 1$, а включая первое присваивание — H_n, тогда:

$$\overline{f_A}(n) = 1 + (n-1)(3+2) + 3(ln(n) + \gamma) = 5n + 3\,ln(n) - 4 + 3\gamma = \Theta(n).$$

Заметим, что для данного алгоритма значение средней трудоемкости значительно ближе к лучшему случаю, т. е. смещено влево относительно середины границ колебаний трудоемкости при фиксированной размерности задачи. Иными словами распределение значений трудоемкости как дискретной ограниченной случайной величины является сильно лево ассиметричным.

Анализ алгоритма сортировки вставками

Это достаточно хорошо известный алгоритм, использующий метод последовательного включения нового элемента в уже отсортированную часть массива. В литературе по алгоритмам его часто называют также алгоритмом сортировки вставками.

Изначально рассматривается массив, состоящий из одного элемента, к которому последовательно добавляются новые элементы из еще не отсортированной части исходного массива. При этом одновременно с поиском места для вставки нового элемента уже отсортированная часть массива справа сдвигается на одну позицию, освобождая место для вставки этого элемента. Таким образом, алгоритм работает «по месту», т. е. сортирует массив, не требуя дополнительного массива для размещения результата. Запись этого алгоритма в принятом алгоритмическом базисе имеет следующий вид:

```
Sort_Ins_A1 (A, n)
  For i ← 2 to n                          1+3(n-1)
    key ← A[i]                            2(n-1)
    j ← i-1                               2(n-1)
    While (j>0) and (A[j]>key)            4(k+1)
      A[j+1] ← A[j]                       4k
      j ← j - 1                           2k
    end While
    A[j+1] ← key                          3(n-1)
  end For
Return (A)
End.
```

Входом данного алгоритма является массив из n элементов — соответствующая задача имеет тип L_n, а трудоемкость алгоритма зависит как от количества чисел в массиве, так и от порядка их расположения — алгоритм является количественно-параметрическим (класс NPR, подкласс $NPRH$).

Значение k в стоках, связанных с циклом **`While`**, обозначает неизвестное число проходов данного цикла. Нельзя заранее сказать, сколько проходов выполнит цикл **`While`** для некоторого элемента массива, но, тем не менее, мы можем получить результат, рассматривая обработку всего массива в целом, т. е. используя амортизационный подход. Для анализа трудоемкости в среднем определим среднее общее количество сравнений, используя метод вероятностного анализа. Для этого заметим, что при вставке очередного элемента на i -ом шаге, когда $i - 1$ элементов уже отсортированы, существует i позиций для вставки нового элемента. В предположении о равновероятном попадании нового элемента в любую позицию алгоритм выполняет в среднем $i/2$ сравнений и перемещений для его обработки, что в сумме по всему внешнему циклу дает амортизационную оценку суммарного числа проходов цикла **`While`** при сортировке всего массива

$$\sum k = \sum_{i=1}^{n-1} \frac{i}{2} = \frac{n^2}{4} - \frac{n}{4}.$$

Используя трудоёмкости, указанные в строках записи алгоритма и методику анализа алгоритмических конструкций с учетом того, что на последнем проходе цикла **`While`** выполняется только сравнение, окончательно получаем функцию трудоемкости этого алгоритма в среднем

$$\overline{f}_{A1}(n) = 1 + (n-1)\cdot(3+2+2+4+3) + (4+4+2)\cdot\left(\frac{n^2}{4} - \frac{n}{4}\right) =$$

$$= 2{,}5n^2 + 11{,}5\cdot n - 13.$$

Таким образом, данный алгоритм относится к группе квадратичных по трудоемкости алгоритмов сортировки. Отметим, что поскольку значение коэффициента при главном порядке мало, то можно прогнозировать рациональность этого алгоритма при малых длинах входа.

Приведенные в таблице 1 экспериментальные данные, полученные усреднением по 10000 массивам для каждой размерности, подтверждают полученный теоретический результат (**Fth** — теоретическое значение по формуле, **Fe** — экспериментальные результаты, **d%** — относительная ошибка).

Таблица 1

n	Fth	Fe	d%
4	73,00	73,09	0,1233%
8	239,00	239,20	0,0837%
16	811,00	810,92	-0,0099%
32	2 915,00	2 917,63	0,0902%
64	10 963,00	10 962,09	-0,0083%
128	42 419,00	42 419,06	0,0001%
256	166 771,00	166 782,82	0,0071%
512	661 235,00	661 249,14	0,0021%

Для данного алгоритма ресурсная сложность определяется главным порядком функции трудоемкости и имеет вид

$$\overline{\Re}_c(A) = \langle \Theta(n^2),\ \Theta(n) \rangle,$$

где второй компонент отражает суммарные затраты памяти, определяемые объемом хранения исходного массива. Дополнительные затраты памяти алгоритма составляют $\Theta(1)$.