Точные алгоритмы и открытые проблемы

Лифшиц Ю.[*]

28 сентября 2005

Содержание

[*]Законспектировал лекцию Рыбак О.

1 Введение

Закон Хоара (о больших задачах):
"Внутри каждой большой задачи
сидит маленькая, которая
старается выбраться наружу"

Это не значит, что большие задачи на самом деле являются маленькими задачами, а значит, что внути себя они содержат маленькие задачи, которые надо сначала обнаружить а потом решить.

1.1 Рассматриваемые задачи

Данная лекция посвящена точным алгоритмам, т.е. алгоритмам для трудных задач, где мы заинтересованы с нахождении не приблизительного, а точного решения. Большинство задач сформулировано в 2005 году и из решение пока никому не известно. Рассматриваются как сложные задачи, так и простые, которые они содержат. Для маленьких задач существуют полиномиальные алгоритмы.

Задачи:

- Сумма размеров
  - Маленькая задача: табличная сумма
- Поиск максимального разреза
  - Маленькая задача: поиск треугольника в графе

2 От табличных сумм к сумме размеров

2.1 Постановка задачи

Задача 1. Даны натуральные числа $w_1, w_2, \dots, w_n$ и натуральное s. Требуется выяснить, существует ли подмножество $w_1, w_2, \dots, w_n$, сумма элементов которого дает ровно s.

Пример 1. Дано: 17 43 23 38 14 20 36 47
Есть ли подмножество с суммой 100?

2.2 Маленькая задача

Задача 2. Имеется таблица $2 \times n$ и заданное число s. Требуется выяснить, есть ли 2 числа из разных строчек, дающих в сумме s.

Алгоритм 1. Полный перебор. Время работы – $O(n^2)$.

Алгоритм 2. Перебор с сортировкой. Отсортировать первую строку, для каждого элемента из второй строки вычесть его из s и искать эту разность в первой строке. Время работы – $O(n \log n)$.

Задача 3. Имеется таблица $k \times n$ и заданное число s. Требуется выяснить, есть ли k чисел (по одному из каждой строки), дающих в сумме s.

Алгоритм 3. Полный перебор. Время работы – $O(n^k)$.

Алгоритм 4. Сведение к табличной 2-сумме. Разделить таблицу пополам. Для каждой половины перебрать все возможные варианты. Получится 2 длинные строки. К ним применить алгоритм для 2-суммы. Время работы – $O(n^{\lceil k/2 \rceil} \log n)$.

Замечание 1. Улучшение для $n = 3$. Для одной строки найти все разности с s, для двух других – перебрать все варианты. Время работы – $O(n^2)$.

Открытый вопрос 1. Существует ли алгоритм для 3-суммы, работающий быстрее, чем за $O(n^2)$?

Замечание 2. Для 4-суммы существует алгоритм Шреппеля-Шамира. Время работы – $O(n^2 \log n)$, требование памяти – $O(n)$.

Упражнение 1. Воспроизвести алгоритм Шреппеля-Шамира для табличной 4-суммы.

Оказывается, что задача табличной k-суммы, в особенности табличной 3-суммы, очень сильно связана с задачами вычислительной геометрии. Рассмотрим задачу "Три точки на прямой":

Задача 4. Даны координаты n точек на плоскости. Требуется определить, лежат ли какие-нибудь три из них на одной прямой.

Оказывается, что $O(n^2)$ для задачи "три-точки-на-прямой" – это такая же трудная граница, как и в табличных 3-суммах. Утверждается, что табличную 3-сумму с одинаковыми строками можно свести к "трем-точкам-на-прямой". Или, более формально:

Утверждение 1. *Если существует алгоритм для решения "трех-точек-на-прямой" трудоемкости $f(n)$, то есть и алгоритм для табличной 3-суммы трудоемкости $c \cdot f(n)$.*

Таки образом, если бы появился более эффективный алгорим решения задачи "три-точки-на-прямой", то и задача о табличной 3-сумме была бы решена более эффективно. Верно и обратное, а именно: до тех пор, пока не будет найден более эффективный алгоритм решения задачи о табличной 3-сумме, задача "три-точки-на-прямой" не будет решена более эффективно. Таким образом, задача про табличные 3-суммы играет важную роль в нахождении более эффективных алгоритмов решения других задач.

Приведем формальное сведение задачи о табличной 3-сумме к задаче "три-точки-на-прямой".

Задача 5. Дана последовательность $x_1, x_2, \ldots, x_n$ и сумма s.(Так как рассматриваются одинаковые строки, то они заменяются одной последовательностью). Определить, существуют x_i, x_j, x_k, такие что $x_i + x_j + x_k = s$.

Ответ 1. Отметим на графике $f(x) = x^3 - s \cdot x^2$ точки $(x_1, f(x_1)), (x_2, f(x_2)), \ldots, (x_n, f(x_n))$. Если три из них (с номерами i, j и k) лежает на некоторой прямой $y = ax + b$, то x_i, x_j и x_k являются корнями уравнения $x^3 - sx^2 - ax - b = 0$, то есть, по теореме Виета, $x_i + x_j + x_k = s$.

Факт 1. *Верно и обратное: если $x_i + x_j + x_k = s$, то $(x_i, f(x_i)), (x_j, f(x_j)), (x_k, f(x_k))$ лежат на прямой*

Упражнение 2. Проверить вышеизложенное решение для совпадающих точек.

Упражнение 3. Применить алгоритм для одинаковых строк к задаче, в которой строки неоднинаковы.

2.3 Большая задача

Теперь опишем применение задачи о табличной k-сумме для решения задачи о сумме размеров.

Задача 6. Имеется n натурадбных чисел и натуральное число s. Требуется определить, существует ли подмножество данных чисел, имеющее сумму s.

Алгоритм 5. Разделить множество на две части, сосчитать все подмножества в частях. После этого получится 2 строки размером $2^{n/2}$. Для них применить алгоритм для табличной 2-суммы. Данный алгоритм имеет недостаток – требование по памяти $O(2^{n/2})$.

Пример 2. Рассмотрим данный подход на приведенном выше примере. Пусть даны числа 17 43 23 38 14 20 36 47. Пусть $s = 100$. Разделим их на два подмножества:

$$\underbrace{\{17,\ 43,\ 23,\ 38\}}\underbrace{\{14,\ 20,\ 36,\ 47\}}$$

Для каждого подмножества составим всевозможные суммы. Получим 2 множества мощности $2^{n/2}$.

$$\{17,\ 43,\ 23,\ 38,\ 60,\ 40,\ 55\ldots\} \text{ и } \{14,\ 20,\ 36,\ 47,\ 34,\ 50,\ 61\ldots\}$$

Для второго множества составим разности его элементов с s и будем искать эти разности в первом множестве.

Можно применить алгоритм Шреппеля-Шамира для табличной 4-суммы:

Алгоритм 6. Разделить множество на 4 части и применить алгоритм Шреппеля-Шамира. Время работы – $O(2^{n/2})$, требование памяти $O(2^{n/4})$.

Замечание 3. Так как числа, дающие в сумме s, могут все лежать в одном подмножестве, то необходимо включить в разности вариант $s - 0$, то есть s.

2.4 Открытые задачи

Открытый вопрос 2. Построить алгоритм для табличной 6-суммы, работающий за $O(m^3 \log m)$ и требующий $O(m)$ памяти.

Открытый вопрос 3. Построить алгоритм для табличной k-суммы, работающий за $O(m^{\lceil k/2 \rceil - \varepsilon})$.

Открытый вопрос 4. Построить алгоритм для суммы размеров, работающий за $O(1,4^n)$.

Открытый вопрос 5. Построить алгоритм для суммы размеров, работающий за $O(1,99^n)$, но *полиномиальный* по памяти.

3 Задача о клике и Алгоритм Вильямса поиска максимального разреза

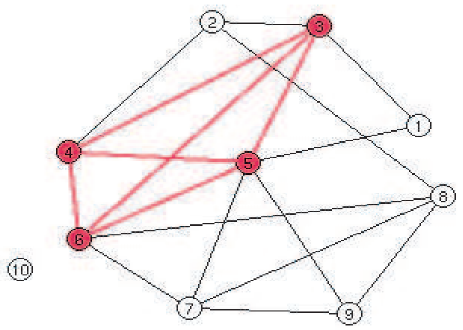
3.1 Постановка задачи

Задача 7. Дан граф G со взвешенными ребрами, имеющий n вершин. Требуется найти такое разбиение множества вершин графа G на две части, что сумма весов ребер между ними максимальна.

Факт 2. *Задача о клике и задача о максимальном разрезе являются $\mathcal{NP}$-полными*

3.2 Маленькая задача

Определение 1. Пусть дан граф G. *k-кликой* называется полный подграф из k-вершин.



Пример 3. На рисунке изображен граф из 10 вершин, содержащий полный подграф из 4 вершин, то есть 4-клику.

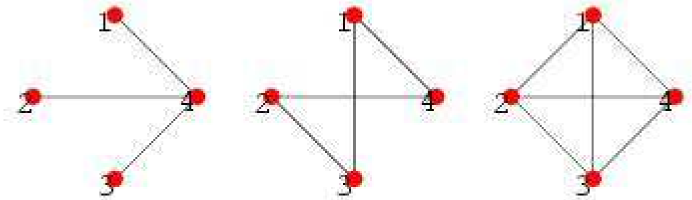
Маленькой задачей в задаче поиска максимального разреза является задача поиска 3-клики или треугольника в графе:

Задача 8. Пусть дан граф. Требуется найти 3-клику, то есть полный подграф из трех вершин.

Алгоритм 7. Простой способ – перебрать все возможные треугольники. Время работы – $O(n^3)$

Для описание более быстрого алгоритма понадобится вспомнить такое понятие, как матрица смежности.

Определение 2. Пусть дан граф $G = \langle V, E \rangle$, где V-множество вершин, E-множество ребер. Пусть $|V| = n$. Матрицей смежности графа G называется марица $M_{n \times n}$, $M = \{m_{i,j}\}$, где $m_{i,j} \neq 0$ тогда и только тогда, когда существует ребро $e \in E$ из вершины i в вершину j, то есть $e = (i, j)$.



$$\begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix} \quad \begin{pmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{pmatrix} \quad \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

Пример 4. На рисунке приведены примеры матриц смежности.

Пусть дана матрица $A_{n \times n}$. Возведем ее в квадрат. Пусть $B = A^2, B = \{b_{i,j}\}$. Тогда $b_{i,j} = \sum a_{i,k} \cdot a_{k,j}$. При этом смысл $b_{i,j}$ – количество путей длины 2 из i-той вершины в j-тую. Основываясь на данном факте, получаем следующий алгоритм:

Алгоритм 8. Возвести матрицу смежности в куб. Пусть $C = A^3$, $C = \{c_{i,j}\}$. Тогда элементы $c_{i,i}$, то есть стоящие на главной диагонали, будут показывать количество путей длины 3 из i-той вершины в i-тую, иначе говоря, количество треугольников. При использовании быстрого умножения матриц время работы алгоритма – $O(n^{2.376})$.

3.3 Открытые задачи для k-клик

Факт 3. *Приведем лучшие алгоритмы для поиска клик:*

3-*клика*	4-*клика*	5-*клика*	6-*клика*	7-*клика*
$O(n^{2.376})$	$O(n^{3.334})$	$O(n^{4.220})$	$O(n^{4.752})$	$O(n^{5.714})$

Время работы для клики произвольного размера – $O(1.2108^n)$.

Открытый вопрос 6. Возможно ли улучшить эти алгоритмы?

Открытый вопрос 7. Возможно ли найти более быстрые алгоритмы для других частных случаев клик, например существует ли алгоритм для 10-клики, время работы которого – $O(n^{7.5})$?

Открытый вопрос 8. Верно ли, что 3-клика так же трудна, как и умножение матриц? Иными словами, существует ли более быстрый алгоритм поиска 3-клики, который не использует умножение матриц?

3.4 Большая задача

Определение 3. Пусть дан граф G. *Разрезом* графа G называется разбиение множества его вершин V на два множества V_0 и V_1, причем $V_0 \cap V_1 = \emptyset$.

Определение 4. Пусть дан граф G. *Максимальным разрезом (MAX-CUT)* графа G называется такой разрез, в котором сумма весов ребер между двумя множествами V_0 и V_1 максимальна.

Перейдем к задаче поиска максимального разреза. В 2004 году Вильямс придумал новый алгоритм поиска максимального разреза. Изложим его основные идеи:

- Разделить вершины исходного взвешенного графа G на 3 равные части: A, B и C.
- В каждой части сделать полный перебор, то есть рассмотреть все возможные разрезы.
- Построить вспомогательный трехдольный граф H размера $2^{n/3} \times 2^{n/3} \times 2^{n/3}$. Вершины в этом графе будут соответствовать разрезам в частях A, B и C.
- Поиск максимального разреза сводится к поиску треугольника максимального веса.

Факт 4. *Каждый треугольник во вспомогательном графе H соответствует разрезу в исходном графе G.*

Требуется определить, какие из вершин отнести к одной части разреза, а какие – к другой.

Факт 5. *Рассмотрим граф H. Для каждой его доли i-тая вершина этой доли соответствует i-тому разрезу соответструющего подмножества вершин (A, B или C) в исходном графе G.*

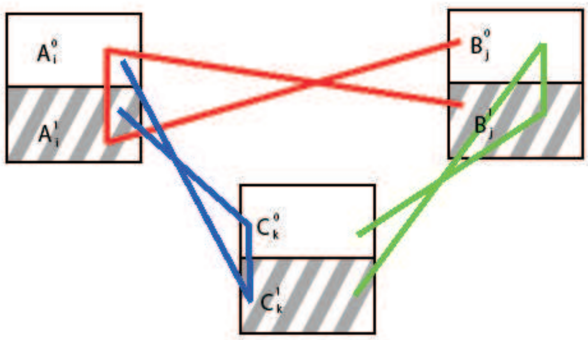
Обозначим i-тые разрезы множеств A, B и C соответственно

$$A = A_i^0 \cup A_i^1, B = B_i^0 \cup B_i^1, C = C_i^0 \cup C_i^1.$$

Так как исходный граф G – взвешенный, то требуется, чтобы вес треугольника (i, j, k) в графе H равнялся "весу" разреза

$$A_i^0 \cup B_j^0 \cup C_k^0 - A_i^1 \cup B_j^1 \cup C_j^1$$

Пусть $W(M, N)$ – сумма весов ребер из M в N, тогда присвоим:
ребру (i, j) – вес $W(A_i^0, A_i^1) + W(A_i^1, B_j^0) + W(A_i^0, C_k^1)$
ребру (j, k) – вес $W(B_j^0, B_j^1) + W(B_j^1, C_k^0) + W(B_j^0, B_j^1)$
ребру (k, i) – вес $W(C_k^0, C_k^1) + W(C_k^0, A_i^1) + W(C_k^1, A_i^0)$



Пример 5. Рисунок иллюстрирует формарование веса у ребер (i, j), (j, k), и (k, i). Составляющие веса обозначены красным, зеленым и синим цветами соответственно.

Алгоритм 9. (Алгоритм Вильямса)

- Построить вспомогательный граф.
- Посчитать веса всех вершин.
- Проверять, существует ли разрез весом больше z.
- Делать перебор по всем тройкам $z_{ab} + z_{bc} + z_{ca} = z$.
- Оставлять между первой и второй долей H только ребра весом хотя бы z_{ab}, аналогично по другим направлениям.
- Из графа H удалить все, что меньше и искать треугольники. Если хотя бы 1 треугольник есть, что разрез существует.

Трудоемкость – $O(2^{n/3})^{2.376} = 2^{0.792 \cdot n} < 1.732^n$

3.5 Открытые задачи

Открытый вопрос 9. Придумать алгоритм для $MAX\text{-}CUT$ быстрее $O(1,732^n)$.

Открытый вопрос 10. Придумать $O(1,99^n)$ алгоритм для $MAX\text{-}CUT$, но полиномиальный по памяти.

4 Заключение

Если из вышеизложенного ничего не запомнилось, то:

- Прогресс по $\mathcal{NP}$-трудным задачам зависит от эффективных алгоритмов решения для очень простых задач.
- Одинаковый барьер эффективности есть одновременно в табличных суммах и вычислительной геометрии.
- Быстрое умножение матриц помогает для оптимизации на графах.