SPb HSE, 1 курс, зима 2024/25
Практика по алгоритмам #17

Вероятная сложность

4 февраля 2025

Собрано 8 февраля 2025 г. в 11:26

Содержание

Вероятная сложность

1. **Понижение вероятности**

   (a) Алгоритм работает за $\mathcal{O}(n^2 \log n)$, вероятность успеха $1/\log n$.
   За какое время можно добиться вероятности успеха $1 - 1/n$?

   (b) Алгоритм работает за $\mathcal{O}(2^n)$, вероятность успеха 3^{-n}.
   За какое время можно добиться вероятности успеха $1 - 2^{-n}$?

2. **Постройте сведение (полиномиальное или по Куку)**

   Цель этой задачи, получить NP-полноту MAX-CUT.

   NAE = not all equal значит, что в каждом клозе хотя бы 1 ноль и хотя бы 1 единица.
   MAX-CUT: дан взвешенный граф, разделить множество вершин на две непустых доли, максимизировать суммарный вес рёбер между долями.

   a) NAE-3-SAT $\to$ 3-SAT (подсказка: сведите для одного клоза)
   b) 3-SAT $\to$ NAE-4-SAT (подсказка: сведите для одного клоза)
   c) NAE-4-SAT $\to$ NAE-3-SAT (подсказка: сведите для одного клоза)
   d) NAE-3-SAT $\to$ MAX-CUT

3. **Числа в окрестности**

   Дан массив и число ε. Известно, что $\frac{2n}{3}$ элементов массива лежат в $[x, x+\varepsilon]$ для некоторого *неизвестного* x. За линейное время выбрать $\frac{n}{3}$ элементов из $[y, y+\varepsilon]$ для некоторого y.

4. **Random в задачах оптимизации**

   Пусть у нас есть алгоритм, который ищет VERTEX-COVER.
   На любом графе матожидание размера покрытия, которое выдаст алгоритм, равно $C \cdot \mathtt{OPT}$.
   Как получить покрытие, которое с высокой вероятностью будет размера не более $C(1+\varepsilon)\mathtt{OPT}$?

5. **Приближение** MAX-SAT

   Для MAX-SAT построить полиномиальный вероятностный алгоритм, ищущий набор переменных, обращающий хотя бы $\frac{1}{2}\mathtt{OPT}$ клозов в истину.

   Обсудите и детерминированный, и рандомизированный. Какао матожидение?

6. **Первообразный корень**

   Первообразный корень по модулю p – такое $x\colon \langle 1, x, x^2, \ldots, x^{p-2} \rangle$ различны. Дано p, найти x.

7. **Проверка ДЗ**

   Дано ДЗ из $n{=}2k$ задач. Пусть студент в каждой из задач делает ошибку с вероятностью $\frac{1}{3}$.
   Ошибки во всех задача независимы.

   Алгоритм #1 проверки дз: проверить первую половину присланных решений. Пусть обнаружено 0 ошибок, какова вероятность наличия ошибок/матожидание количества ошибок?

   Алгоритм #2 проверки дз: проверить *случайную* половину присланных решений. Пусть обнаружено 0 ошибок, какова вероятность наличия ошибок/матожидание количества ошибок?

8. **RandomShuffle**

   Дан массив, перемешайте его элементы так, чтобы все перестановки были равновероятны.

9. **Random walk**

   На лекции мы поговорили про random walk решение 3-SAT. По аналогии постройте random walk решение 4-SAT. Оцените вероятность успеха так, чтобы было α^n для $\alpha < \frac{1}{2}$.

10. **Вложения**

    Классы EXP, NP, RP, ZPP, P, NEXP. Вспомните кто куда вкладывается? Почему?

11. (*) 3-COLORING $\to$ EXACT-SET-COVER

    EXACT-SET-COVER: даны U и $\{B_i \subseteq U\}$, есть ли такое $I\colon \cup_{i \in I} B_i = U \ \wedge \ \forall i, j \ B_i \cap B_j = \varnothing$.

12. (*) DOMINATING-SET $\in$ NPc

    DOMINATING-SET: найти min $A \subseteq V\colon A \cup N(A) = V$.

13. (*) STEINER-TREE $\in$ NPc

    STEINER-TREE: даны взвешенный граф и $S \subseteq V$, найти поддерево min веса, содержащее все вершины S.

Разбор задач практики

1. **Понижение вероятности**

   a) После $\log n$ повторов вероятность ошибки $(1 - \frac{1}{\log n})^{\log n} \leqslant e^{-1}$.
   Всё это вместе $\ln n$ раз, тогда $\frac{1}{n}$. Итого $\log n \cdot \ln n$ повторов, время работы $\mathcal{O}(n^2 \log^3 n)$.

   b) $6^n \cdot n$

2. **Постройте сведение**

   a) 3-SAT $\leftrightarrow$ NAE-3-SAT.

   NAE-3-SAT $\to$ 3-SAT: запишем для каждого клоза $(l_1 \vee l_2 \vee l_3)$ парный $(\neg l_1 \vee \neg l_2 \vee \neg l_3)$. Решим SAT из $2m$ клозов.

   3-SAT $\to$ NAE-4-SAT: добавим в каждый клоз переменную w (одну и ту же).
   Было решение 3-SAT $\Rightarrow$ берем его и делаем $w = 0$.
   Есть решение $(y_1, y_2, \ldots, y_n, w)$ для NAE-4-SAT $\Rightarrow$ ответ для 3-SAT это $x_i = y_i \hat{\ } w$.

   NAE-4-SAT $\to$ NAE-3-SAT: i-й клоз из 4 литералов $(l_1 \vee l_2 \vee l_3 \vee l_4)$ переводим в клозы $(l_1 \vee l_2 \vee w_i)$, $(\neg w_i \vee l_3 \vee l_4)$, w_i – новая переменная (так же, как в 4-SAT $\to$ 3-SAT).
   Чтобы перевести решение NAE-4-SAT в решение NAE-3-SAT и обратно, разбор 5 случаев: $l_1 = l_2 = 0/1$, $l_3 = l_4 = 0/1$, иначе.

   b) NAE-3-SAT $\to$ MAX-CUT

   Дана NAE-3-SAT формула, нам нужно построить граф.

   Для каждой переменной x_i заводим две вершины: x_i, $\overline{x_i}$, соединяем их ребром.
   Каждый клоз преобразуем в три ребра: $\triangle$ на литералах.
   Получили граф из $2n$ вершин и $3m + n$ рёбер.

   В треугольнике разрез пересекают максимум 2 ребра.
   Итого размер MAX-CUT не более $n{+}2m$ (n рёбер между x_i и $\overline{x_i}$ и по 2 от каждого $\triangle$) $\Rightarrow$ ($\exists$ разрез размера $n{+}2m \Leftrightarrow \exists$ решение NAE-3-SAT).

3. **Числа в окрестности**

   Простой способ.
   Ткнем в случайное число a_i, выведем в ответ все числа из $[a_i..a_i + \varepsilon]$.
   Обозначим за B минимальные $\frac{n}{3}$ чисел из $[x..x{+}\varepsilon]$. Если $a_i \in B$, то на $[a_i..a_i{+}\varepsilon]$ лежит хотя бы $\frac{n}{3}$ чисел. Вероятность попасть в B равна $\frac{n/3}{n} = \frac{1}{3}$.

   Надежный способ. Найдем статистики с номерами $\frac{n}{3}$ и $\frac{2n}{3}$, берем всё между ними.

4. **Random в задачах оптимизации**

   Алгоритм: запустить много раз, выбрать наименьший ответ.

   Марков: $\mathsf{Pr}[x \geqslant 2 \cdot \mathsf{E}[x]] \leqslant \frac{1}{2}$, более общо $\mathsf{Pr}[x \geqslant (1{+}\varepsilon) \cdot \mathsf{E}[x]] \leqslant \frac{1}{1+\varepsilon}$. По неравенству Маркова на одном запуске ответ будет $>C(1{+}\varepsilon)\mathtt{OPT}$ с вероятностью $<\frac{1}{1+\varepsilon} = 1{-}\frac{\varepsilon}{1+\varepsilon}$ (вероятность ошибки).

   Обозначим $p = \frac{\varepsilon}{1+\varepsilon}$, после $\frac{1}{p} = \frac{1+\varepsilon}{\varepsilon}$ запусков Pr[ошибки] $= (1-p)^{\frac{1}{p}} \leqslant e^{-1}$.

5. **Приближение** MAX-SAT

Детерминированный способ.
Возьмём произвольные значения переменных, если выполнено меньше $\frac{m}{2}$ клозов, инвертируем все переменные. Итого: мы выполнили $\geqslant \frac{m}{2} \geqslant \frac{1}{2}\mathtt{OPT}$ клозов.

Рандомизированный способ. Подставим все переменные случайно.
Если в клозе k литералов, он выполнился с вероятностью $Pr = (1-2^{-k})$ (из 2^k вариантов нам не подходит ровно 1). Заметим, что $k \geqslant 1 \Rightarrow Pr \geqslant \frac{1}{2}$. E[количества выполненных клозов] $= \sum((1-2^{-k_i})) \geqslant \frac{1}{2}m \geqslant \frac{1}{2}\mathtt{OPT}$.

Но нам нужно не матожидание, а гарантия. Воспользуемся предыдущей задачей. Максимизировать число выполненных клозов $\Leftrightarrow$ минимизировать число нарушенных.
Если отклонились от X менее, чем на $\frac{1}{2}$, то из-за целочисленности попали в X.
$\frac{1}{2}m + \frac{1}{2} = \frac{1}{2}m(1+\frac{1}{m})$. Пользуемся понижением вероятности для $\varepsilon = \frac{1}{m}$, нужно

$$\frac{1+\varepsilon}{\varepsilon} = \frac{1+1/m}{1/m} = m+1$$

запусков.

6. **Первообразный корень**

Идея. Ткнём в случайное число из $[2, p-1]$ и проверим. Будем тыкать, пока не найдем.

Оценка. g – первообразный $\Rightarrow \forall k\colon (k, p-1) = 1$ g^k тоже первообразный.
Итого первообразных корней $\varphi(p-1) \Rightarrow$ вероятность попасть $\frac{\varphi(p-1)}{p-1} \geqslant \frac{1}{\log\log p}$.

Проверка числа g. Достаточно убедиться, что $x^{(p-1)/d} \neq 1$ для всех простых $d \mid (p-1)$.
Для этого один раз факторизуем $p-1$ и каждую проверку будем делать за $\mathcal{O}(\log^2 p)$. Факторизовать $p-1$ сейчас умеем за $\mathcal{O}(p^{1/4}\log p)$.

Итого. $\mathcal{O}(p^{1/4}\log p + k\log^2 p \log\log p)$ для вероятности ошибки e^{-k}.

7. **Проверка ДЗ**

$n{=}2k$. Если мы проверяем первую половину дз, то в каждой из оставшихся задач $\frac{1}{3}$ ошибок $\Rightarrow \mathsf{E} = \frac{1}{3}\cdot k$, вероятность наличия ошибок $1-\left(\frac{2}{3}\right)^k$. Если мы проверяем случайную половину, ровно также. Непроверенные биты всё равно не зависят от проверенных.

8. **RandomShuffle**

Строим по индукции. Всвопываем новый элемент в случайное место перестановки.

```
    void Shuffle(int n, int *a)
        for (int i = 0; i < n; i++)
            swap(a[i], a[rand() % (i+1)]);
```

Док-во: индукция. База: после фазы $i{=}0$ все перестановки длины 1 равновероятны.
Переход: выбрали равновероятно элемент, который стоит на i-м месте, после этого часть массива $[0..i)$ по индукции содержит случайную из $i!$ перестановок.

9. **Random walk**

$\sum_k \frac{C_n^k}{2^n}4^{-k} = \frac{1}{2^n}\cdot(1+\frac{1}{4})^n = \left(\frac{5}{8}\right)^n$

10. **Вложения**

P $\subseteq$ ZPP $\subseteq$ RP $\subseteq$ NP $\subseteq$ EXP $\subseteq$ NEXP

11. (*) 3-COLORING $\to$ EXACT-SET-COVER

Универсум: все вершины, все ребра и все тройки $\langle v, e, c\rangle$, где v – конец e, c – один из цветов (итого $n + m + 6m$ элементов). Множества:

a) $\forall$ вершины v, цвета c: $\{v\} \cup \{\langle v, e, c\rangle \mid e = (v, u)\}$.
Если взято, то красим v в цвет c. Каждая вершина покрасится, причем в один цвет.

b) $\forall$ ребра $e = (v, u)$, пары цветов $c_1 \neq c_2$: $\{e\} \cup \{\langle v, e, c\rangle \mid c \neq c_1\} \cup \{\langle u, e, c\rangle \mid c \neq c_2\}$.
Раз каждое ребро чем-то покрыто, то оно «забирает с собой» все цвета у своих концов, кроме двух неравных.

12. (*) DOMINATING-SET $\in$ NPc: VERTEX-COVER $\to$ DOMINATING-SET.

Из графа $G = (V, E)$ делаем новый $G' = (V', E')$.
$V' = V \cup E$. $E' = \{(v, u) \mid v, u \in V\} \cup \{(v, e) \mid e = (v, u) \in E\}$.
Чтобы задоминировать все вершины, которые соответствуют старым ребрам, нужно как раз набрать VERTEX-COVER. Полный граф на старых вершинах нужен, чтобы они сами собой задоминировались.

13. (*) STEINER-TREE $\in$ NPc: SET-COVER $\to$ STEINER-TREE.

Вершины графа – элементы и множества.
Ребра веса 0 между множествами и лежащими в них элементами.
Фиктивная вершина v_0, соединенная ребрами веса 1 со всеми множествами.
S – множество элементов. Вес min дерева = (количеству множеств в min SET-COVER $-\, 1$).

Домашнее задание

3.1. Обязательная часть

1. **(1) Приближённый** MAX-3-SAT

   Придумайте приближенный ZPP алгоритм с константой лучше $\frac{3}{4}$ для MAX-3-SAT, про который дополнительно известно что в каждом клозе *ровно три различные переменные.*

2. **(2) Понижение вероятности**

   Алгоритм работает за $\mathcal{O}(n^3)$, вероятность **успеха** $1/n^3$ (ошибка односторонняя).
   За какое время можно добиться вероятности успеха $1 - 1/2^n$?

3. **(2) Randomized NP**

   Что будет, если в NP проверяющий подсказку алгоритм будет из ZPP, а не из P? Как полученный класс соотносится с уже известными?

   Сперва покажите, что этот класс содержит NP. *Задачу сдавайте до мягкого дедлайна так как, опыт прошлых поколений показывает вероятность ошибок в решении* ≈100*%.*

4. **(2) Сдать Полларда**

   Эта задача ровно про ваш контест.

   В вашей тестирующей системе $C_1 = 100$ тестов. Поллард работает с вероятностью $\frac{1}{2}$.
   Сколько раз нужно запустить Полларда, чтобы с вероятностью $1 - C_2 = 1 - \frac{1}{10}$ получить `OK`?
   Больше баллов получат простые вычисления, которые можно сделать в уме для $\forall\ C_1, C_2$.

5. **(2)** EXACT-SET-COVER → SUBSET-SUM.

   EXACT-SET-COVER: даны U и $\{B_i \subseteq U\}$, есть ли такое $I\colon \cup_{i \in I} B_i = U \ \wedge\ \forall i, j\ B_i \cap B_j = \varnothing$.
   SUBSET-SUM: набрать сумму ровно S.
   Тут и дальше придётся думать.

3.2. Дополнительная часть

1. **(2)** SUBSET-SUM → PARTITION → JOB-SHEDULING

   PARTITION: разделить множество предметов на два, равных по суммарному весу.
   JOB-SHEDULING: даны n заказов, у каждого дано время выполнения t_i. Есть k одинаковых рабочих, распределить заказы по рабочим: время завершения всех работ → min.

2. **(3)** MAX-2-SAT ∈ NPc

3. **(3+2) Полиномиальные нижние оценки!**

   a) **(3)** Покажите SETH ⇒ нельзя найти доминирующее множество размера k за $\mathcal{O}(n^{k-\varepsilon})$.
   b) **(2)** Покажите SETH ⇒ нельзя найти доминирующее множество размера k за $\mathcal{O}(n^{k-1-\varepsilon})$ в графах с маленькой средней степенью ($m = \mathcal{O}(n \operatorname{polylog}(n))$).