1.♣

Подойдут, например $f(n) = 2^n$ и $a(n) = 2^n$. Подсказка такого размера нужна, чтоб в неё можно было уложить бит (принадлежит или не принадлежит) для каждого слова длины n, экспоненциальная временная сложность нужна, чтоб прочесть подсказку.

1.♢

Подойдут, например $f_1(n) = n^3$ и $f_2(n) = n^4$. Подсказки такой длины не увеличивают класс языков, поскольку распознаватель не успеет прочесть подсказку за ограниченное n^2 время.

Критерии:

- Нет решения или написанный текст не ведет к решению задачи: 0
- В написанном есть необходимые для решения идеи, но полное решение не приведено: 0.25
- В написанном есть все необходимые идеи, но восстановить детали доказательства не получается: 0.5
- В аргументации есть небольшие огрехи, решение в целом верное: 0.9
- Полностью верное решение: 1

2.♣

Можно просто перебрать все делители числа x перебором от двойки. Если в процессе перебора по z число x поделилось на z, проверяем, что z простое – это тоже можно делать перебором. Отдельно считаем сумму. Все числа – линейны по памяти от входа, деление также требует полиномиальной памяти. Естественно, алгоритм можно существенно улучшить.

Критерии

0,25 - Идея верная, но исходная задача не решена

0,5 - Верное решение, но упустили проверку делителей на простоту

0,75 - Верное решение, но не обосновали проверку делителей на простоту

1 - Верное решение

2.$\diamond$

Можно просто перебрать все последовательности вершин (в алфавитном порядке, скажем) – сперва длины 1, затем 2 и так далее до числа вершин в графе n. Для каждой последовательности проверить, является ли она путем из s в t простым сравнением со входом. Если да, увеличивать отдельный счетчик, который требует не более линейной от входа памяти. Последовательность вершин также линейна по входу, так как вход сам состоит из n вершин. Естественно, алгоритм можно существенно улучшить.

Критерии

0,2 - В качестве решения предоставлен только поиск в глубину/LSS

0,5 - В решении нет проверки пути на простоту, что существенно влияет на корректность алгоритма

0,8 - В решении нет проверки пути на простоту, но эту проверку можно добавить в влгорит без существенных изменений

1 - Верное решение

3.♣

1) Это класс всех языков ALL. В самом деле, машина, которая принимает все строки, подходит (её вероятность принять слово есть $1 > 4/5$, отвергнуть – $0 < 1/5$).

2) Это $\mathcal{BPP}$. Пусть $L \in X_2$, то запустим машину, которая существует по условию, два раза на случайных r и примем слово, если оба запуска приняли слово, иначе отвергнем слово. Если слово принадлежит языку, то вероятность его принять - квадрат вероятности принятия одного запуска $\geqslant (5/6) * (5/6) = 25/36 \geqslant 5/9$. Если слово не принадлежит языку, то вероятность его принять - аналогично $< 4/9$, а вероятность отвергнуть тогда $\geqslant 5/9$. Тогда $L \in \mathcal{BPP}$ с границей двусторонней ошибки $4/9$, что равно стандартному $\mathcal{BPP}$. Пусть $L \in \mathcal{BPP}$. Тогда $L \in \mathcal{BPP}$ с границей двусторонней ошибки $1/6$, которая подходит под определение X_2.

3.◇

1) Это $\mathcal{BPP}$, поскольку существование такой M эквивалентно существованию M', которая моделирует M, но выдаёт противоположный ответ. M' подтверждает принадлежность языка $\mathcal{BPP}$ с границей двусторонней ошибки $1/5$, что равно стандартному $\mathcal{BPP}$.

2) Это $\mathcal{PP}$. Пусть $L \in X_2$, пусть M – машина, распознающая L в соответствии с определением X_2. Возьмём машину M', которая сперва с вероятностью $3/8$ отвергает слово, с вероятностью $1/8$ принимает слово, а с вероятностью $1/2$ далее запускает машину M и выдаёт её ответ. (Моделировать вероятности в долях $1/8$ можно тремя битами.) Тогда если слово принадлежит L, то M' принимает его с вероятностью $> 1/8 + 1/2 \cdot 3/4 = 1/2$. Если слово не принадлежит L, то M' принимает его с вероятностью $< 1/8 + 1/2 \cdot 3/4 = 1/2$, что удовлетворяет определению $\mathcal{PP}$. Пусть $L \in \mathcal{PP}$, пусть M – машина, распознающая L в соответствии с определением $\mathcal{PP}$. Возьмём машину M', которая сперва с вероятностью $1/2$ принимает слово, а с вероятностью $1/2$ далее запускает машину M и выдаёт её ответ. Тогда если слово принадлежит L, то M' принимает его с вероятностью $> 1/2 + 1/2 \cdot 1/2 = 3/4$. Если слово не принадлежит L, то M' принимает его с вероятностью $< 1/2 + 1/2 \cdot 1/2 = 3/4$, что удовлетворяет определению X_2.

Критерии

0.4 – X_1

0.2 балла ставилось если была получена правильная таблица ошибок и проделаны верные рассуждения, но сделан неверный вывод о классе языков в конце

0.6 – X_2

Ненулевой балл за этот пункт обычно ставился только при наличии корректного алгоритма запуска и интерпретации результатов данной МТ для получения искомой таблицы. Верные идеи, не доведенные до формального описания, не оценивались. Некорректные алгоритмы - тоже.

Фраза “используем амплификацию” без описания алгоритма обычно также приводила к 0 баллов за пункт(кроме известных с лекций и семинаров, например, для изменения ошибки в определении $\mathcal{BPP}$)

При правильном определении класса доказательство включения в каждую сторону оценивалось из 0.3 балла

4.♣ и **4.♢**

Определим машину M, которая принимает на вход пару (x, r). Используя x (которая интерпретируется как граф), машина M проверяет, что r состоит из двух частей: r_1 и r_2 одинаковой длины. Слово r_1 интерпретируется как сертификат (гам.цикла или клики) в подходящей кодировке, слово r_2 – как случайная строка. Если в графе x объект, кодируемый r_1 не является искомым гам.циклом или кликой (т.е. подан плохой сертификат), M отвергает пару и останавливается. Если же r_1 это нужный объект (гам.цикл или клика), то M читает r_2 и игнорирует его содержание (можно, к примеру, записывать бит сертификата на ленту и переписывать его каждый раз, чтоб не было сомнений, что разные r_2 порождают разные вычислительные пути). Если слово x не принадлежит языку L (гамильтоновы графы или графы с кликой), то каждый r отвергается, так что отвергается 100% всех путей. Если слово x принадлежит языку L, то по крайней мере один r_1 подходит в качестве сертификата. Если $|r_1| = n$, то отвергается не более $2^n - 1$ путей. Каждый подходящий r_1 (в худшем случае он уникален) порождает 2^n принимающих путей за счёт произвольности r_2. Итого, не менее 2^n путей принимается. Всего принимается не менее $2^n/(2^n - 1 + 2^n) > 50\%$ всех путей. Для получения оценки $\mathcal{BPP}$ достаточно взять r_2 длины $n + 1$, а не n, так как $2^{n+1}/(2^{n+1} + 2^n - 1) > 2/3$.

Критерии

0.1 – любые нетривиальные идеи, которые могли бы привести к решению. (Идея приравнять r к сертификату без подробностей кодирования является тривиальной, не ведет напрямую к решению и оценивалась в 0 баллов)

0.3 – попытка описать содержание r и комбинаторно оценить количество возможных "подходящих"сертификатов и вычислительных путей

0.4 – верная догадка о том, что "сертификат"должен являться только частью r для получения искомых соотношений

5.♣

1) Принадлежность $\mathcal{NL}$: сертификат это два пути через разделитель, сперва короткий путь, затем длинный. Машина проверяет, что каждый путь и правда является путём (одним проходом слева направо), при этом считает его длину в отдельный счётчик логарифмической по памяти длины. Всего нужно два счётчика. Затем (если сертификат корректный) машина просто сравнивает длины путей.

2) Принадлежность $\mathcal{NL}$-hard. Сводимость из DPATH следующего вида: добавим в орграф G путь длины $2|V|$ из s в t и уберём все исходящие из t рёбра и входящие в s рёбра. $(G,s,t) \to (G',s,t)$

Пусть $(G,s,t) \in$ DPATH. Тогда существует путь из s в t, тогда существует простой путь ν из s в t. Длина ν не более $|V|-1$. В преобразованном графе также есть путь ν (поскольку он простой, он не использует рёбер в s или из t) и есть добавленный по построению путь длины $2|V|$.

Пусть $(G,s,t) \notin$ DPATH. Тогда в преобразованном графе G' существует лишь один путь из s в t – добавленный. В самом деле, если выйти из s по другому ребру, вернуться в s будет нельзя, потому что в s не ведёт рёбер. Выйти из s и попасть в t не через добавленный пусть нельзя, поскольку в исходном графе пути из s в t нет. Наконец, дойдя по t по добавленному пути, дальше уйти (и потом вернуться в t) невозможно, потому что из t не выходит рёбер.

Очевидно, функция сводимости вычислима на лог.памяти: требуется удалить линейное число рёбер с фиксированными концами и добавить на выход линейное число вершин и рёбер, для чего нужно хранить константное число вершин и счетчик длины пути.

5.♢

1) Принадлежность $\mathcal{NL}$: сертификат это два пути через разделитель, из s в v и из t в v. Машина проверяет, что каждый путь и правда является путём (одним проходом слева направо), при этом она запоминает последние вершины в каждом пути, затем сравнивает их.

2) Принадлежность $\mathcal{NL}$-hard. Сводимость из DPATH следующего вида: добавим в орграф G вершину v и ребро (v,t), затем уберём все исходящие из t рёбра. $(G,s,t) \to (G',s,v)$.

Пусть $(G,s,t) \in$ DPATH. Тогда существует путь из s в t, тогда существует простой путь ν из s в t. Кроме того, существует путь (длины один) из v в t по построению. Тогда $(G',s,v) \in$ WEAK-PATH.

Пусть $(G',s,v) \in$ WEAK-PATH. Тогда в преобразованном графе G' существует вершина, в которую можно попасть и из s, и из v. Но из v можно попасть лишь в t, дальше рёбер нет (по построению G'). Значит, вершина, слабо связывающая s и v – это вершина t. Значит, в G' есть (простой) путь из s в t, а тогда он есть и в исходном графе G.

Очевидно, функция сводимости вычислима на лог.памяти: требуется удалить линейное число рёбер с фиксированными концами и добавить на выход одну вершину и ребро.

Критерии

0.2 балла ставились за каждый из следующих пунктов решения:

1) Доказательство принадлежности языка классу $\mathcal{NL}$.

2) Построение корректной сводимости $\mathcal{NL}$-полного языка к языку из условия задачи.

3-4) Доказательства корректности сводимости в одну и другую сторону при условии того, что сводимость корректна.

5) Доказательство того, что функция сводимости вычислима на логарифмической памяти, при условии, что она корректна.

6.♣ и **6.◇**
1) Пусть $n = 2^k$, пусть c_j – обозначает количество строк, в которых число j покрашено. Рассмотрим случайное (равновероятно) слово v. Рассмотрим вероятность того, что число j не будет включено в S_v. Для этого каждый раз, когда число j окрашено, цвет в слове v должен быть "не подходящим". Для случайного v вероятность этого равна $\left(\frac{1}{2}\right)^{c_j} \leqslant \frac{1}{2^{\log n+1}} = \frac{1}{2n}$. Пусть X_j – случайная величина, равная единице, если j не вошло в S_v, и нулю, если j вошло в S_v. Матожидание $E[X_j] \leqslant 1 \cdot \frac{1}{2n} + 0 \cdot \ldots = \frac{1}{2n}$. Пусть $X = \sum_{j=1}^{n} X_j$ (количество чисел, не вошедших в S_v), тогда $E[X] \leqslant n \cdot \frac{1}{2n} = \frac{1}{2}$. Если для любого выбора v по крайней мере одно число остаётся не включённым в S_v, то матожидание X (как среднее по всем v) равно как минимум единице. Поскольку в нашем случае матожидание меньше 1/2, то существует слово v такое, что S_v содержит все числа от 1 до n.
2) Будем последовательно идти по строкам таблицы. В каждой строке нам необходимо выбрать красный или синий цвет. Пусть первые $i-1$ строки уже обработаны, обрабатываем строку номер i. Матожидание количества чисел, не включенных в финальное множество до выбора цвета в строке номер i равно

$$E[X \mid v_1 = a_1, \ldots, v_{i-1} = a_{i-1}]$$

Здесь a_i – уже выбранные цвета R или B. Если будет выбран цвет R, то матожидание станет равно

$$E[X \mid v_1 = a_1, \ldots, v_{i-1} = a_{i-1}, v_i = R]$$

Если будет выбран цвет B, то матожидание станет равно

$$E[X \mid v_1 = a_1, \ldots, v_{i-1} = a_{i-1}, v_i = B]$$

Нужно выбрать то из двух, которое меньше. Рассмотрим подробнее, чему они равны. Каждое число, которое уже было выбрано в силу выбора $a_1, \ldots, a_{i-1}$ добавляет нуль в матожидание. Каждое прочее число добавляет $1/2^p$ в матожидание, где p – количество оставшихся необработанных строк, в которых это число покрашено. Пусть A_R – сумма добавок в матожидание по всем красным числам строки i, A_B – то же, но по синим. Выбор красного цвета обнуляет всё A_R, то есть вычитает A_R из матожидания, при этом A_B удваивается, так как каждое число теперь встречается покрашенным на один раз меньше. Выбор синего цвета делает наоборот. Тогда мы либо прибавляем $A_B - A_R$, либо $A_R - A_B$ к матожиданию – нужно выбрать из этих чисел меньшее (неположительное) и тем самым уменьшить матожидание.
Иначе можно понять решение так. Сопоставляем каждому числу j величину p_j, которая интерпретируется как "вероятность числу j не попасть в финальное множество". Вычисляем начальное значение всех $p_j = 1/2^{n_j}$, где n_j – число строк, в которых j встречается в покрашенном виде. Общая сумма $p = \sum_{j=1}^{n} p_j$ меньше единицы, так как $p_j \leqslant \frac{1}{2^{\log n+1}} = \frac{1}{2n}$. Пусть в строке i множество R_i – это красные числа, а B_i – синие. Тогда вычислим

$$T = \sum_{j \in R_i} p_j - \sum_{j \in B_i} p_j$$

Если эта разность отрицательна, выберем в строке i синий цвет, иначе выберем красный. После выбора цвета положим $p_j = 0$ для всех выбранных чисел и удвоим p_j для всех невыбранных (это означает, что новое значение p станет равно старому значению p плюс T, то есть p не увеличивается). После обработки всех строк каждое число добавляет в p нуль, если оно было выбрано, или один, если не было. Значит в конце p целое неотрицательное. Но p было меньше единицы вначале и не увеличивалось в процессе работы алгоритма. Тогда $p = 0$ и все числа выбраны.
Критерии
1) 0.5 первый пункт, из них
– 0.3 за корректную оценку матожидания/вероятности получения набора из (не) всех чисел при случайном выборе цвета,
– 0.2 за корректную интерпретацию, почему из этого следует требуемое в пункте утверждение
2) 0.5 за второй пункт
Полный балл за этот пункт ставился только при наличии описания того, как эффективно произвести выбор цвета

Отдельно ставилось -0.1 балла в пунктах за неясные комментарии, непонятные обозначения и т.п.

Комментарии

Частая ошибка – “рассмотрим вероятность клетки быть покрашенной”. Это не имеет смысла, поскольку раскраска дана на вход и не является случайной величиной. Усреднение по всем раскраскам также не имеет смысла, поскольку требуется найти алгоритм, работающий на всех раскрасках, а не “в среднем по раскраске”, что бы это ни значило.

По этой же причине вероятность числа быть или не быть взятым не может явно зависеть от m. Любые такие решения автоматически неверны.

Во втором пункте часто писали “будем выбирать так, чтоб условное матожидание было выше” (или ниже в зависимости от решения). Это отражает суть метода, но не описывает решение – весь смысл в том, что не так-то просто понять, как максимизировать условное матожидание (иначе дерандомизация была бы тривиальной). Без явного предоставления эффективной процедуры решение не считается предоставленным.