ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
МОСКОВСКИЙ ФИЗИКО-ТЕХНИЧЕСКИЙ ИНСТИТУТ
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

Алгоритмы и модели вычислений.
Домашняя работа 2.

Георгий Никишин
Группа Б05-909

Задача 1.

Докажите, что следующие языки принадлежат классу P. Считайте, что графы заданы матрицами смежности.

а)язык двудольных графов, содержащих не менее 2021 треугольников;

б)язык несвязных графов без циклов;

в)язык квадратных $\{0;1\}$-матриц порядка n $\geq$ 3000, в которых есть квадратная подматрица порядка n $-$ 2021, заполненная одними единицами;

д) язык $L(a,m)$, зависящий от параметров $a,m \in \mathbb{N}$, определяемый следующим образом:

$$L = \{x_0, x_1 \ldots | \ \ x_0 = a(mod \ \ m), \ \ x_{i+1} = x_i^{2021}(mod \ \ m)\}$$

Решение.

а) Очевидно, что искомый язык является пересечением языков двудольных графов и графов, содержащих не менее 2021 треугольников. Заметим, что если граф является двудольным, то в нем не может быть циклов, то есть в том числе треугольников $\Rightarrow$ искомый язык пуст $\Rightarrow$ лежит в P. б)Будем решать задачу с помощью поиска в глубину. Тк искомый язык является пересечением языка несвязных графов и графов без циклов, то составим два алгоритма на на основе поиска в глубину, каждый из которых будет решать свою задачу.

1) Модифицируем известный алгоритм поиска в глубину следующим образом: заведем переменную, в которой будем хранить количество посещенных вершин, тогда, если по завершению программы ее значение будет равно n(количеству вершин), то граф связен, тк мы посетили все его вершины, иначе остались непосещенные верщщины, которые лежат не в этой компоненте связности $\Rightarrow$ граф не является связным. Сложность данного алгоритма $O(|V| + |E|)$, те в за - $O(n)$, где $n-$ количество вершин.

2) Произведём серию поисков в глубину в графе. Те из каждой вершины, в которую мы ещё ни разу не приходили, запустим поиск в глубину, который при входе в вершину будет красить её в серый цвет, а при выходе - в чёрный. И если поиск в глубину пытается пойти в серую вершину, то это означает, что мы нашли цикл (если граф неориентированный, то случаи, когда поиск в глубину из какой-то вершины пытается пойти в предка, не считаются).Этот алгоритм также работает за линейное время.

Таким образом, запуская два алгоритма на данном графе мы поймем, принадлежит ли он языку за линейное время $\Rightarrow$ язык лежит в классе P.

в) Воспользуемся динамическим программированием для решения этой задачи. Введем дополнительную переменную $d[i;j]$-ближайший сверху ноль для элемента $A[i;j]$, то есть $d[i;j]$ равняется наибольшему номеру строки, в которой в j столбце стоит нолик, если такой строки нет, то ставим туда -1 для определенности. Тогда значение массива D, очевидно, вычисляется за один проход по матрице. Тогда, перебирая для текущей строки номер левого и правого столбца искомой подматрицы, а затем, обращаясь к вычисленному массиву D, вычисляя ее верхнюю границу, найдем наибольшую подматрицу исходной матрицы, состоящую из одних единичек. Если максимум из строк и столбцов найденной матрицы $\geq n - 2021$, то матрица принадлежит языку, если нет, то не принадлежит. Оценим сложность алгоритма мы вычисляем массив D за $O(n^2)$, проходя по всей исходной матрице, затем, в основной части алгоритма, находим наибольшую подматрицу за $O(n^3)$, после чего проверяем ее наибольшую квадратную подматрицу на условие $\geq n - 2021$ за $O(n)$. Итоговая сложность $O(n^3) \Rightarrow$ язык лежит в P.

Задача 2.

а) Верно ли, что алгоритм деления столбиком нацело $F(a,b) = \lfloor a/b \rfloor$ полиномиален?

б) Верно ли, что алгоритм быстрого возведения в степень $F(a,b) = a^b$ полиномиален?

в) Существует ли полиномиальный алгоритм вычисления функции $f(n,k) = \lfloor \sqrt[k]{n} \rfloor$?

Решение.

а) Рассмотрим алгоритм, который был разобран на курсе алгоритмов в прошлом году:

```
функция DIVIDE(x, y)
{Вход: n-битовые x и y, причём y ⩾ 1.}
{Выход: частное и остаток от деления x на y.}
если x = 0:   вернуть (q, r) = (0, 0)
(q, r) ← DIVIDE(⌊x/2⌋, y)
q ← 2·q, r ← 2·r
если x нечётно:   r ← r + 1
если r ⩾ y:   r ← r − y, q ← q + 1
вернуть (q, r)
```

Такой алгоритм, очевидно, работает за полиномиальное время от длины входа (если быть точным за квадрат) $\Rightarrow$ да, верно. б) Пусть $b = (\overline{b_k b_{k-1} \cdots b_1 b_0})_2$- двоичное представление для b, где $b_k = 1$, $m_i \in \{0;1\} i \in \overline{1; k-1}$, тогда:
$a^b = a^{((\cdots((b_k \cdot 2 + m_{k-1}) \cdot 2 + m_{k-2}) \cdot 2 + \cdots) \cdot 2 + m_1) \cdot 2 + m_0}$, опишем работу алгоритма словами.
Рассмотрим двоичное представление b, если на $i-$ом шаге $b_i = 1$, то имеющееся число возводится в квадрат и к результат возводится в квадрат, после чего еще домножается на a, если же $b_i = 0$, то домножение на a не происходит. Оценим сложность алгоритма: всего у нас шагов порядка $n = \log b$, на каждом шаге операции требуют $O(n^2) \Rightarrow$ итоговая сложность $O(n^3)$
в) Известно, что существует Алгоритм извлечения корня n-ной степени сдвигом , который работает аналогично алгоритму деления, то есть за полиноиальное время от длины входа. Приведем псевдокод данного алгоритма .
root = num / rootDegree -начальное приближение корня
rn = num - значение корня последовательным делением
countiter = 0 - число итераций
While abs(root - rn) >= eps:

rn = num;

for(int i = 1; i < rootDegree; i++)

rn = rn / root;

root = 0.5 * (rn + root);

countiter++

Здесь умножение на 0.5 эквивалентно сдвигу в двоичной записи числа. Таким образом алгоритм работает за полином от длины входа, тк совершается полиномиальное число шагов и сложность на каждом шаге полиномиальна.

Задача 3.

a) Сформулируйте напрямую определение класса coNP (без ссылки на класс NP).
b) Докажите, что замкнутость NP относительно дополнений влечёт равенство NP = coNP.
c) Докажите, что класс NP $\cap$ coNP замкнут относительно дополнений.

Решение.

a) $coNp = \{L \subset \Sigma^*\}$, L: $\exists MT \quad M, p \in Poly : L = \{x \in \Sigma^* \mid \forall y : |y| < p(|x|); \quad M(x;y) = 0\}$
b) Согласно опредлению, данному на семинаре $coNP = \{A \mid \overline{A} \in NP\}$, предположим, что NP замкнут относительно дополнений, тогда $\forall A \in NP \longrightarrow \overline{A} \in NP \Rightarrow NP = coNP$
c) Пусть $A \in$ NP $\cap$ coNP $\Rightarrow \quad A \in coNP \Rightarrow \quad \overline{A} \in NP$, рассмотрим $\overline{A} : \overline{A} \in NP, \; A = \overline{\overline{A}} \in NP \Rightarrow \overline{A} \in coNP$, чтд.

Задача 4.

Докажите, что следующие языки принадлежат классу NP:
a) CLIQUE = (G, k) | в графе G максимальная клика имеет размер не меньше k;
b) GI = (G1, G2) | графы G1 и G2 изоморфны;
c) язык несовместных систем линейных уравнений с целыми коэффициентами от 2021 неизвестных.

Решение.

a)Воспользуемся утверждением, доказанным сегодня на семинаре: задача о независимом множестве вершин является NP- полной. Покажем, как задача о клике сводится к задаче о независимом множестве:рассмотрим дополнение графа, тогда если в исходном графе была клика размера k, то в его дполнении ,очевидно, будет независимое множество размера k, таким образом данные задачи эквиваленты$\Rightarrow$ язык лежит в NP.
b) Воспользуемся определением NP с помощью сертификатов, данным на семинаре. В качестве сертификата в данной задаче предоставим изоморфизм, отображающий G_1 в G_2. Тогда мы, очевидно, сможем проверить за линейное время, являются ли графы изоморфными, используя нашу подсказку, просто проводя соответсвие между вершинами, причем время работы будет линейным относительно количества вершин $\Rightarrow$ язык лежит в NP.
c) Воспользуемся критерием Кронокера-Капелли, который гласит, что система линейных алгебраических уравнений совместна тогда и только тогда, когда ранг её основной матрицы равен рангу её расширенной матрицы. Тогда будем применять к матрице системы и к расширенной матрице преобразования Гаусса, которые позволят за полнимиальное время выделить в них главные миноры и определить их ранг, таким образом, получаем, что язык лежит в $P \Rightarrow$ лежит в NP.