1. Скорость роста длины записи коэффициентов при реализации метода Гаусса.

Асимптотика метода Гаусса при работе с длинными числами O(M*n^3). Длина записи каждого из элементов матрицы не превосходит M.

2. Представление «длинного» числа в файле (массиве, списке) как числа в системе счисления по модулю p. Запись из файла. Оценка числа шагов. Вывод в файл. Оценка числа шагов.

Представление

 # (у Косовской в методичке m, в названии вопроса p, наверное можно просто заменить m на p)

При $m \geq 2$ любое целое неотрицательное число можно представить в m-ичной системе счисления в виде:

$$x = m^{k-1}x_0 + m^{k-2}x_1 + \ldots mx_{k-2} + x_{k-1}.$$

Здесь k - длина записи m-ичного представления числа x, $0 \leq x_i \leq m-1$ при $i = 0, \cdots, k-1$.
Числа $x_0, \cdots, x_{k-1}$ будем называть **макроцифрами**.

Замечание (важное)
Если m - максимальное целое, которое может быть записано в одну ячейку памяти, то представление в виде списка

k	x_{k-1}	...	x_0

удобно для хранения и осуществления операций с числами произвольной разрядности.

Запись из файла

В файле записано десятичное число, заданное словом $a_1 \cdots a_n$, $0 \leq a_i \leq 9$. Хотим представить его динамическим массивом (или списком), т.е.

k	x_{k-1}	...	x_0

Алгоритм выполняется за **квадратичное от длины *десятичной* записи** исходного числа количество шагов.

$$O(n^2)$$

Вывод в файл

У нас есть неотрицательное многоразрядное число, записанное в массив.

k	x_{k-1}	...	x_0

Хотим вывести его десятичную запись, заданную словом $a_1 \cdots a_n,\ 0 \le a_i \le 9$, в файл.

Алгоритм выполняется за **линейное от длины *десятичной* записи** исходного числа количество шагов.

$$O(n)$$

3. Сложение двух «длинных» положительных чисел. Оценка числа шагов.

У нас есть два неотрицательных многоразрядных числа, записанных в массивы A и B.

k_1	x_{k_1-1}	...	x_0

k_2	y_{k_2-1}	...	y_0

Хотим вычислить x + y.

Алгоритм выполняется за **линейное от максимума из длин записей** исходных чисел количества шагов.

$$O(max\{A[0],\ B[0]\}) = O(\max\{k_1,\ k_2\})$$

4. Предикаты равенства и неравенств «длинных» положительных чисел. Оценка числа шагов.

У нас есть два многоразрядных числа x и y, записанных в массивы A и B.

k_1	x_{k_1-1}	...	x_0

k_2	y_{k_2-1}	...	y_0

Хотим вычислять предикаты $x = y$, $x \ne y$, $x < y$, $x \le y$.

Так как $x \ne y \Leftrightarrow \neg(x = y)$ и $x \le y \Leftrightarrow \neg(y < x)$ или $x \le y \Leftrightarrow \neg(y < x) \vee x = y$, то достаточно уметь вычислять значения предикатов $x = y$ и $x < y$.

Алгоритм выполняется за **линейное от минимума из длин записей** исходных чисел количества шагов.

$$O(\min\{A[0],\ B[0]\})\ =\ O(\min\{k_1,\ k_2\})$$

5. Вычитание двух «длинных» положительных чисел. Оценка числа шагов.

У нас есть два многоразрядных числа x и y, записанных в массивы A и B.

k_1	x_{k_1-1}	...	x_0

k_2	y_{k_2-1}	...	y_0

Хотим вычислить $x-y$.

может понадобиться:

Запись отрицательного числа

Это можно сделать, например:

- Отведя дополнительный элемент массива $A[-1]$ или $A[A[0]+1]$, в котором будет храниться 0 для положительных чисел и 1 для отрицательных (или любой другой признак положительности/отрицательности)
- Храня знак числа в $A[0]$, но при этом по всех случаях, описанных выше, следует писать $|A[0]|$

В любом случае, будем считать, что $A[0]\ >\ 0$ и имеется признак знака числа

Алгоритм выполняется за **линейное от максимума из длин записей** исходных чисел количества шагов.

$$O(max\{A[0],\ B[0]\})\ =\ O(\max\{k_1,\ k_2\})$$

6. Умножение «длинного» числа на короткое. Оценка числа шагов.

У нас есть многоразрядное число, записанное в массив A, и макроцифра C.

k	x_{k-1}	...	x_0

C

Хотим вычислить их произведение, т.е. xC.

Алгоритм выполняется за **линейное от длины записи** исходного числа количества шагов.

$$O(A[0]) = O(k)$$

7. Умножение «длинных» чисел. Оценка числа шагов.

У нас есть два многоразрядных числа x и y, записанных в массивы A и B.

k_1	x_{k_1-1}	...	x_0

k_2	y_{k_2-1}	...	y_0

Хотим вычислить их произведение, т.е. xy.

Алгоритм выполняется за **произведение длин записей** исходных чисел количества шагов. **(полином от длины записи чисел)**.

$$O(A[0]B[0]) = O(k_1k_2)$$

8. Деление «длинных» чисел. Оценка числа шагов.

У нас есть два многоразрядных числа x и y, записанных в массивы A и B.
Предполагаем, что число, записанное в A, больше числа, записанного в B (иначе неполное частное равно 0, а остаток совпадает с делимым).

k_1	x_{k_1-1}	...	x_0

k_2	y_{k_2-1}	...	y_0

Хотим посчитать результат деления x на y (неполное частное и остаток).

Будем подбирать неполное частное делением промежутка, в котором оно может находиться, пополам.

Алгоритм выполняется за **произведение длин записей исходных чисел и их разности** количества шагов. **(полином от длины записи чисел)**.

$$O(A[0]B[0](A[0]-B[0])) = O(k_1k_2(k_1-k_2))$$

9. Скорость роста длины записи коэффициентов при реализации метода Гаусса

- Алгоритм Гаусса O(n^3).
- Метод Гаусса является является точным, но неустойчивым.
- Теорема Сильвестра: для всех k >= 2 каждый элемент b^{k}_{ij} (i = k + 1 ,..., m, j = k + 1 , … , n) делится нацело на $b^{k-2}_{(k-1)(k-1)}$.

10. Сортировки и оценки числа их шагов: пузырёк, сортировка слияниями фон Неймана.

- Сложность сортировки пузырьком O(n^2).
- Сложность сортировки пузырьком при работе с многоразрядными числами O(k*(n^2)). Где k - максимальное количество макроцифр в сортируемых многоразрядных числах.
- Сложность сортировки вставками O(n^2).
- Сложность сортировки вставками при работе с многоразрядными числами O(k*(n^2)). Где k - максимальное количество макроцифр в сортируемых многоразрядных числах.
- Сложность сортировки слияниями Фон Неймана O(n*log2n).
- Сложность сортировки слияниями Фон Неймана при работе с многоразрядными числами O(k*(n*log2n)). Где k - максимальное количество макроцифр в сортируемых многоразрядных числах.
- Сложность сортировки подсчетом O(n+N). Где N - наибольшее значение массива.
- Сложность сортировки подсчетом при работе с многоразрядными числами O(k*(n+N)). Где k - максимальное количество макроцифр в сортируемых многоразрядных числах.

11. Алгоритмы на графах, различные способы представления графа в компьютере.

- Матрица смежности

2.1.1. Матрица смежности

Матрица смежности графа — это квадратная матрица $A_{n\times n}$, элементы которой определены так

$$a_{ij} = \begin{cases} 1 & \text{если } \{v_i, v_j\} \in E, \\ 0 & \text{иначе} \end{cases} .$$

2.1.2. Списки смежности

Списки смежности — это одномерный массив, i-ым элементом которого является список вершин, смежных с v_i, т.е. окружение вершины v_i.

Очевидно, что для нахождения всех вершин, смежных с v, требуется $deg(v)$ проверок. Просуммировав эту величину по всем v получаем $\sum_{v \in V} deg(v) = 2m$. [4]

- Списки смежности
- Матрица инцидентности

2.1.3. Матрица инцидентности

Матрица инцидентности графа — это матрица $B_{n \times m}$, элементы которой определены так

$$b_{ij} = \begin{cases} 1 & \text{если } v_i \in e_j, \\ 0 & \text{иначе} \end{cases} .$$

Основными свойствами такой матрицы являются следующие два: $\sum_{i=1}^{n} b_{ij} = 2$, $\sum_{j=1}^{m} b_{ij} = deg(v_i)$.

- Матрицы V и E

  В одномерном массиве V хранятся имена вершин. В двумерном массиве E хранятся пары номеров смежных вершин.

  Массив E′, в котором вместе с каждой парой {vi,vj}, присутствует пара {vj,vi}, отсортированные по первому элементу. Вместе с элементом vi массива V хранится номер строки массива E′, в которой впервые встречается пара {vi,u} для некоторой вершины u.
- У Романа спросила на коллоквиуме когда лучше юзать глубину а когда ширину.
  - Ответ Романа: ширину сказал при двудольности, со скрипом прошло. глубину кто-то другой говорил циклы мосты гальминтонов - приняла.
  - Для проверки на двудольность используется поиск в ширину, а для выделения остова графа используется поиск в глубину.

12. Алгоритм поиска в глубину. Оценки числа шагов в зависимости от способа представления графа.

13. Алгоритм поиска в ширину. Оценки числа шагов в зависимости от способа представления графа.

- Сложность алгоритма поиска в глубину (ширину) с использованием матрицы смежности O(n^2).
- Сложность алгоритма поиска в глубину (ширину) с использованием списков смежности O(n+m).
- Сложность алгоритма поиска в глубину (ширину) с использованием матрицы инцидентности O(nm).
- Сложность алгоритма поиска в глубину (ширину) с использованием массивов V и E O(n+m*logm).

14. Задачи, решаемые с помощью этих алгоритмов: выделение компонент связности, проверка на двудольность и выделение долей, выделение остова графа.

Орграф - граф, ребрам которого присвоено направление.
Взвешенный граф - граф, каждому ребру которого поставлено в соответствие некое значение.
Остов - это дерево, подграф данного графа, с тем же числом вершин, что и у исходного графа.
Связный граф - граф, у которого между любой парой вершин существует как минимум один путь.
Дерево - связный граф из n вершин, у которого n-1 ребро.
Компонента связности - часть графа (подграф), являющаяся связной.
Двудольный граф - граф, в котором все его простые циклы имеют четную длину.

Асимптотика задачи **выделения компонент связности: O(n+m).**
Асимптотика задачи **проверки на двудольность и выделения долей: O(m).**
Асимптотика задачи **выделения остова графа: O(n+m).**

15. Нахождение остова минимального веса. Метод Роберта Прима. Оценки числа шагов.

Асимптотика метода **Роберта Прима O(n^2)**.

16. Алгоритм Дейкстры поиска кратчайшего пути. Оценки числа шагов.

Асимптотика **алгоритма Дейкстры O(n^2).**

17. Нахождение циклов и мостов в графе. Оценки числа шагов.

Цикл графа - путь, в котором начальная и конечная вершины совпадают.
Мост графа - ребро, которое не содержится ни в одном цикле (его удаление увеличивает количество компонент связности).
Петля - ребро, инцидентное одной и той же вершине.
Кратное ребро - это два и более ребер, инцидентных одним и тем же двум вершинам.

Асимптотика задачи **нахождения циклов** в графе **O(n+m)**.
Асимптотика задачи **нахождения мостов** в графе **O(n+m)**.

18. Эйлеров цикл. Оценки числа шагов.

Эйлеров путь - это путь в графе, проходящий через все его рёбра.

Эйлеров цикл - это эйлеров путь, являющийся циклом.

Граф называется **эйлеровым**, если он содержит эйлеров цикл.

Мультиграф - граф, в котором разрешается присутствие кратных рёбер.

Асимптотика задачи **поиска эйлерова пути O(m)** при использовании списков смежности.

19. Гамильтонов цикл. Оценки числа шагов.

Гамильтонов цикл - цикл , который проходит через каждую вершину данного графа ровно по одному разу.
Гамильтонов граф — граф, содержащий гамильтонов цикл.
Асимптотика задачи **нахождения гамильтонова цикла** в графе **O(d^(n)-1)**. Где d это максимальная степень вершины в графе - 1.

20. Алгоритм генерации всех независимых множеств. Оценки числа шагов

Определение. *Множество V' называется вершинным покрытием графа $G = (V, E)$, если всякое ребро графа инцидентно вершине из V'.*

$$\forall uv(\{u, v\} \in E \to (u \in V' \vee v \in V'))$$

Определение. *Множество V' называется независимым множеством графа $G = (V, E)$, если никакие две вершины из V' не смежны.*

$$\forall uv((u \in V' \ \& \ v \in V') \to \{u, v\} \notin E)$$

Определение. *Множество V' называется кликой в графе $G = (V, E)$, если любые две вершины из V' смежны.*

$$\forall uv((u \in V' \ \& \ v \in V') \to \{u, v\} \in E)$$

Теорема 2.2.1. *Следующие утверждения равносильны:*

1. *V' является вершинным покрытием в $G = (V, E)$;*
2. *$V \setminus V'$ является независимым множеством в $G = (V, E)$;*
3. *$V \setminus V'$ является кликой в $\overline{G} = (V \setminus V', \overline{E})$;*

Алгоритм для поиска клик подойдет для поиска НМ. Ищем клики в $\overline{G}$ (дополнение), в G это будут независимые множества

Алгоритм Брона-Кербоша - метод ветвей и границ для поиска всех клик. Вычислительная сложность алгоритма линейна относительно количества клик в графе. В худшем случае алгоритм работает за O($3^{n/3}$) шагов.

21. Теорема о НМ, ВП, КЛИКА.

В прошлом билете.

22. Отличия между интуитивным и математическим понятиями алгоритма. Представление о рекурсивных функциях. Тезис Чёрча.

Алгоритм(интуитивное) - произвольная строго определенная последовательность действий, приводящую к решению той или иной конкретной задачи.
Требования: Определенность данных, Дискретность, Детерминированность, Элементарность шага, Массовость, Направленность.

Алгоритм(математическое) - рекурсивные функции - машины тьюринга.

Тезис Чёрча. Всякая интуитивно вычислимая функция является общерекурсивной.

23. Машины Тьюринга и их модификации. Тезис Тьюринга-Чёрча.

Тезис Тьюринга-Чёрча. Всякая интуитивно вычислимая функция может быть вычислена на машине Тьюринга.

Модификации МТ

Многоленточные. Имеется k лент, на каждой из которых может быть записано свое слово.
Многоголовчатые. Имеется m головок, каждая из которых может обозревать одну ячейку ленты.
Недетерминированные. Если в программе классической машины Тьюринга разрешить использование несогласованных команд.

24. Теорема о числе шагов МТ, моделирующей работу k-ленточной МТ.

Теорема 3.3. *По всякой k-ленточной машине Тьюринга MTk, заканчивающей работу с исходными данными X за t шагов, можно построить одноленточную машину Тьюринга MT1, результат работы которой с исходными данными X совпадает с результатом работы исходной и число шагов которой составляет* $O(t^2)$.

25. Недетерминированные МТ. Теорема о числе шагов МТ, моделирующей работу недетерминированной МТ.

Теорема 3.4. *По всякой недетерминированной машине Тьюринга, проверяющей предикат $\exists Y P(X,Y)$ и заканчивающей работу с исходными данными X за t шагов, можно построить одноленточную машину Тьюринга, результат работы которой с исходными данными X совпадает с результатом работы исходной и число шагов которой составляет $2^{O(t)}$.*

26. Понятия сложности алгоритма от данных, сложность алгоритма, сложность задачи. Верхняя и нижняя оценки сложности.

Под вычислительной сложностью алгоритма понимают функцию, зависящую от ДЛИНЫ записи исходных данных и характеризующую

- *число шагов работы алгоритма над исходными данными* (временна́я сложность);
- *объём памяти, необходимой для работы алгоритма над исходными данными* (ёмкостная или зональная сложность).

Определение. *Сложностью $S_A(P)$ алгоритма A при работе над данными P называется число шагов или объём памяти, затраченные в процессе работы алгоритма A над данными P.*

Определение. *Верхней (нижней) оценкой сложности алгоритма A при работе над данными длины n называется*

$$S_A^U(n) = \max_{P:||P||=n} \{S_A(P)\}$$

(соответственно

$$S_A^L(n) = \min_{P:||P||=n} \{S_A(P)\}).$$

27. Соотношение между временем работы алгоритма и требуемой памятью.

Требуемая память это функция от длины записи числа, как и вычислительная сложность. Не всегда под временем работы алгоритма понимается именно вычислительная сложность, иногда это число шагов, и эти шаги могут быть достаточно сложными. Шаг может зависеть от самого числа, а не от его длины. В таких случаях время работы алгоритма может быть меньше, чем требуемая память, которая зависит от длины записи самого числа. Как пример на парах был рассмотрен 2^n.

28. Классы алгоритмов и задач. Схема обозначений.

Задача принадлежит классу сложности C, если существует алгоритм из класса C, решающий эту задачу.

Буква **F** в начале класса используется для обозначения класса функций, если её нет, то это класс предикатов.

Буквы **D** и **N** обозначают, что в определении класса сложности использована детерминированная или соответственно недетерминированная машина Тьюринга. D обычно не пишется(типа дефолт).

Функция сложности - это функция от длины записи исходных данных, ограничивающая число шагов или число шагов соответствующей машины Тьюринга.

29. Классы P, NP, и P-SPACE. Соотношения между этими классами.

Класс P - это класс предикатов, для которых существует алгоритм, который может быть реализован на детерминированной машине Тьюринга, число шагов которого не превосходит полинома от длины записи исходных данных.

Класс NP - это класс предикатов, для которых существует алгоритм, который может быть реализован на недетерминированной машине Тьюринга, число шагов которого не превосходит полинома от длины записи исходных данных.

Класс P-SPACE - это класс предикатов, для которых существует алгоритм, который может быть реализован на детерминированной машине Тьюринга, число использованных ячеек которой не превосходит полинома от длины записи исходных данных.

Соотношение между классами сложности: $\mathbf{P} \subseteq \mathbf{NP} \subseteq \mathbf{P\text{-}SPACE} \subset \mathbf{EXP}.$

Теорема 4.1. *Если предикат вида* $\exists Y\ P(X,Y)$ *принадлежит классу* **NP**, *то существует проверяющая его одноленточная машина Тьюринга, число шагов которой составляет* $2^{p(n)}$, *где* $p(n)$ *– полином от длины записи исходных данных* $n = \|X\|$.

30. Полиномиальная сводимость и полиномиальная эквивалентность.

Определение. *Задача* Z_1 *вида* $\exists Y\ P_1(X,Y)$ *при* $X \in D_1$ *полиномиально сводится к задаче* Z_2 *вида* $\exists Y\ P_2(X,Y)$ *при* $X \in D_2$

$$Z_1 \propto Z_2,$$

если существует функция f, отображающая D1 в D2 и такая, что

1) Существует машина Тьюринга, вычисляющая функцию f не более чем за полиномиальное от длины записи исходных данных число шагов
2) Задача Z1 имеет решение с исходными данными X тогда и только тогда, когда задача Z2 имеет решение с исходными данными f(X)

$$\forall X_{\in D_1}(\exists Y\ P_1(X,Y) \leftrightarrow \exists Y\ P_2(f(X),Y)).$$

Лемма 4.1. *Отношение полиномиальной сводимости рефлексивно* $\forall Z(Z \propto Z)$ *и транзитивно* $\forall Z_1Z_2Z_3(Z_1 \propto Z_2\ \&\ Z_2 \propto Z_3 \to Z_1 \propto Z_3)$.

Лемма 4.2. *Если* $Z_1 \propto Z_2$ *и* $Z_2 \in \mathbf{P}$, *то* $Z_1 \in \mathbf{P}$.

31. Полиномиальная сводимость задачи ГЦ к задаче КОММИВОЯЖЁР.

Определение. *Задача* Z_1 *полиномиально эквивалентна задаче* Z_2 $(Z_1 \sim_p Z_2)$, *если* $Z_1 \propto Z_2$ *и* $Z_2 \propto Z_1$.

Теорема 4.2. *Отношение полиномиальной эквивалентности является отношением эквивалентности, то есть оно*

— *рефлексивно* $\forall Z(Z \sim_p Z)$,

— *симметрично* $\forall Z_1 Z_2(Z_1 \sim_p Z_2 \to Z_2 \sim_p Z_1)$ *и*

— *транзитивно* $\forall Z_1 Z_2 Z_3(Z_1 \sim_p Z_2 \,\&\, Z_2 \sim_p Z_3 \to Z_1 \sim_p Z_3)$.

Доказательство очевидно.

Отношение $\sim_p$ разбивает класс **NP** на классы эквивалентности. Один из них – класс **P** – самые «быстро решаемые» задачи из класса **NP**.

32. Классы эквивалентности по отношению полиномиальной эквивалентности. Класс P - пример такого класса.

Определение. *Задача* Z_1 *полиномиально эквивалентна задаче* Z_2 $(Z_1 \sim_p Z_2)$, *если* $Z_1 \propto Z_2$ *и* $Z_2 \propto Z_1$.

Теорема 4.2. *Отношение полиномиальной эквивалентности является отношением эквивалентности, то есть оно*

— *рефлексивно* $\forall Z(Z \sim_p Z)$,

— *симметрично* $\forall Z_1 Z_2(Z_1 \sim_p Z_2 \to Z_2 \sim_p Z_1)$ *и*

— *транзитивно* $\forall Z_1 Z_2 Z_3(Z_1 \sim_p Z_2 \,\&\, Z_2 \sim_p Z_3 \to Z_1 \sim_p Z_3)$.

Доказательство очевидно.

Отношение $\sim_p$ разбивает класс **NP** на классы эквивалентности. Один из них – класс **P** – самые «быстро решаемые» задачи из класса **NP**.

Задачи, принадлежащие классу P

Примерами задач из класса P являются целочисленное сложение, умножение, деление, взятие остатка от деления, умножения матриц, выяснение связности графов, сортировка множества из n чисел, нахождение эйлерова цикла на графе из m рёбер, обнаружение в тексте длиной n некоторого слова, построение покрывающего дерева минимальной стоимости, линейное программирование и некоторые другие.

33. NP-полные задачи. Класс NP-полных задач - класс эквивалентности по отношению полиномиальной эквивалентности.

Задача Z называется NP-полной, если она принадлежит классу NP и любая задача из класса NP полиномиально сводится к ней.

Теорема: Класс NP-полных задач образует класс эквивалентности по отношению полиномиальной эквивалентности в классе NP.

34. Задача ВЫПОЛНИМОСТЬ (ВЫП). Теорема Кука.

Определение:
Выполнимость(ВЫП)
Дано:
U = {u1,..., un} - множество произвольных переменных
C = {c1,..., cm} - множество предложений над U
Вопрос: выполнимо ли множество C, т.е. существует ли набор значений для переменных из U, для которого истинны все предложения из C?

$$\exists u_1, ..., u_n (c_1 \ \& \ ... \ \& \ c_m).$$

Теорема Кука: Задача ВЫП NP-полна.

35. Задача 3-ВЫПОЛНИМОСТЬ (3-ВЫП). Её NP-полнота.

3-ВЫПОЛНИМОСТЬ (3-ВЫП)
ДАНО:
U - множество пропозициональных переменных,
C - множество предложений над U с тремя переменными каждое.

ВОПРОС:
Существует ли набор значений для переменных, выполняющий множество C?
∃ u1, ..., un(c1 & ... & cm)?
Теорема: Если для задачи Z и NP-полной задачи Z1 выполнены условия:

- Z ∈ NP
- Z1 полиномиально сводится к Z
  то Z - NP-полная

Теорема. Задача 3-ВЫП NP-полна.

36. Задачи ВЕРШИННОЕ ПОКРЫТИЕ (ВП), НЕЗАВИСИМОЕ МНОЖЕСТВО (НМ), КЛИКА. NP-полнота задачи ВП. Полиномиальная эквивалентность этих трёх задач.

Определение задач

ВЕРШИННОЕ ПОКРЫТИЕ (ВП)

ДАНО: Граф $G = (V, E)$,
$K \in Z_+$, $K \leq |V|$.

ВОПРОС: Имеется ли в G вершинное покрытие не более чем из K элементов?

$$\exists V'(V' \subseteq V \ \& \ |V'| \leq K \ \& \ \forall uv(\{u, v\} \in E \to (u \in V' \vee v \in V')))$$

НЕЗАВИСИМОЕ МНОЖЕСТВО (НМ)

ДАНО: Граф $G = (V, E)$,
$J \in Z_+$, $J \geq |V|$.

ВОПРОС: Имеется ли в G независимое множество не менее чем из J элементов?

$$\exists V'(V' \subseteq V \ \& \ |V'| \geq J \ \& \ \forall uv((u \in V' \ \& \ \ \in V') \to \{u, v\} \notin E))$$

КЛИКА

ДАНО: Граф $G = (V, E)$,
$J \in Z_+$, $J \leq |V|$.

ВОПРОС: Содержит ли G клику не менее чем из J вершин?

$$\exists V'(V' \subseteq V \ \& \ |V'| \geq K \ \& \ \forall uv(u \in V' \ \& \ v \in V' \to \{u, v\} \in E))$$

Теорема. Задача вершинного покрытия NP-полна.

Полиномиальная эквивалентность этих трёх задач

Теорема 2.2.1. *Следующие утверждения равносильны:*

1. V' *является вершинным покрытием в* $G = (V, E)$;
2. $V \setminus V'$ *является независимым множеством в* $G = (V, E)$;
3. $V \setminus V'$ *является кликой в* $\overline{G} = (V \setminus V', \overline{E})$;

Для доказательства запишем формулы, определяющие соответствующие понятия, для каждого из утверждений теоремы.

1. $\forall uv(\{u, v\} \in E \to (u \in V' \vee v \in V'))$.
2. $\forall uv((u \in V \setminus V' \;\&\; v \in V \setminus V') \to \{u, v\} \notin E) \Leftrightarrow$
   $\forall uv(\{u, v\} \in E \to \neg(u \in V \setminus V' \;\&\; v \in V \setminus V')) \Leftrightarrow$
   $\forall uv(\{u, v\} \in E \to (u \notin V \setminus V' \vee v \notin V \setminus V')) \Leftrightarrow$
   $\forall uv(\{u, v\} \in E \to (u \in V' \vee v \in V'))$.
3. $\forall uv((u \in V \setminus V' \;\&\; v \in V \setminus V') \to \{u, v\} \in \overline{E}) \Leftrightarrow$
   $\forall uv(\{u, v\} \notin \overline{E} \to \neg(u \in V \setminus V' \;\&\; v \in V \setminus V')) \Leftrightarrow$
   $\forall uv(\{u, v\} \in E \to (u \notin V \setminus V' \vee v \notin V \setminus V')) \Leftrightarrow$
   $\forall uv(\{u, v\} \in E \to (u \in V' \vee v \in V'))$. ■

37. NP-полнота задач ГЦ и ГП (без доказательства).

ГАМИЛЬТОНОВ ЦИКЛ
ДАНО: Граф $G = (V, E)$.
ВОПРОС: Содержит ли G гамильтонов цикл?

$$\exists(v_{i_1}, ..., v_{i_n})(\{v_{i_1}, ..., v_{i_n}\} = V \;\&\; \{v_{i_1}, v_{i_2}\} \in E \;\&\; , ..., \;\&\; \{v_{i_n}, v_{i_1}\} \in E)$$

- ВП полиномиально сводится к ГЦ
- ГЦ $\in NP$
  $\Rightarrow$ ГЦ - NP-полная

А ГП хз

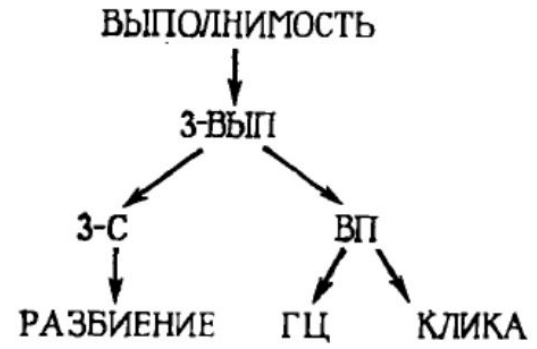

Рис. 3.1. Диаграмма последовательности сведения задач, используемых для доказательства NP-полноты шести основных задач.

38. NP-полнота задач 3-С и РАЗБИЕНИЕ (без доказательства).

ТРЕХМЕРНОЕ СОЧЕТАНИЕ (3-С)
УСЛОВИЕ. Дано множество $M \subseteq W \times X \times Y$, где W, X и Y — непересекающиеся множества, содержащие одинаковое число элементов q.
ВОПРОС. Верно ли, что M содержит трехмерное *сочетание*, т. е. подмножество $M' \subseteq M$, такое, что $|M'| = q$ и никакие два разных элемента M' не имеют ни одной равной координаты?

РАЗБИЕНИЕ
ДАНО: Конечное множество A,
для каждого $a \in A$ его «вес» $s(a) \in Z_+$.
ВОПРОС: Существует ли разбиение множества A на два подмножества одинакового веса.

$$\exists A' \left(A' \subseteq A \;\&\; \sum_{a \in A'} s(a) = \sum_{a \in (A \setminus A')} s(a) \right)$$

У последней задачи может быть и другая формулировка.

РЕШЕНИЕ ЛИНЕЙНОГО ОДНОРОДНОГО УРАВНЕНИЯ В ЧИСЛАХ ИЗ $\{-1, 1\}$
ДАНО: $\{s_1, \ldots, s_n\}$ при $s_i \in Z_+$ $(i = 1, \ldots, n)$.
ВОПРОС: Существует ли решение линейного однородного уравнения с коэффициентами $\{s_1, \ldots, s_n\}$ в числах из $\{-1, 1\}$?

$$\exists x_1 \ldots x_n (\&_{i=1}^{n} (x_i \in \{-1, 1\}) \;\&\; s_1 x_1 + \ldots + s_n x_n = 0)$$

- 3-ВЫП полиномиально сводится к 3-С

- 3-C $\in NP$
  ⇒ 3-C - NP-полная
- 3-C полиномиально сводится к РАЗБИЕНИЕ
- РАЗБИЕНИЕ $\in NP$
  ⇒ РАЗБИЕНИЕ - NP-полная

39. Метод сужения доказательства NP-полноты.

Определение. *Задача Z_1 является сужением на множество D_1 задачи Z_2 с исходными данными из множества D_2, если $D_1 \subseteq D_2$ и* $\forall X(X \in D_1 \to (Z_1 \leftrightarrow Z_2))$.

Если задача принадлежит классу **NP**, а её подзадача NP-полна, то и исходная задача NP-полна, т.к. подзадача полиномиально сводится к исходной задаче с помощью тождественного отображения.

40. «Похожие» задачи и их сложность.

P	NP-полные
КРАТЧАЙШИЙ ПУТЬ МЕЖДУ МЕЖДУ ДВУМЯ ВЕРШИНАМИ	ДЛИННЕЙШИЙ ПУТЬ МЕЖДУ МЕЖДУ ДВУМЯ ВЕРШИНАМИ
Дано: Граф $G = (V, E)$, «длины» $l(e) \in Z_+$ $(e \in E)$, «длины» $l(e) \in Z_+$ $(e \in E)$, выделенные вершины $s,\ f \in V$, число $B \in Z_+$. **Вопрос:** Существует ли в G, простой путь из s в f длины	
$\leq B$	$\geq B$

Для первой задачи мы знаем алгоритм Дейкстры. Вычисляем кратчайший путь, если он больше B, то решения нет. Если <= B, то решение есть.

РЁБЕРНОЕ ПОКРЫТИЕ	ВЕРШИННОЕ ПОКРЫТИЕ
Дано: Граф $G = (V, E)$, число $K \in Z_+$. **Вопрос:** Существует ли в подмножество	
$E' \subseteq E,\ \|\|E'\|\| \leq K$ $\forall v_{\in V} \exists e_{\in E'}$	$V' \subseteq V,\ \|\|V'\|\| \leq K$ $\forall e_{\in E} \exists v_{\in V'}$
$v \in e$	

Реберное покрытие - существует полиномиальный алгоритм (Косовская его не знает, видимо и мы не должны)
Вершинное покрытие - одна из основных NP-полных задач.

ТРАНЗИТИВНАЯ РЕДУКЦИЯ	МИНИМАЛЬНЫЙ ЭКВИВАЛЕНТНЫЙ ОРГРАФ
Дано: Орграф $G = (V, A)$, число $K \in Z_+$. **Вопрос:** Существует ли подмножество $A' \subseteq V \times V$ | $A' \subseteq A$ такое что $\|\|A'\|\| \leq K$ и для всех $u, v \in V$ граф $G' = (V, A')$ содержит путь из u в v тогда и только тогда граф G содержит такой путь.	

“построить заново легче, чем отремонтировать”
В первой задаче мы просто забываем, что у нас были какие-то ребра, и можем прокладывать любые.
Во второй задаче из старых ребер выбираем подмножество.

РАСПИСАНИЕ ДЛЯ ПРЯМОГО ДЕРЕВА ЗАДАНИЙ	РАСПИСАНИЕ ДЛЯ ОБРАТНОГО ДЕРЕВА ЗАДАНИЙ
Дано: Множество T заданий с единичной длительностью, директивный срок $d(t) \in Z_+$ $(t \in T)$, число $m \in Z_+$ частичный порядок $< \cdot$ на T, относительно которого каждое задание имеет не более одного непосредственно следующего за ним | предшествующего ему **Вопрос:** Можно ли составить для множества T расписание на m процессорах, на каждом из которых задания выполняются согласно заданному частичному порядку, так что все директивные сроки выполнены.	

41. Анализ подзадач.

Билеты Наташи:

> **Подзадачей** данной задачи называется задача, полученная из исходной наложением некоторых ограничений на ее исходные данные.

Методичка:

> **Определение.** *Задача Z_1 с исходными данными из множества D_1 является подзадачей задачи Z_2 с исходными данными из множества D_2 $(Z_1 \subseteq Z_2)$, если $D_1 \subseteq D_2$ и $\forall X(X \in D_1 \rightarrow (Z_1 \leftrightarrow Z_2))$.*

Вывод. Прежде, чем отказываться от программирования для многократного использования (для исходных данных большого размера) алгоритма, решающего NP-полную задачу, проверьте, не поставлена ли перед Вами ее подзадача, имеющая полиномиальный алгоритм.

Билеты Наташи:
Анализ подзадач NP-полной задачи имеет важное практическое значение при ответе на вопрос, будет решающая ее программа работать приемлемое время или нет.

42. Алгоритм решения задачи РАЗБИЕНИЕ.

Эта задача псевдополиномиальная.

РАЗБИЕНИЕ
ДАНО: Конечное множество A,
для каждого $a \in A$ его «вес» $s(a) \in Z_+$.
ВОПРОС: Существует ли разбиение множества A на два подмножества одинакового веса.

$$\exists A' \left(A' \subseteq A \;\&\; \sum_{a \in A'} s(a) = \sum_{a \in (A \setminus A')} s(a) \right)$$

Алгоритм

1. Вычислим $s_1 + ... + s_n$. Если число нечётное, то задача решения не имеет. В противном случае определим $B = \frac{1}{2}(s_1 + ... + s_n)$.
2. Определим таблицу с элементами t_{ij} при $i = 1, ..., n$, $j = 0, 1, ..., B$. $t_{ij} \Leftrightarrow$ «в множестве $\{s_1, \ldots, s_i\}$ есть подмножество веса j».
3. Заполним таблицу, используя свойства t_{ij}:
   $t_{i0} = T$ для всех i,
   если $t_{ij} = T$, то $t_{(i+1)j} = T$,
   если $t_{ij} = T$, то $t_{(i+1)(j+s_{i+1})} = T$.
4. Если в столбце с номером B появилось значение T, то задача имеет решение. Если же после заполнения таблицы ни в одной строке в последнем столбце нет значения T, то задача решения не имеет.

43. Задачи с числовыми параметрами. Псевдополиномиальные задачи.

Билеты Наташи:

> Если временная сложность решения задачи с числовыми входными данными не превосходит полинома от этих чисел (а не их длины) и длины записи остальных исходных данных, то такая задача называется **псевдополиномиальной.**

Методичка:

Определение. *Задача с числовыми параметрами называется псевдополиномиальной, если число шагов решающей её машины Тьюринга не превосходит полинома от этих числовых параметров и длины записи остальных исходных данных.*