



Московский институт электроники и  
математики им. А. Н. Тихонова

Проект 1476

Москва  
2025

# Алгоритм PEG: предназначение и принцип работы

Подготовил Нугманов Мухамед Ильгисович



## План доклада

- 1) Основы – граф Таннера и его роль в декодировании
- 2) Обхват графа и проблема коротких циклов
- 3) Решение задачи – алгоритм PEG
- 4) Пошаговое описание алгоритма PEG

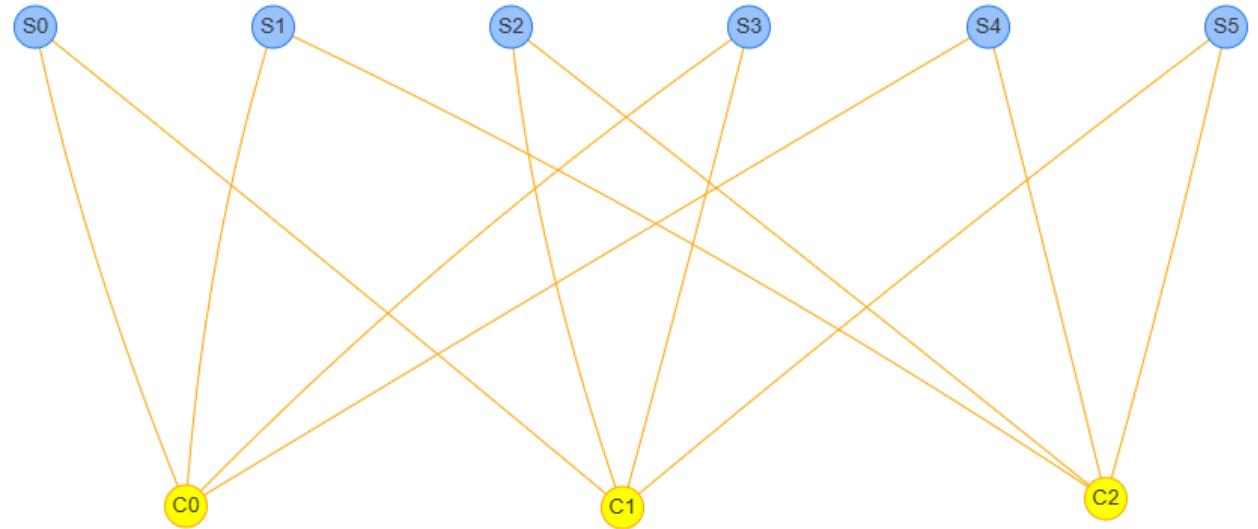
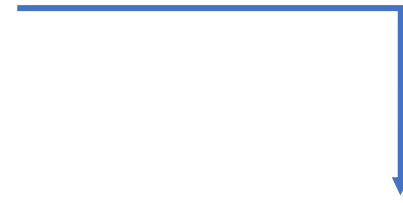


Московский институт электроники и  
математики им. А. Н. Тихонова

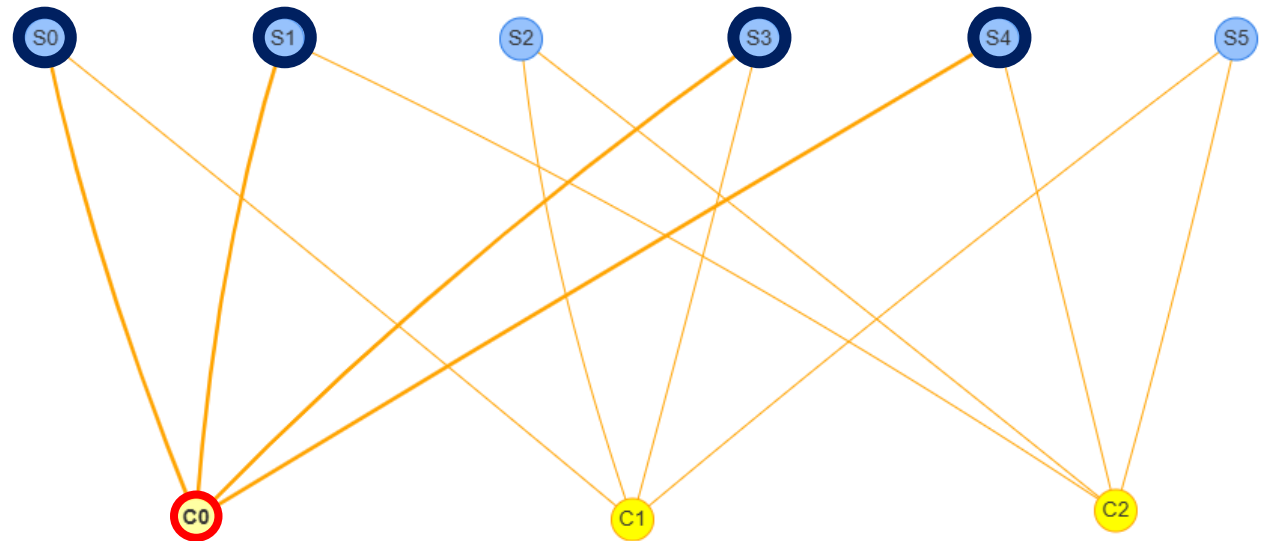
# Основы – граф Таннера и его роль в декодировании

$$H = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 \end{bmatrix}$$

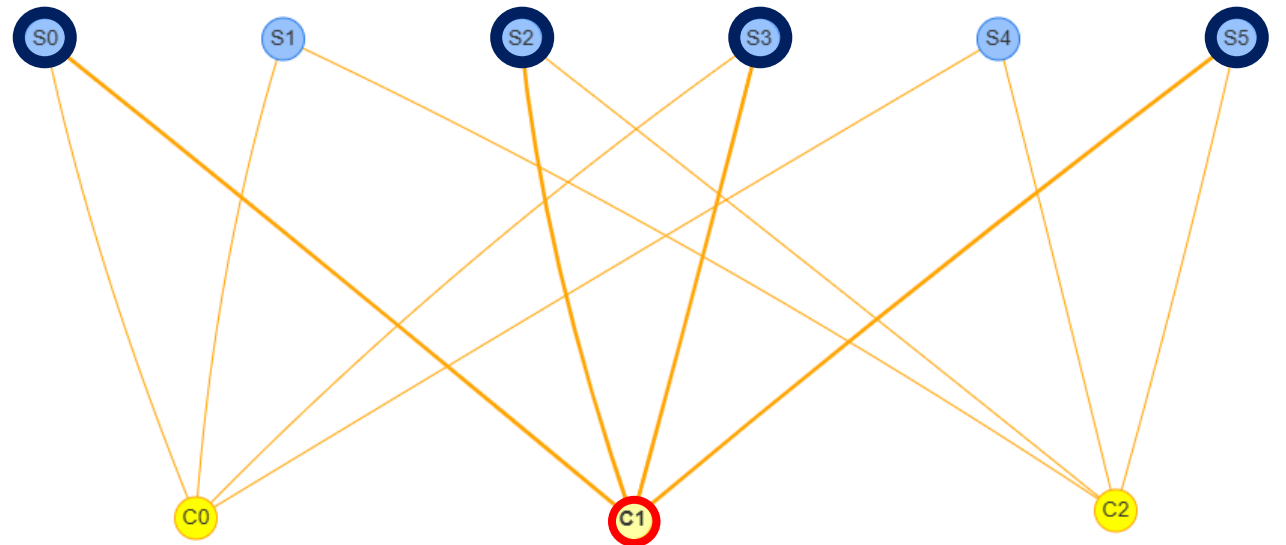
$$H = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 \end{bmatrix}$$



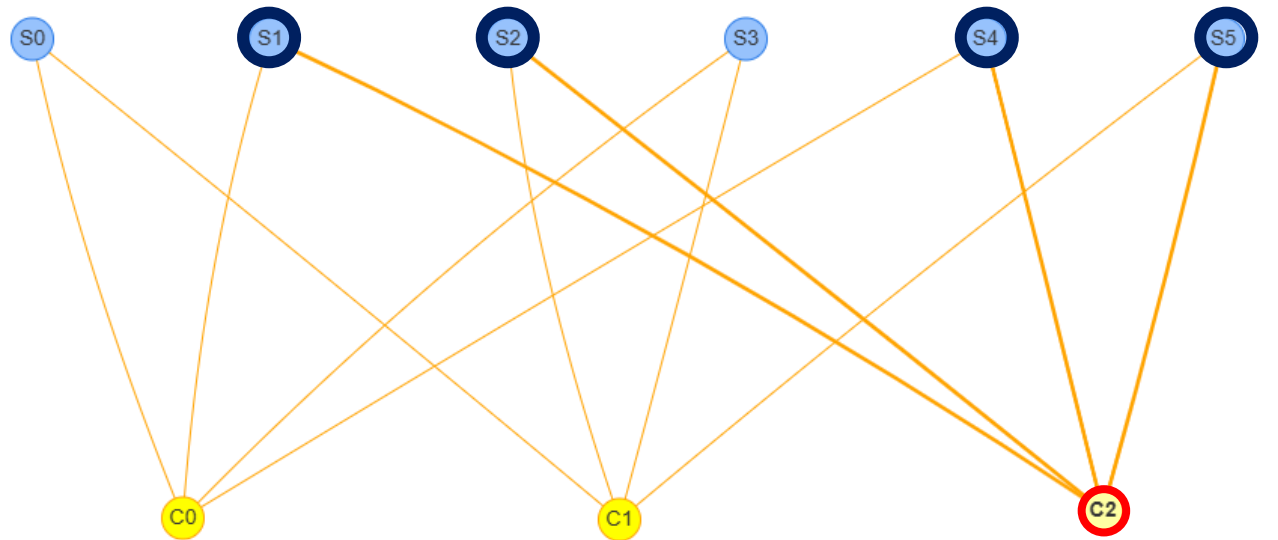
$$H = \begin{bmatrix} \textcircled{1} & \textcircled{1} & 0 & \textcircled{1} & \textcircled{1} & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 \end{bmatrix}$$

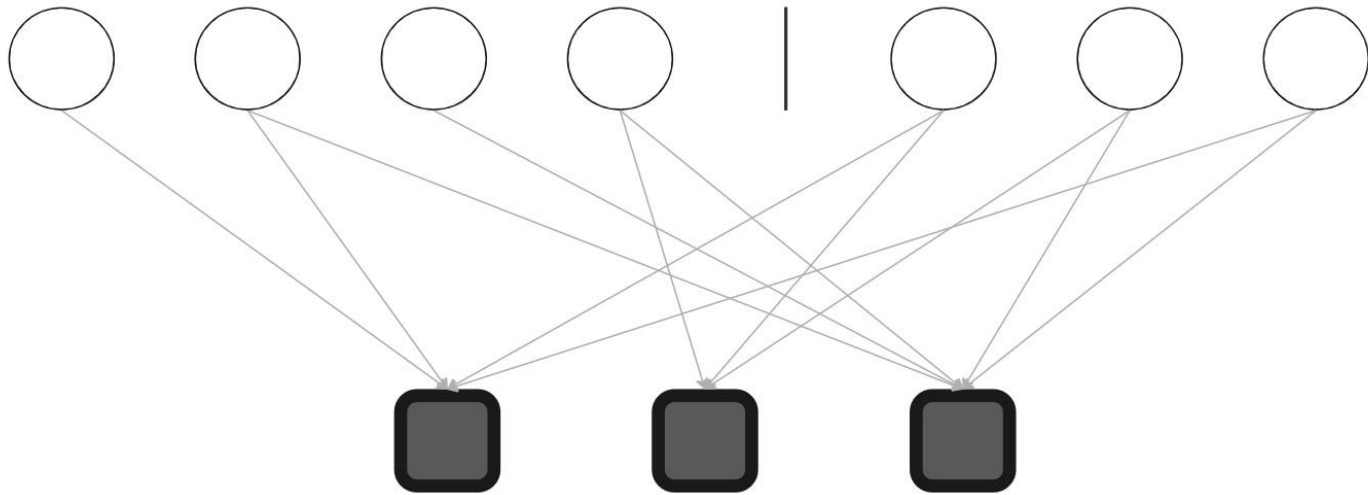


$$H = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 \\ \textcircled{1} & 0 & \textcircled{1} & \textcircled{1} & 0 & \textcircled{1} \\ 0 & 1 & 1 & 0 & 1 & 1 \end{bmatrix}$$



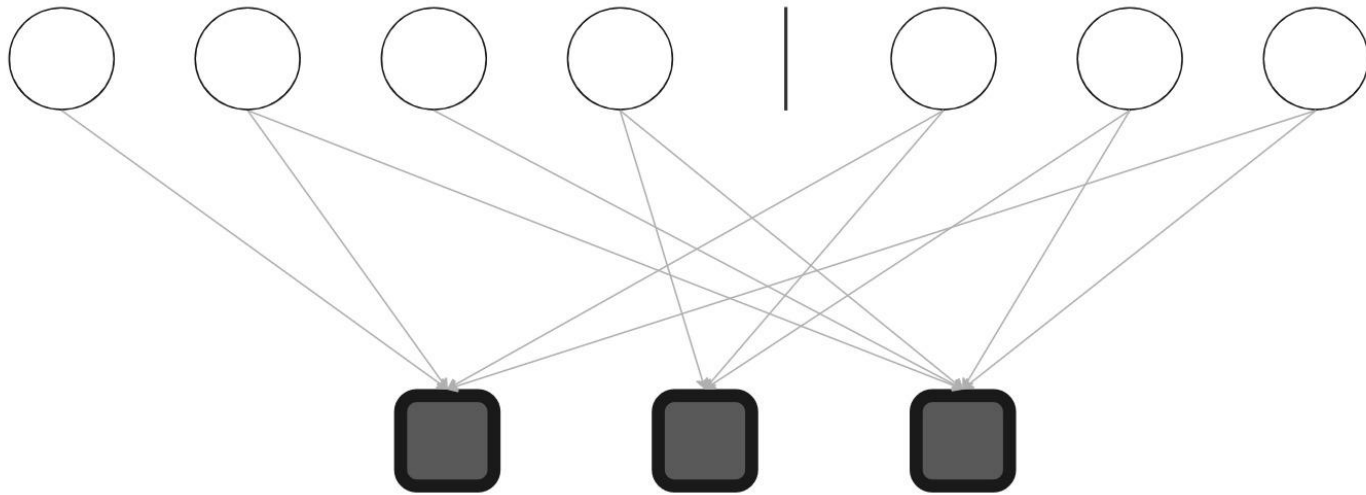
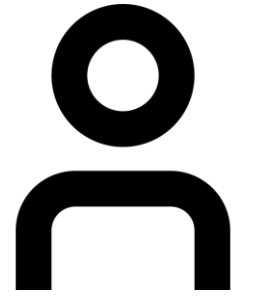
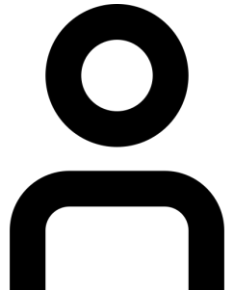
$$H = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & \textcircled{1} & \textcircled{1} & 0 & \textcircled{1} & \textcircled{1} \end{bmatrix}$$

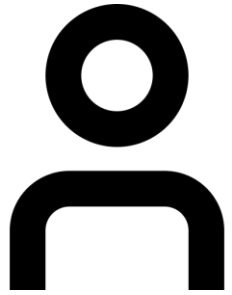




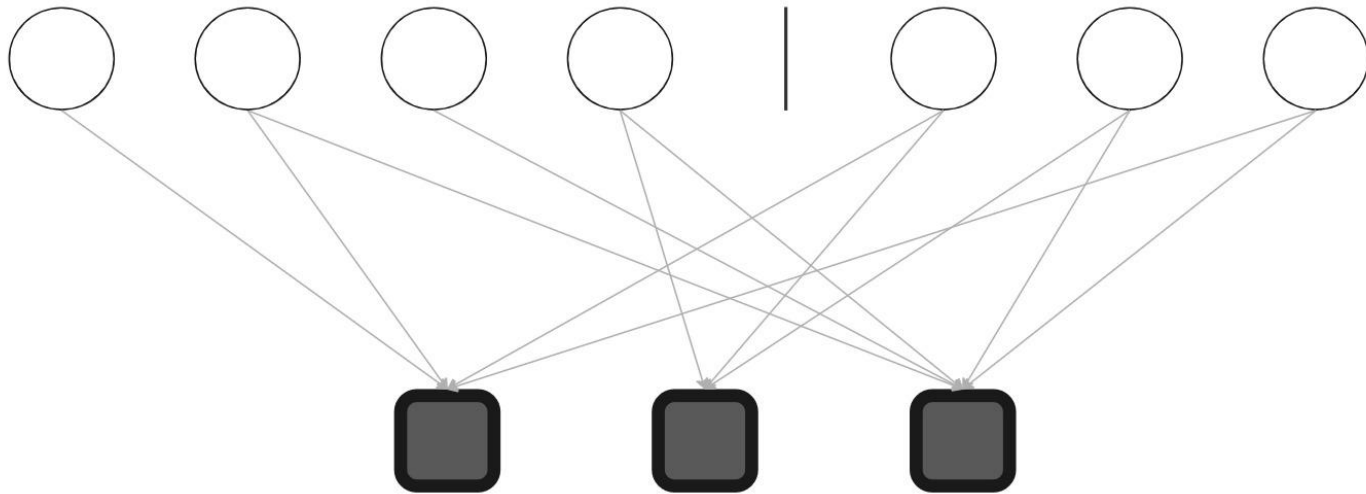
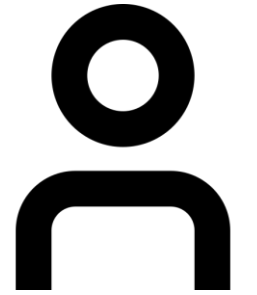


10



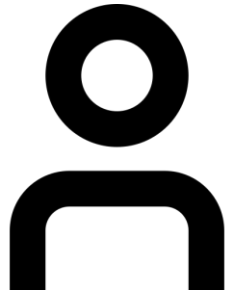


1010 001

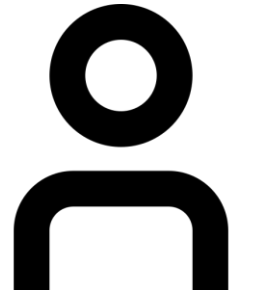




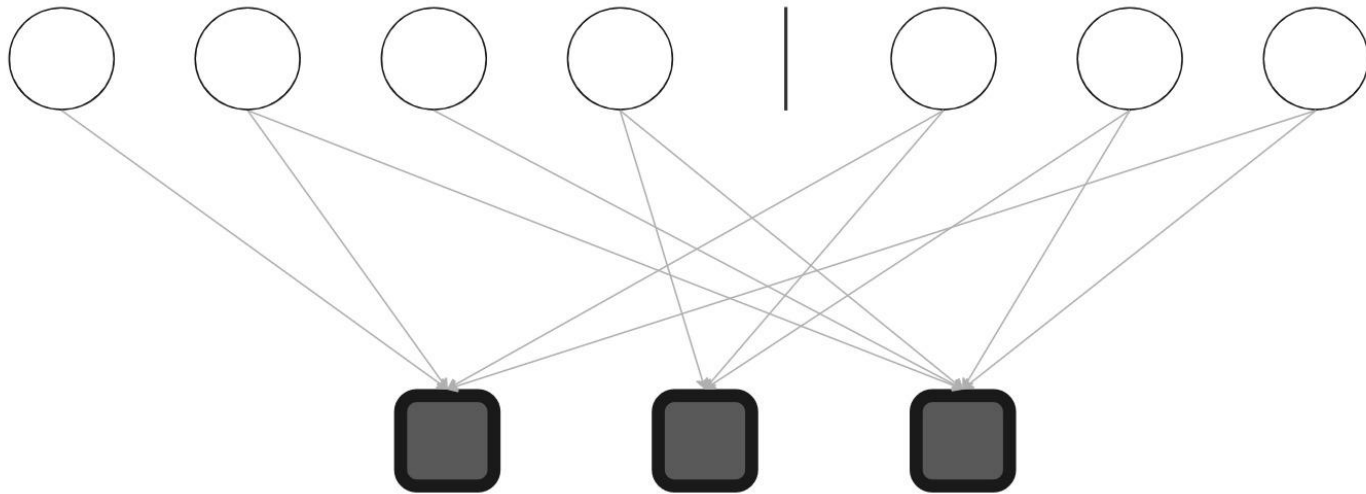
12



1010 001

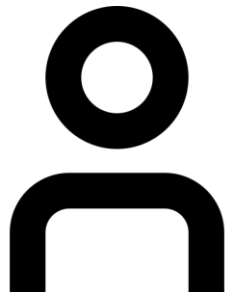


10?0 ??1

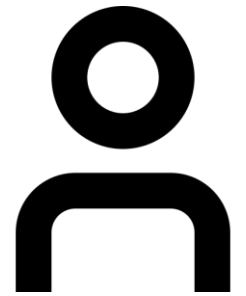




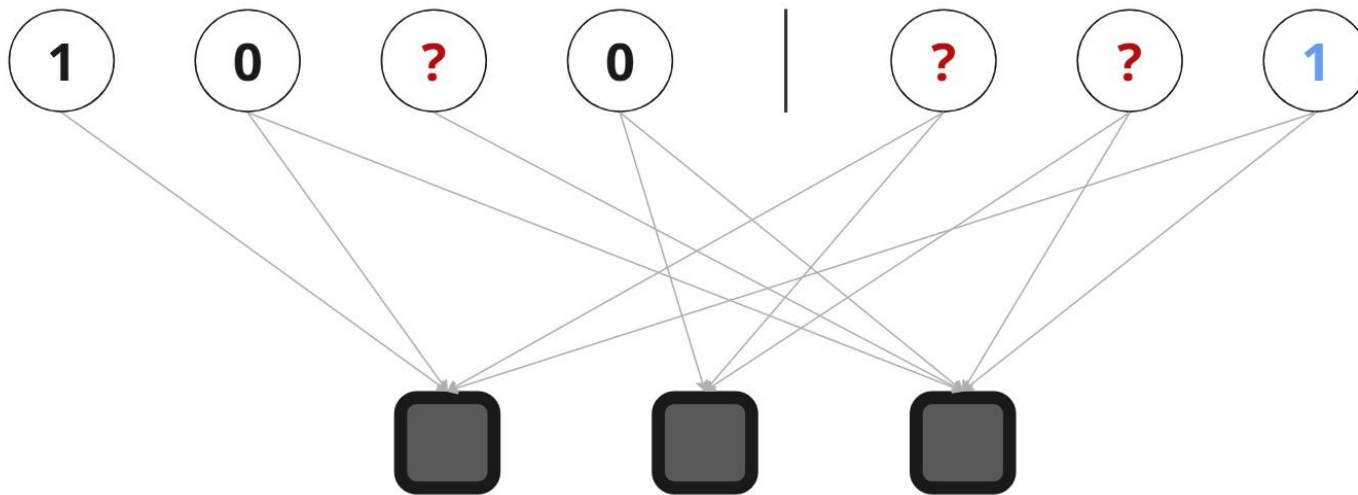
13



1010 001

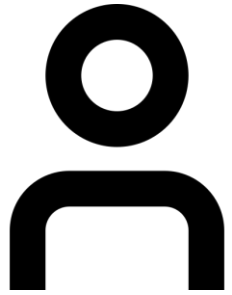


10?0 ??1

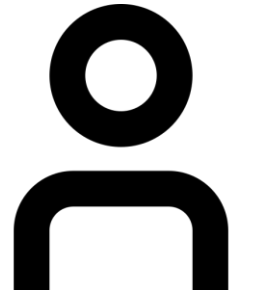




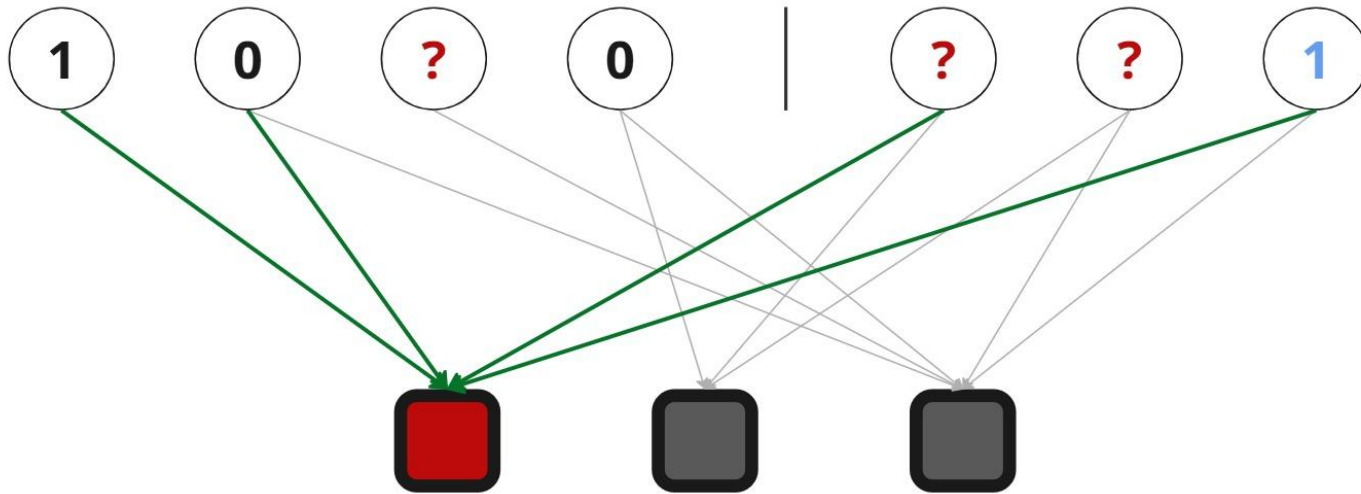
14



1010 001

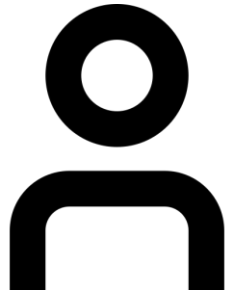


10?0 ??1

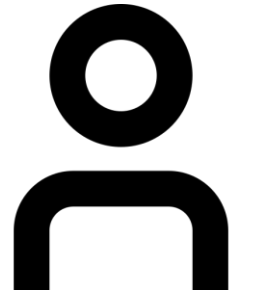




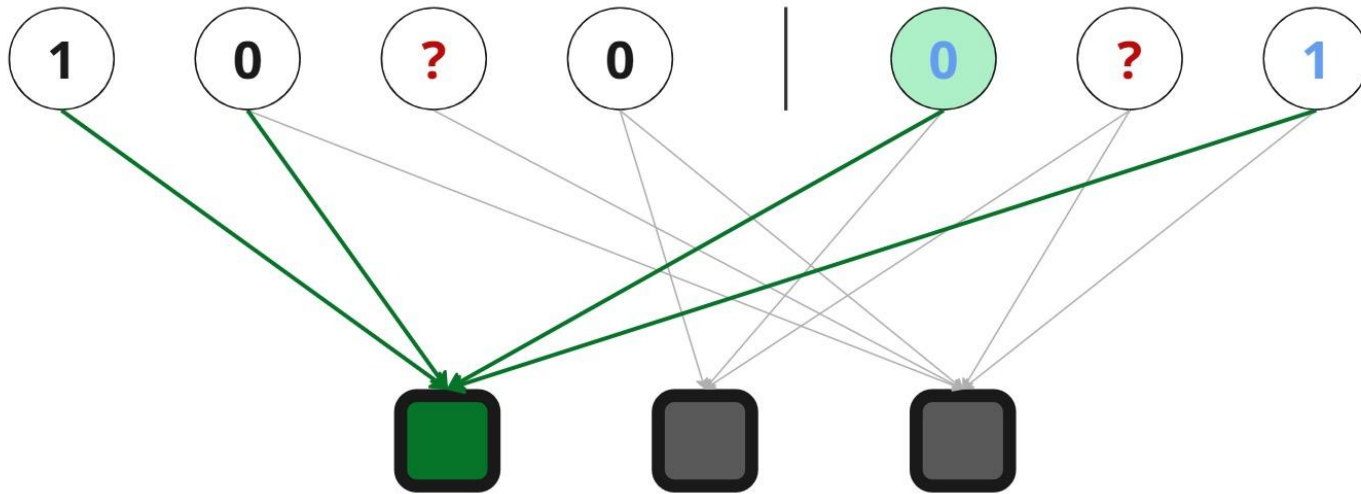
15



1010 001

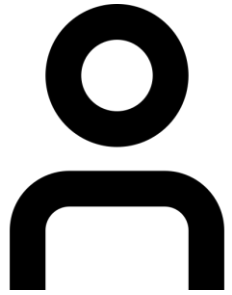


10?0 0?1

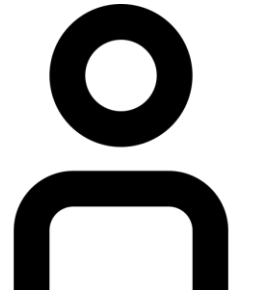




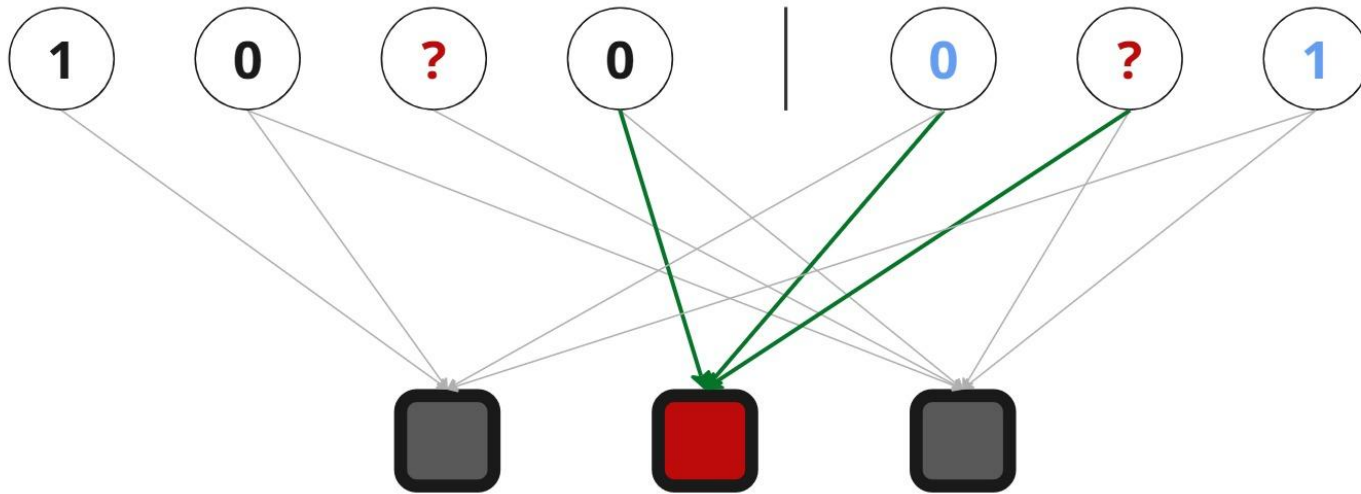
16



1010 001

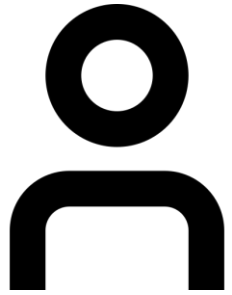


10?0 0?1

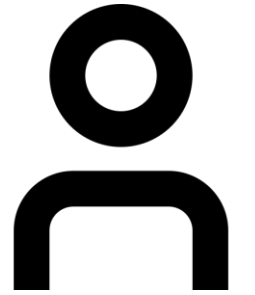




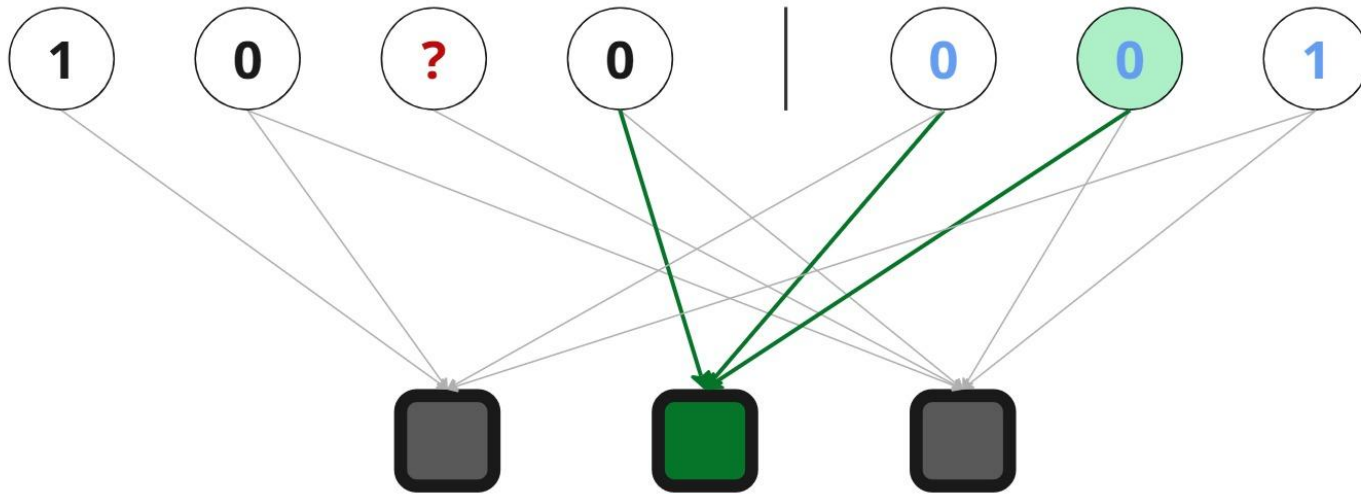
17



1010 001

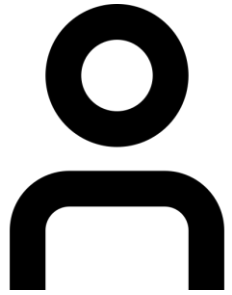


10?0 001

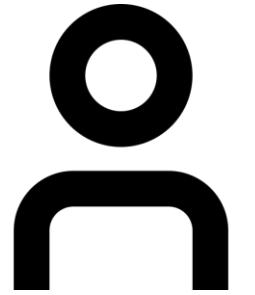




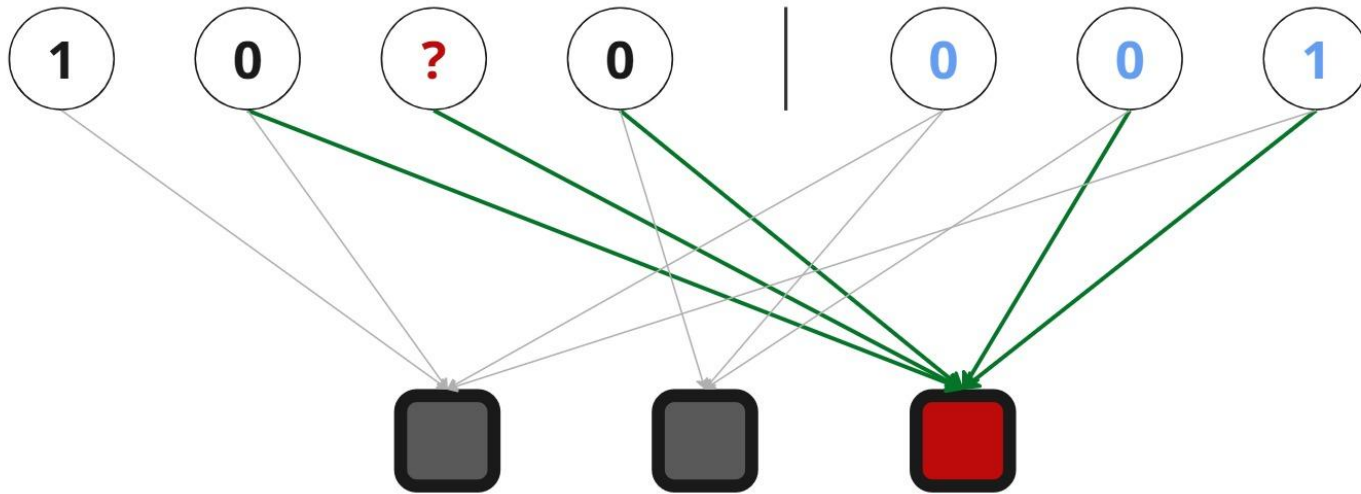
18



1010 001

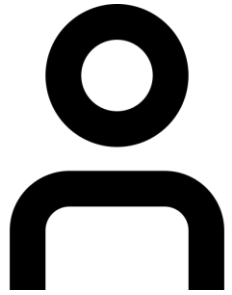


10?0 001

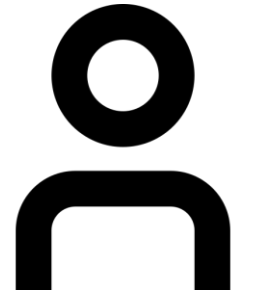




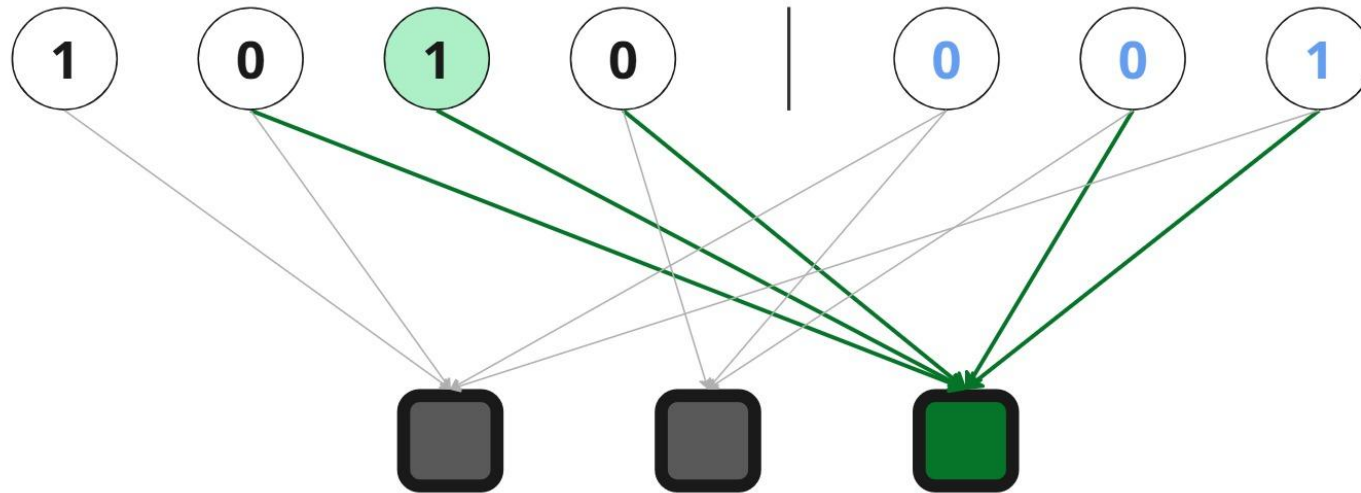
19



1010 001



1010 001





Таким образом, граф Таннера естественным образом описывает информационный поток в алгоритме декодирования.

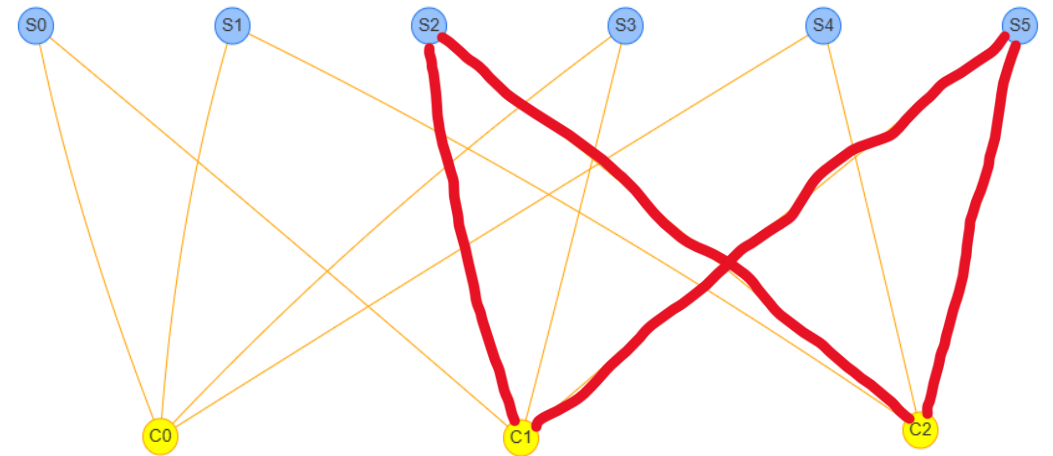


Московский институт электроники и  
математики им. А. Н. Тихонова

# Обхват графа и проблема коротких циклов

Обхват (girth) графа Таннера — это длина самого короткого цикла в этом графе, измеряемая в количестве рёбер.

Пример: Цикл вида  $c_1 - s_2 - c_2 - s_5 - c_1$  имеет длину 4 (четыре ребра).





Московский институт электроники и  
математики им. А. Н. Тихонова

# Решение задачи – алгоритм PEG

## Проблема оптимального решения

Задача построения графа Таннера с максимально возможным обхватом при заданных параметрах (количество узлов, степени узлов) является NP-трудной комбинаторной задачей.

Перебор всех возможных конфигураций рёбер требует проверки экспоненциального числа вариантов и практически невозможен даже для небольших графов.

## Подход алгоритма PEG

Вместо поиска абсолютно оптимального решения, алгоритм PEG использует жадный подход (greedy algorithm):

- Строит граф поэтапно, по одному ребру за раз
- На каждом шаге выбирает добавление ребра, которое минимально ухудшает локальный обхват
- Не может отступить назад — если выбор оказался неудачным, вернуться к предыдущему состоянию невозможно

**Результат:** Хотя это не гарантирует абсолютный максимум, алгоритм обычно дает относительно большой обхват практически за линейное время, что является хорошим компромиссом между качеством и вычислительной сложностью.

## Определения

**степень символьного узла** - это количество ребер, соединенных с символьным узлом, т.е. Количество единиц в столбце матрицы проверки четности, соответствующей этому символьному узлу

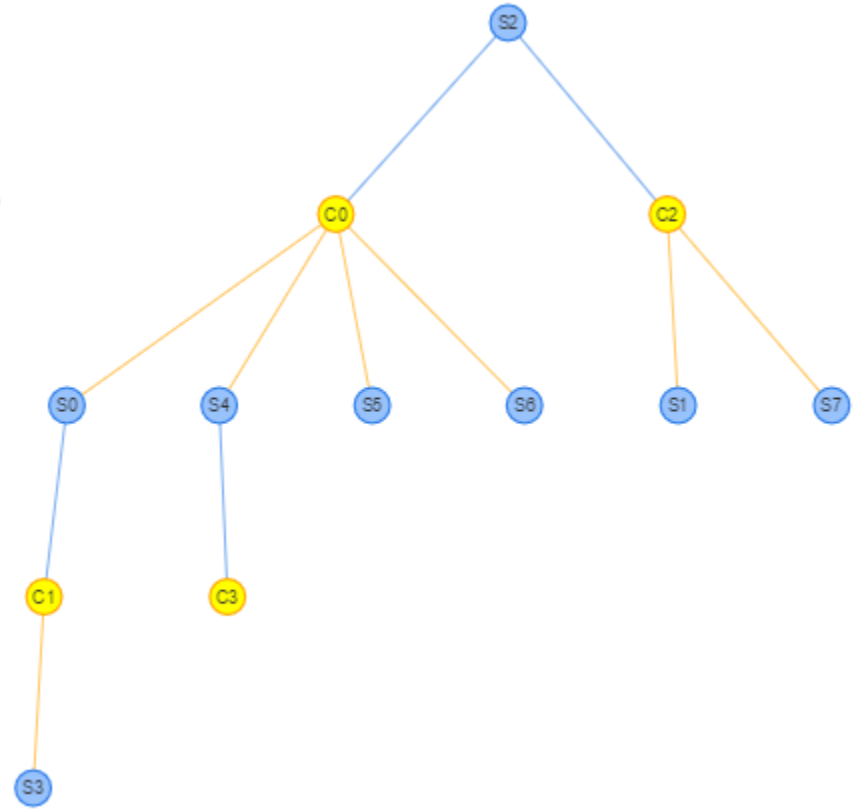
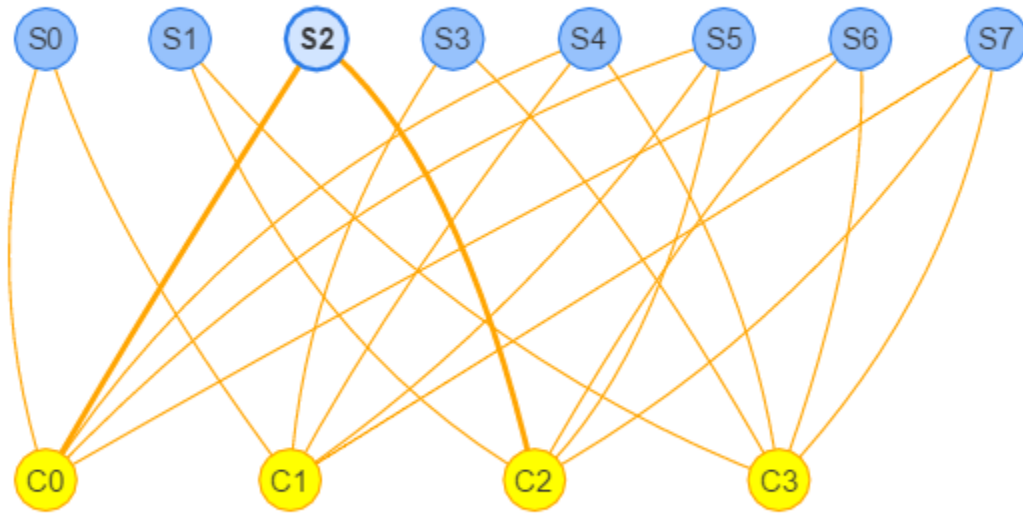
**степень контрольного узла** - это количество ребер, соединенных с контрольным узлом, т.е. Количество единиц в строке матрицы проверки четности, соответствующей этому контрольному узлу

**последовательность степеней символьных узлов** - это список, содержащий степени всех узлов символов

**последовательность степеней узлов проверки** - это список, содержащий степени всех контрольных узлов

**обхват** - это длина самого короткого цикла на графике Таннера

**подграф узла глубины  $l$**  определяется путем представления графа в виде дерева, начиная с заданного узла, пока глубина дерева не достигнет  $l$ . Узлы, уже включенные в дерево, не нужно добавлять снова, поскольку мы имеем дело только с первыми появлениями узлов. Итак, если все узлы будут включены до того, как глубина достигнет  $l$ , расширение дерева прекращается.



## Входные параметры алгоритма PEG

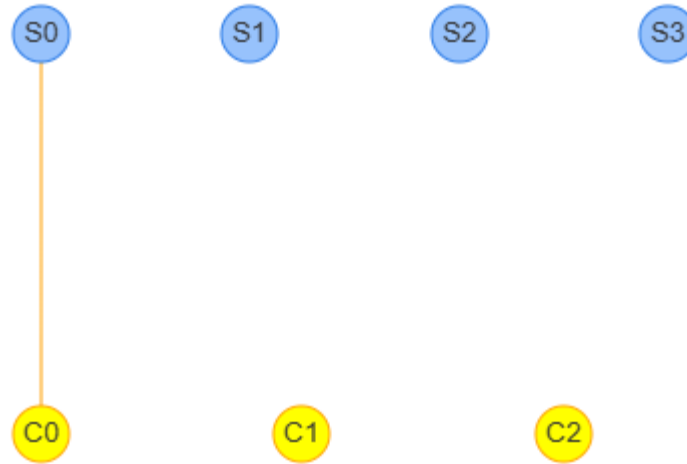
Алгоритм начинает работу с тремя параметрами:

- $m$  — количество узлов-проверок (число строк матрицы  $H$ )
- $n$  — количество узлов-символов (число столбцов матрицы  $H$ )
- Последовательность степеней узлов-символов  $[d_1, d_2, \dots, d_n]$  — массив, где  $d_i$  указывает, сколько рёбер должно быть у узла-символа  $s_i$

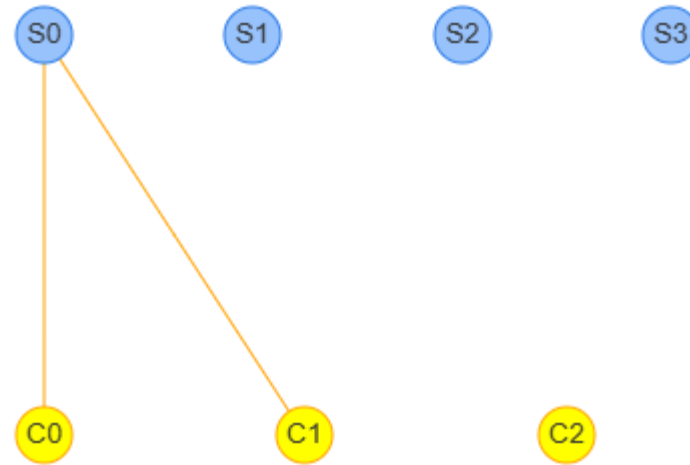
**Важно:** Количество узлов и рёбер известно заранее. Алгоритм отвечает только за оптимальное размещение этих рёбер.

## Структура алгоритма PEG

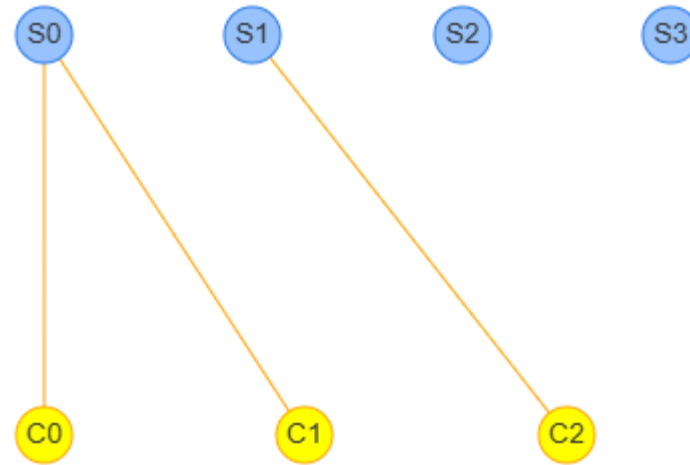
1. ЗАПУСК
2. ВВОД ( $n$ ,  $m$ , последовательность степеней)
3. Инициализация пустого графа с узлами, без ребер
4. ДЛЯ каждого узла символа  $S_i$ :
  5. ДЛЯ каждого ребра (от 1 до степени узла  $S_i$ ):
    6. Вопрос: Первое ребро?
      - ДА: Выбрать узел проверки наименьшей степени
      - НЕТ: Проверить покрытие подграфа:
        - Вопрос: Все узлы покрыты?
          - НЕТ: Выбрать «непокрытый» узел проверки с наименьшей степенью
          - ДА: Выбрать «самый дальний» по подграфу узел проверки с наименьшей степенью
  7. ДОБАВИТЬ РЕБРО (соединить выбранный символьный и проверочный узлы)
  8. Остались ребра? Вернуться к шагу 5
  9. Остались символьные узлы? Вернуться к шагу 4
10. ВЫВОД (граф Таннера)
11. ЗАВЕРШЕНИЕ



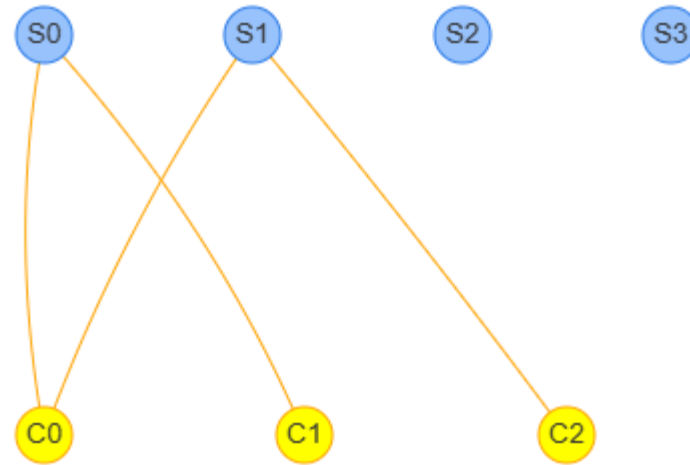
$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$



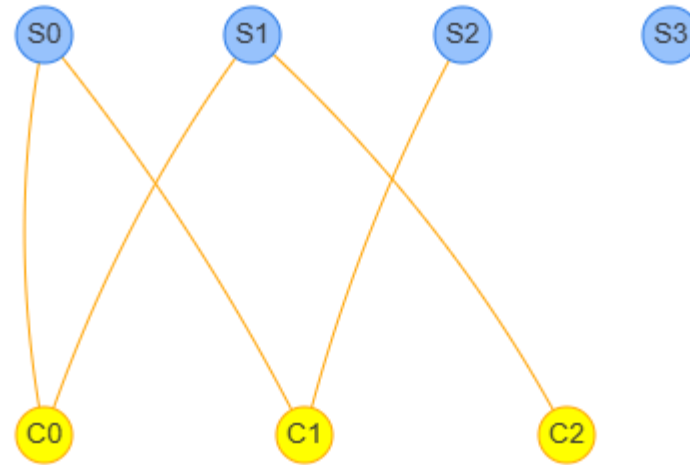
$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$



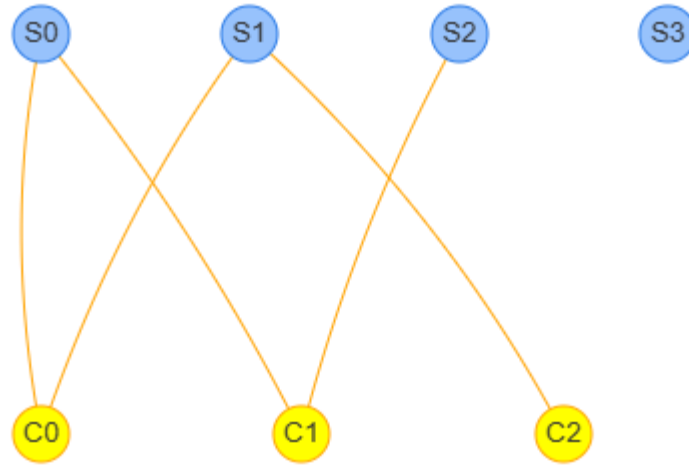
$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$



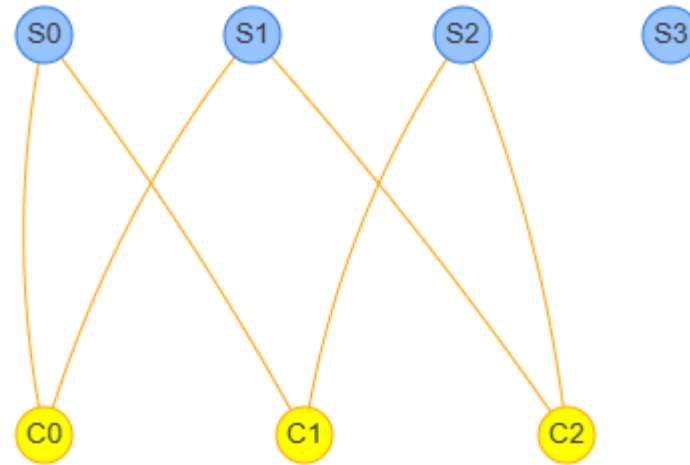
$$\begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$



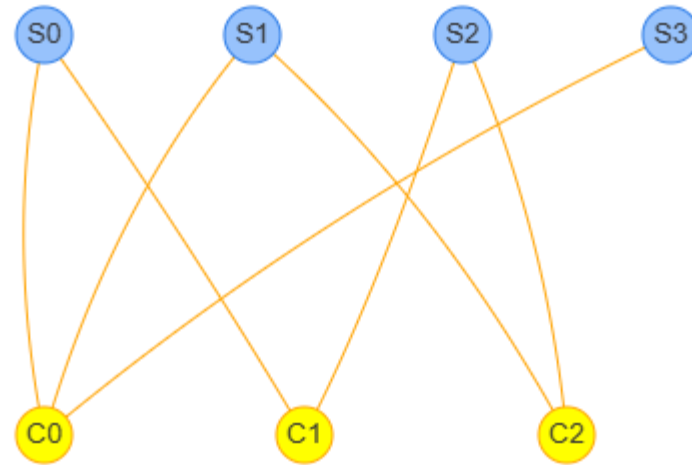
$$\begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$



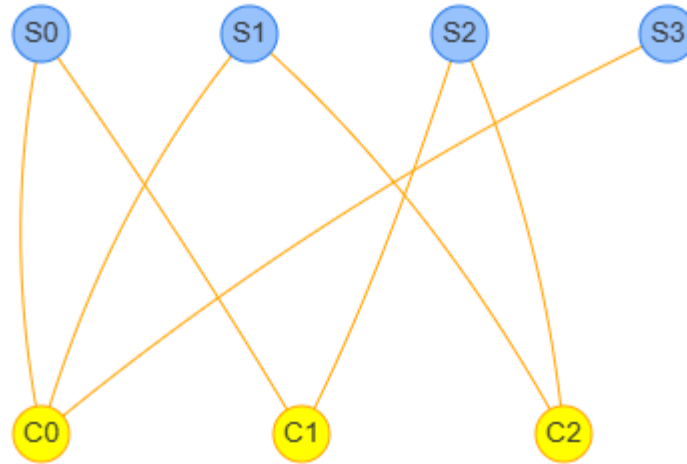
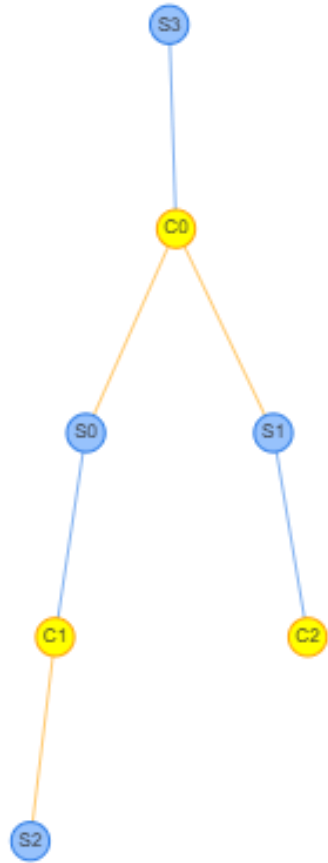
$$\begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$



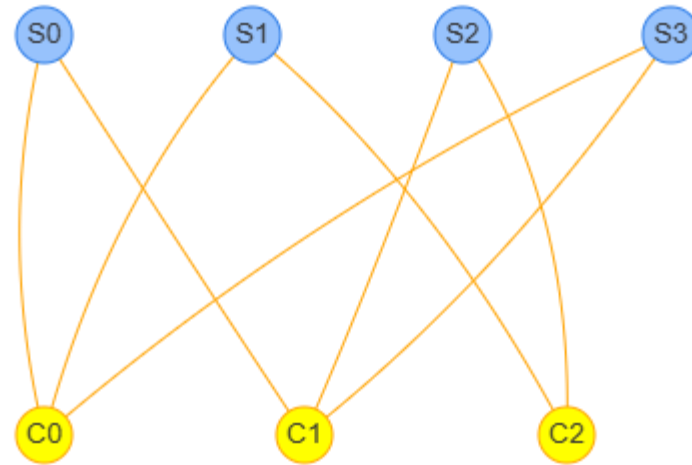
$$\begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$



$$\begin{bmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$



$$\begin{bmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$



$$\begin{bmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

