алгоритму двоичной сортировки, а номера вагонов запишем в фибоначчиевой системе счисления. Вагоны, которые на k-м шаге попали в стек с номером 1 (т.е. у которых в k-м разряде стоит 1), будут вытянуты в хвосте состава и после следующего роспуска все вместе отправятся в 0-й стек (ведь перед 1 обязательно стоит 0!).

Возникает идея: не будем вытягивать эти вагоны из стека, а на k+1-м шаге перенумеруем стеки.

Алгоритм (*F*) поразрядной сортировки Фибоначчи

(1) Нумеруем вагоны в фибоначчиевой системе.

(2) На k-м шаге распускаем состав в соответствии с поразрядной сортировкой.

(3) Затем вытягиваем вагоны из 0-го стека.

(4) Перенумеровываем стеки для k + 1-го шага.

Упражнение 3. Разберитесь на примерах решения задач 6,б) и 7,б), как работает алгоритм (*F*).

Задача 13. Попробуйте, прежде чем читать дальше, обобщить этот алгоритм для горки с большим, чем 2, числом стеков.

Обобщение алгоритма (*F*) на несколько стеков

Прежде чем обобщать алгоритм сортировки Фибоначчи для горки с числом стеков большим двух, попробуем переформулировать свойство фибоначчиевой системы из задачи 12.

Запишем его так:

ф2.0) перед цифрой 0 может стоять любая цифра,

ф2.1) перед цифрой 1 может стоять только цифра 0.

Обобщим эти свойства для создания «системы счисления» для любого k, например для k = 3.

Построим «систему счисления», использующую три цифры 0, 1, 2 и удовлетворяющую следующим свойствам:

ф3.0) перед цифрой 0 может стоять любая цифра,

ф3.1) перед цифрой 1 может стоять только цифра 0,

ф3.2) перед цифрой 2 может стоять только цифра 1.

Вот набор одноразрядных чисел этой системы: 0, 1, 2.

Множество двузначных чисел этой системы: 00, 01, 10, 12, 20.

Множество трехзначных чисел включает в себя 9 чисел:

000, 001, 010, 012, 100, 101, 120, 200, 201.

Уже по набору этих «чисел» видно, что $3 < W(3, 3) = 4$; $5 < W(4, 3) = 7$; $9 < W(5, 3) = 13$. Т.е. мы получили что-то не то.

Кроме того, то, что мы сейчас определили, не является системой счисления. Ведь в любой системе счисления, если существует число 2, то должно существовать и число 02, и число 002 и т.д. А в нашей «системе» есть число 2, но нет числа 02. Наши числа — это, скорее, «слова в некотором языке с алфавитом и с некоторыми правилами». Эти правила называются *контекстно-свободной грамматикой.*

Постараемся объяснить, что это такое. Если вас заинтересует, как применяются КС-грамматики в программировании, то и об этом можно прочитать в упоминавшейся уже книге А.Х.Шеня.

Контекстно-свободные грамматики

Рассмотрим конечный алфавит, обозначим его буквы *a*, *b*, *c*, ... Множество слов в этом алфавите мы будем называть *языком*. Но в качестве слов мы будем рассматривать не произвольные наборы букв, а только те, которые можно получить по некоторым правилам, называемым *грамматическими*. Грамматические правила устроены так: некоторые слова или подслова можно заменять на другие.

Более формально,

Определение 5. Для того чтобы задать контекстно-свободную грамматику (КС-грамматику), нужно

(1) задать множество символов алфавита, эти символы называются *терминальными* (окончательными);

(2) задать некоторое другое множество символов, эти символы называются *нетерминальными* (промежуточными);

(3) выбрать среди нетерминалов один символ, называемый *начальным*;

(4) задать конечное число правил, имеющих вид $S \to X$, где S — некоторый нетерминальный символ, а X — слово, в которое могут входить как терминальные, так и нетерминальные символы.

Выводом в этой грамматике называется последовательность слов X_0, X_1, ..., X_n, в которой X_0 — начальный символ, а X_{k+1} получается из X_k заменой некоторого нетерминального символа S на слово X по одному из правил грамматики. Слово, составленное из терминальных символов, называется *выводимым*, если существует вывод, который этим словом кончается. Множество всех выводимых слов, состоящих из терминальных символов, называется *языком, порождаемым данной грамматикой.*

Обычно интересуются вопросом: «выводимо ли данное слово в данной грамматике?» Алгоритмы, которые для любого слова отвечают на этот вопрос, составляют существенную часть современных трансляторов, которая называется «грамматическим разбором».

Наша задача скромнее: «построить КС-грамматику, поразрядная сортировка p-буквенных слов языка которой эквивалентна алгоритму $\left(C_k(p+1)\right)$».

Пример 3. Рассмотрим алфавит, состоящий из одной буквы *a* и правил

1) $S \to$,

2) $S \to aS$.

S — начальный символ. Обычно, если не оговорено противное, начальным символом считается символ, стоящий в левой части первого правила. Первое правило означает, что S можно заменять на пустое слово или, попросту говоря, вычеркивать из слова. Второе правило позволяет порождать новые слова, приписывая к ним букву *a*. Например, если нам нужно породить слово *aaa*, то, применяя второе правило, мы можем написать $S \to aS \to aaS \to aaaS$, а применив первое правило — $aaa \to Saaa$. Этот процесс, использующий грамматические правила языка, и есть вывод.

Упражнение 4. Сколько слов длины *n* (такие слова мы будем еще называть *n*-буквенными) в этом языке?

Пример 4. Рассмотрим язык всех слов в алфавите из двух букв *a* и *b*. Правила грамматики здесь такие:

1) $S \to$,

2) $S \to aS$,

3) $S \to bS$.

Упражнение 5. Выведите слова *aababba*.

Пример 5. Следующий язык описывает все регулярные скобочные структуры. Регулярные скобочные