Число вытягиваний (p) \ Число стеков (k)	1	2	3	4	5	6	7	8
1	1	1	1	1	1	1	1	1
2	1	2	2	2	2	2	2	2
3	1	3	4	4	4	4	4	4
4	1	5	7	8	8	8	8	8
5	1	8	13	15	16	16	16	16
6	1	13	24	29	31	32	32	32
7	1	21	44	56	61	63	64	64
8	1	34	81	108	120	125	127	128

Рис. 8

(3) зеленые вагоны распускаются в первые $(k-1)$ стеки в соответствии с алгоритмом $(C_k(p))$ (они помнят, перед какими вагонами они стояли), а желтые в последний.

Теперь в первых $(k-1)$ стеках все зеленые вагоны стоят левее синих.

Вспоминая номера синих и желтых вагонов при сортировке алгоритмом $(C_k(p))$ (зеленые вагоны получают номера соответствующих синих), мы получим ситуацию после первого роспуска в алгоритме $(C_k(p))$. Дальше сортируем с помощью алгоритма $(C_k(p))$. Теорема 5 доказана.

Окончательный вид таблицы $W(p, k)$ показан на рисунке 8.

О некоторых рекурсивных программах и избавлении от рекурсии

Этот параграф для тех, кто умеет программировать или любит решать олимпиадные задачи по программированию. Но в нем может разобраться и любой школьник, а может и безболезненно пропустить его.

В программистском практикуме встречается задача о вычислении s-го числа Фибоначчи:

$$F(0) = 1,\ F(1) = 1,\ F(s) = F(s-1) + F(s-2),\ s > 1.$$

Вот рекурсивная программа вычисления $F(s)$:

```
functionF(s : integer) : integer;
    |{s >=0}
begin
    |if(s = 0)or(s = 1)thenbegin
    ||F :=1;
    |endelsebegin1 < s
    ||F :=F(s − 1) + F(s − 2);
    |end;
end
```

Что можно сказать о времени работы этой программы и необходимой для нее памяти?

Используемая программой память пропорциональна s. Глубина рекурсии (s) умножается на количество памяти, необходимое для одного экземпляра процедуры, т.е. на константу. А вот время растет экспоненциально, так как вычисление $F(s)$ сводится к двум вызовам $F(s-1)$ и $F(s-2)$, те — к четырем вызовам $F(s-2)$, $F(s-3)$, $F(s-3)$ и $F(s-4)$ и так далее. Поэтому время растет экспоненциально, правда, не как 2^s, а как $F(s) \simeq \lambda_1^s$, где $\lambda_1 = \frac{1+\sqrt{5}}{2}$ — наибольший корень уравнения $\lambda^2 = \lambda + 1$.

Хорошо известно, как избежать рекурсии (если вы случайно не знаете, как это сделать, то попробуйте, прежде чем читать дальше, придумать самостоятельно нерекурсивную программу вычисления $F(s)$).

Нужно вместо одного значения $F(s)$ вычислить одновременно пару (вектор) $(F(s), F(s-1))$, начиная с пары $(1, 1)$. Вот кусок этой программы:

```
begin
    |c := b;
    |b := a;
    |a := a + c;
end
```

Повторив это вычисление s раз, начиная со значений $a := 1$; $b := 1$;, вы получите в a значение $F(s)$. Время работы такого алгоритма порядка s, а памяти он требует порядка константы.

Упражнение 2. Допишите аккуратно вторую программу и сравните ее работу с работой первой.

Прием, который был применен, обычно называется *индуктивным расширением по Кушниренко*. Прочитать об этом можно в совершенно замечательной книге А.Х.Шеня «Программирование: Теоремы и задачи».

В нашем случае ситуация аналогична и работа рекурсивной процедуры усугубляется еще тем, что каждый рекурсивный вызов обращается не к простой функции, а к сложной программе.

Избавление от рекурсии в нашем случае заключается в обращении к алгоритму поразрядной сортировки. Только система счисления нам понадобится не 2-ичная или основанная на других геометрических прогрессиях, а основанная на последовательности Фибоначчи.

Еще раз о системах счисления

Более общий способ построения систем счисления таков. Задается некоторый «базис» системы — набор целых чисел $1 = q_0 < q_1 < q_2 < \ldots$ Для того чтобы записать число n в этой системе счисления, нужно выбрать наибольшее из чисел $q_s \le n$ и разделить n на q_s с остатком: $n = a_s q_s + n_{s-1}$. Затем нужно разделить n_{s-1} на q_{s-1} с остатком: $n_{s-1} = a_{s-1}q_{s-1} + n_{s-2}$ и т.д. В результате получатся такие равенства:

$$n = a_s q_s + n_{s-1},$$
$$n_{s-1} = a_{s-1}q_{s-1} + n_{s-2},$$
$$\ldots$$
$$n_0 = a_0 q_0 \left(n_0 = a_0\right),$$

или

$$n = a_s q_s + a_{s-1}q_{s-1} + \ldots + a_0 q_0 .$$

Число n записывается в виде выражения $\overline{a_s a_{s-1} \ldots a_0}$, и эта запись называется записью числа в системе счисления с базисом $1 = q_0 < q_1 < q_2 < \ldots$ Числа a_i играют роль цифр и удовлетворяют неравенствам $a_i < \frac{q_{i+1}}{q_i}$. Заметим, что в таких системах счисления в разных разрядах могут стоять, вообще говоря, разные наборы цифр. Десятичная система обладает базисом из степеней десятки, двоичная из степеней двойки, q-ичная из степеней числа q.

Для следующего алгоритма нам понадобится система счисления с базисом 1, 2, 3, 5, 8, 13, ... — последовательностью чисел Фибоначчи. Такая система так и называется *фибоначчиевой*. Вот пример записи первых девяти чисел в фибоначчиевой системе счисления:

$$0 = 0_f,\quad 1 = 1_f,\quad 2 = 10_f,\quad 3 = 100_f,$$
$$4 = 101_f,\quad 5 = 1000_f,\quad 6 = 1001_f,$$
$$7 = 1010_f,\quad 8 = 10000_f .$$

С этой системой можно познакомиться по книге Н.Н.Воробьева «Числа Фибоначчи».

Задача 12. Докажите, что в системе счисления Фибоначчи, как и в двоичной, используются всего две цифры 0 и 1 и запись любого числа в фибоначчиевой системе счисления не содержит двух рядом стоящих единиц.

Посмотрим теперь, что будет происходить на горке с двумя стеками, если мы будем сортировать состав по