



лекция 23



ПЕРЕСТАНОВКИ

Про соревнования 3 мая 2013

- Соревнования будут проходить в пятницу 3 мая, с 14-00, в 305-307 классах.
- Подходить для регистрации к 13-30 к 306 классу со студенческим билетом.
- Предложение такое, что одна решенная задача будет засчитываться как задача на экзамене, так как соревнования командные, то при решении в команде из 2-х человек надо будет решить две задачи.
- И при этом и проблем с допуском до экзамена у преподавателя в группе не должно быть.

План лекции

- Перестановки и инверсии
- Инверсии
 - Связь со сложностью сортировки
 - Алгоритм восстановления перестановки по таблице инверсий
 - Итерационный алгоритм генерации всех таблиц инверсий
- Перебор перестановок
 - Рекурсивный, через перебор таблиц инверсий, итерационный с лексикографическим упорядочением (Дейкстры)

Перестановки

- Перестановкой порядка N называется расположение N различных объектов в ряд в некотором порядке
- Для объектов a, b и c есть шесть перестановок
 - $abc, acb, bac, bca, cab, cba$.
- Далее будем рассматривать перестановки элементов множества $\{1, 2, 3, \dots, N\}$

Перестановки

- Для множества из N элементов можно построить $N!$ различных перестановок
 - Первую позицию можно занять N способами
 - Вторую — $(N - 1)$ способом и т. д.
 - На последнее место можно поставить только один оставшийся элемент
- Следовательно, число перестановок из N элементов равно $N * (N - 1) * (N - 2) * \dots * 1 = N!$

Инверсии

- Пусть даны множество из N элементов $1, 2, 3, \dots, N$ и его перестановка $a_1, a_2, \dots, a_{N-1}, a_N$
- Пара (a_i, a_j) называется инверсией (инверсионной парой) перестановки, если $a_i > a_j$ при $i < j$
- Единственной перестановкой, не содержащей инверсий, является упорядоченная перестановка $1, 2, 3, \dots, N$
- Каждая инверсия — это пара элементов перестановки, нарушающих ее упорядоченность

Инверсии -- пример

- Перестановка 4, 1, 3, 2 имеет четыре инверсии (4,1), (3,2), (4,3) и (4,2)
- Почему?

Таблица инверсий

- Таблицей инверсий перестановки $a_1, a_2, \dots, a_N$ называется последовательность чисел $b_1, b_2, \dots, b_N$, где b_j = число инверсий вида (x, j)
- Пример $\Pi=591826473$ $ТИ=236402210$

Свойства таблиц инверсий

- Для элементов таблицы инверсий справедливы неравенства
 - $b_N = 0$
 - $0 \leq b_i \leq N-i$
 - $0 \leq b_1 \leq N-1$
- Таблица инверсий определяет перестановку однозначно

Построение перестановки по таблице инверсий

- На каждом шаге вставляем следующий по величине элемент с учетом того, сколько элементов, больших него, должно стоять перед ним
- Таблица инверсий: 2 3 6 4 0 2 2 1 0
- 9
9 8
9 8 7
9 8 6 7
5 9 8 6 7
5 9 8 6 4 7
5 9 8 6 4 7 3
5 9 8 2 6 4 7 3
5 9 1 8 2 6 4 7 3

Построение перестановки по таблице инверсий справа

Вход

$N > 0$ - количество элементов перестановки
 $b_1, b_2 \dots, b_N$ – ТИ, $0 \leq b_j \leq N - j$

- Алгоритм

Π = пустая последовательность

цикл по j от N вниз до 1

 вставить число j в Π после b_j элементов
конец цикла

- Выход

Π – перестановка, соответствующая ТИ

Построение перестановки по таблице инверсий слева

направо

Создается заготовка пустой перестановки длины N

- На каждом шаге для каждого элемента перестановки, начиная с 1, отсчитывается в ней столько пустых ячеек, какое число записано в соответствующей позиции в таблице инверсий
- В следующее за ними пустое место вставляется этот элемент

Таблица инверсий

2 3 6 4 0 2 2 1 0



Перестановка

5	9	1	8	2	6	4	7	3
---	---	---	---	---	---	---	---	---

Построение перестановки по таблице инверсий слева

Вход
направо

$N > 0$ - количество элементов перестановки

$b_1, b_2, \dots, b_N$ – ТИ, $0 \leq b_j \leq N - j$

- Алгоритм

P = последовательность из N пустых элементов

цикл по i от 1 до N с шагом 1 выполнять

пропустить b_i пустых мест в P

поместить i на следующее пустое место

конец цикла

- Выход

P – перестановка, соответствующая ТИ



Инверсионный метод поиска всех перестановок

- Таблицы инверсий взаимно однозначно соответствуют перестановкам
 - Почему?
- Перебор ТИ сводится к перебору перестановок и наоборот

Инверсионный метод поиска всех перестановок

- Рассмотрим таблицу инверсий как N -значное число в «системе счисления», где количество цифр, которое можно использовать в i -м разряде (с конца, начиная с 0) равно i
- Возьмем «нулевую» таблицу и будем последовательно прибавлять к ней в нашей СС единицу, пользуясь алгоритмом сложения с переносом для многоразрядных чисел
- Последовательно получим все ТИ и для каждой восстановим перестановку

Генерация таблиц инверсии

4	3	2	1	0
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	0	2	0	0
0	0	2	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
0	1	2	0	0
...	...	...	...	...
4	3	2	1	0

Шаг 0

Шаг 1

Шаг 2

Шаг 3

Шаг 4

Шаг 5

Шаг 6

Шаг 7

Шаг 8

Шаг 9

Шаг 10

...

Шаг 119

Нахождение следующей таблицы инверсий

- $B = b_1, b_2, \dots, b_N$ – ТИ

- $i = N$

не_все = истина

пока не_все

$x = b_i + 1$

если $x > N - i$ то

$b_i = 0$

$i = i - 1$

иначе

$b_i = x$

не_все = ложь

все

все

- Что же тут не так?

Поиск следующей по алфавиту перестановки (алг. Дейкстры)

- П. $b = b_1, b_2, \dots, b_N$ предшествует п. $c = c_1, c_2, \dots, c_N$ в алфавитном (лексикографическом) порядке, если $b_1=c_1, b_2=c_2, \dots, b_{k-1}=c_{k-1}$ и $b_k < c_k$ для некоторого k
- Перестановка 1 2 3 4 5 предшествует перестановке 1 2 4 5 3 ($k = 3$)
- Первой перестановкой в алфавитном порядке является перестановка 1, 2, 3, ..., N , а последней — $N, N-1, N-2, \dots, 1$

Алгоритм Дейкстры

- От заданной перестановки перейдем к следующей за ней в алфавитном порядке и т.д., пока не получим $N, N-1, \dots, 1$
- Например, для перестановки
1 4 6 2 9 5 8 7 3
следующей по алфавиту является
перестановка 1 4 6 2 9 7 3 5 8

Генерация следующей по алфавиту перестановки

- Вход
 $\Pi = a_1, a_2, \dots, a_{[N-1]}, a_N$ – перестановка $N > 0$ элементов
- Алгоритм
Начиная с последнего элемента, найдем такой номер i , что $a_{i+1} > \dots > a_N$ и $a_i < a_{[i+1]}$
Если такого i нет, то Π – последняя в алф. порядке
Обозначим a_j наименьший элемент среди тех $a_{[i+1]}, \dots, a_N$, которые строго больше a_i
Поменяем местами a_i и a_j
Упорядочим $a_{[i+1]}, \dots, a_N$ по возрастанию
- Выход
Следующая по алфавиту перестановка

Пример построения следующей по алфавиту перестановки

Для перестановки

1 4 6 2 9 5 8 7 3

Найти следующую по алфавиту.

Шаг 1: 1 4 6 2 9 5 8 7 3

i j

Шаг 2: Поменять местами



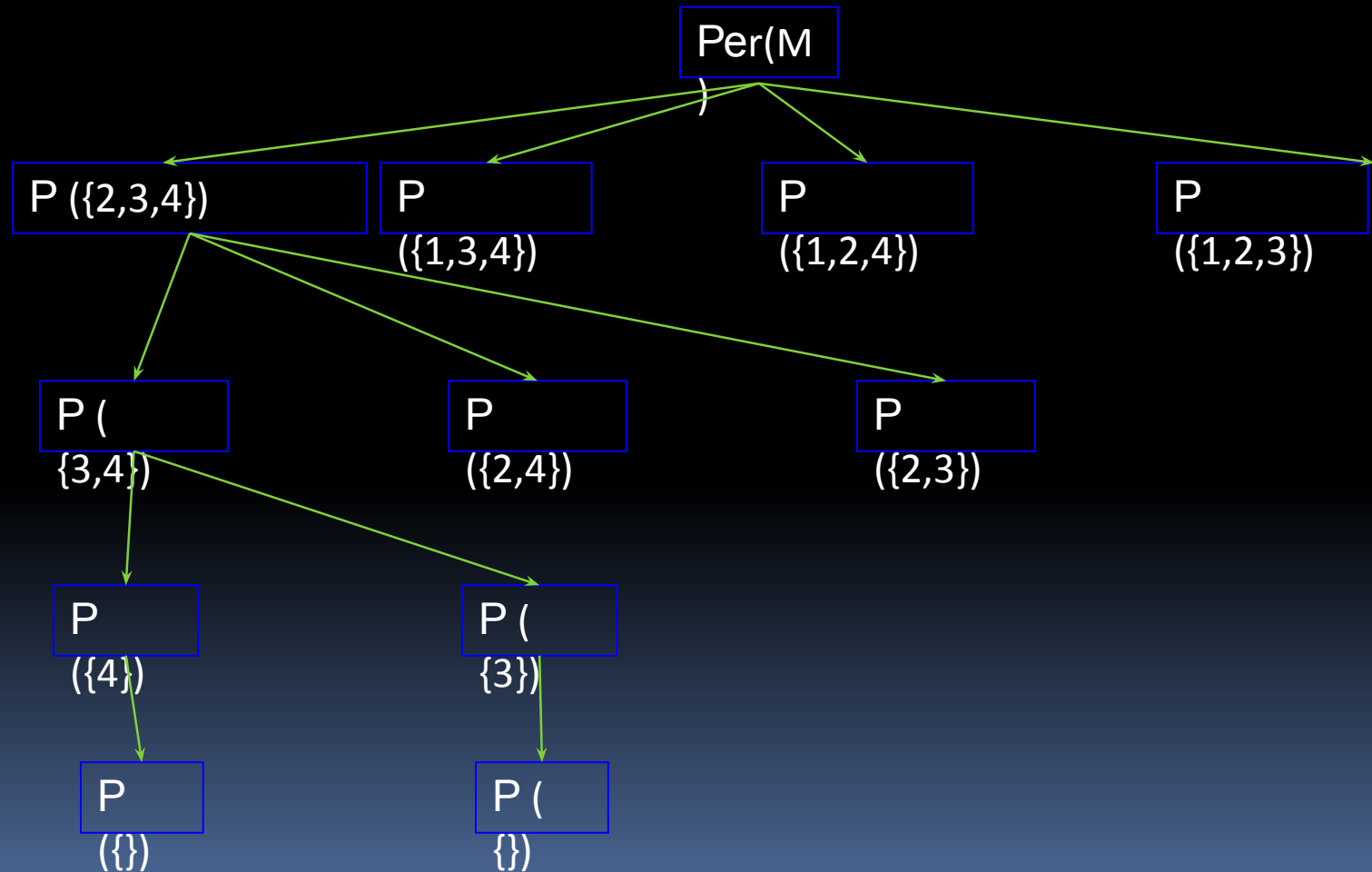
Шаг 3: Перевернуть хвост



Рекурсивный метод поиска всех перестановок

- Метод рекурсивного перебора перестановок основан на идее сведения исходной задачи к аналогичной задаче на меньшем наборе входных данных
- $P(\{\ }) = \{\ }$
- $P(M) = \bigcup_{x \in M} \{(x, p) | p \in P(M \setminus \{x\})\}$

Пример рекурсивного перебора для $M = \{1, 2, 3, 4\}$



Реализация на языке Си

```
▪ typedef char string[256];
void permut(string start, string rest) {
    int lenr = strlen(rest), lens = strlen(start);
    int i=0;
    string s1="", s2="";
    if (lenr == 0) printf("%s\n", start);
    else {
        for (i = 0; i < lenr; i++) {
            /* Добавляем i-ый символ к строке start */
            strcpy(s1,start);
            strncpy(s1+lens,rest+i,1);
            strncpy(s1+lens+1,"\\0",1);
            /* Удаляем i-ый символ из строки rest */
            strncpy(s2,rest,i);
            strncpy(s2+i,rest+i+1,lenr-i-1);
            strncpy(s2+lenr,"\\0", 1);
            /* Рекурсивный переход */
            permut( s1, s2 );
        }
        /* s1+lens ≡ &(s1[lens]) */
        /* rest+i ≡ &(rest[i]) */
    }
}
```

Реализация на языке Си

```
#include <string.h>
typedef char mystring_t[256];
void permut(mystring_t start, mystring_t rest)
{
    int lenr = strlen(rest);
    if (0 == lenr)
        printf("%s\n", start);
    else
    {
        int i;
        for (i = 0; i < lenr; i++)
        {
            mystring_t mystart="", myrest="";
            strcpy (mystart, start);
            strcpy (myrest, rest);
            append (mystart, delete (myrest, i) );
            permut (mystart, myrest);
        }
    }
}
// TODO: написать append и delete
```

Генерация всех перестановок методом Кнута

- **Идея**
Рекурсивная генерация начиная с пустой перестановки методом расширения базового множества перестановки элементами 1, 2, 3, и т.д.
Если построены все перестановки длины N , то для каждой такой перестановки можно построить $N+1$ перестановку длины $N+1$
- **Пример**
Для перестановки 3241 можно построить 5 различных перестановок длины 5
53241
35241
32541
32451
32415

Генерация перестановок методом Кнута – способ 1

- Пусть дана перестановка длины N
- Допишем в конец перестановки числа $(2i+1)/2$ ($0 \leq i \leq N$)
- Перенумеровать элементы полученных перестановок в порядке их возрастания

- Пример

3 2 4 1 0.5 --> 4 3 5 2 1

3 2 4 1 1.5 --> 4 3 5 1 2

3 2 4 1 2.5 --> 4 2 5 1 3

3 2 4 1 3.5 --> 3 2 5 1 4

3 2 4 1 4.5 --> 3 2 4 1 5

Генерация перестановок методом Кнута – способ 2

- Пусть дана перестановка длины N : $a_1 a_2 \dots a_N$
- Дописать в конец перестановки числа k
 $1 \leq k \leq N + 1$
- Все $a_i > k$ заменить на $a[i] + 1$

- Пример

3 2 4 1 1 --> 4 3 5 2 1

3 2 4 1 2 --> 4 3 5 1 2

3 2 4 1 3 --> 4 2 5 1 3

3 2 4 1 4 --> 3 2 5 1 4

3 2 4 1 5 --> 3 2 4 1 5

Заключение

- Перестановки и инверсии
- Инверсии
 - Связь со сложностью сортировки
 - Алгоритм восстановления перестановки по таблице инверсий
 - Итерационный алгоритм генерации всех таблиц инверсий
- Перебор перестановок
 - Рекурсивный, через перебор таблиц инверсий, итерационный с лексикографическим упорядочением (Дейкстры), Кнута с перенумерацией