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 преподователя 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, … , 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 $N * (N - 1) * (N - 2) * \ldots * 1 = N!$

Инверсии

- Пусть даны множество из N элементов 1,2, 3,..., N и его перестановк$a_1, a_2, ..., a_{N-1}, a_N$
- Пара(a_i, a_j) называется инверсией (инверсионной парой) пере$a_i > a_j$вки , если при i < j
- Единственной перестановкой, не содержащей инверсий, является упорядоченная перестановка 1, 2, 3, … , 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 a1, a2, …, aN называется последовательность чисел b1, b2, …, bN, где bj = число инверсий вида (x, j)

- Пример П=591826473 ТИ=236402210

Свойства таблиц инверсий

- Для элементов таблицы инверсий справедливы неравенства
  - $b_N = 0$
  - $0 \le b_i \le N-i$
  - $0 \le b_1 \le N-1$
- 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 налево

- Вход

  N > 0 - количество элементов перестановки

  b1, b2 …, bN – ТИ, 0 ≤ bj ≤ N − j

- Алгоритм

  П = пустая последовательность

  цикл по j от N вниз до 1

  вставить число j в П после bj элементов

  конец цикла

- Выход

  П − 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 направо

- Вход
  N > 0 - количество элементов перестановки
  b1, b2 …, bN – ТИ, 0 ≤ bj ≤ N – j
- Алгоритм
  П = последовательность из N пустых элементов
  цикл по i от 1 до N с шагом 1 выполнять
  пропустить bi пустых мест в P
  поместить i на следующее пустое место
  конец цикла
- Выход
  П – 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	Шаг 0
0	0	0	1	0	Шаг 1
0	0	1	0	0	Шаг 2
0	0	1	1	0	Шаг 3
0	0	2	0	0	Шаг 4
0	0	2	1	0	Шаг 5
0	1	0	0	0	Шаг 6
0	1	0	1	0	Шаг 7
0	1	1	0	0	Шаг 8
0	1	1	1	0	Шаг 9
0	1	2	0	0	Шаг 10
…	…	…	…	…	…
4	3	2	1	0	Шаг 119

Нахождение следующей таблицы инверсий

- B = b1, b2, ..., bN – ТИ
- i = N
  не_всё = истина
  пока не_всё
  x = bi + 1
  если x > N – i то
  bi = 0
  i = i – 1
  иначе
  bi = x
  не_всё = ложь
  всё
  всё
- Что же тут не так?

Поиск следующей по алфавиту перестановки (алг. Дейкстры)

- П. b = b1, b2, …, bN предшествует п. c = c1, c2, …, cN в алфавитном (лексикографическом) порядке, если b1=c1, b2=c2, …, b[k-1]=c[k-1] и bk < ck 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, ..., 1

Алгоритм Дейкстры

- От заданной перестановки перейдем к следующей за ней в алфавитном порядке и т.д., пока не получим N, N-1, …, 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
  П = a1, a2, …, a[N-1], aN – перестановка N > 0 элементов
- Алгоритм
  Начиная с последнего элемента, найдем такой номер i, что ai+1 > … > aN и ai < a[i+1]
  Если такого i нет, то П – последняя в алф. порядке
  Обозначим aj наименьший элемент среди тех a[i+1], …, aN, которые строго больше ai
  Поменяем местами ai и aj
  Упорядочим a[i+1], …, aN 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

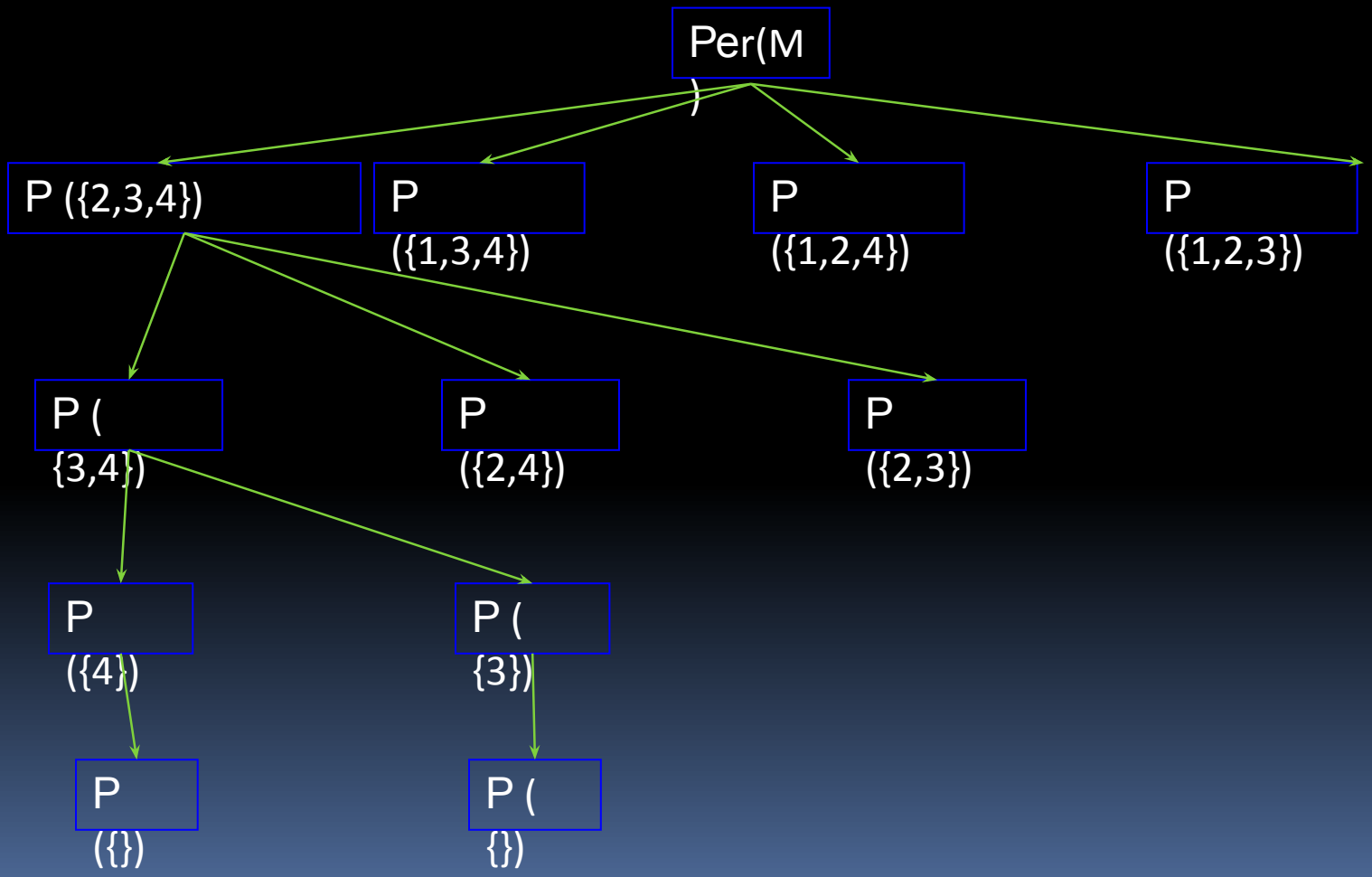
Шаг 3: Перевернуть хвост

Рекурсивный метод поиска всех перестановок

- Метод рекурсивного перебора перестановок основан на идее сведения исходной задачи к аналогичной задаче на меньшем наборе входных данных

- $P(\{\quad\}) = \{\}$
- $P(M) = \bigcup_{x \in M} \{(x, p) | p \in P(M \backslash \{x\})\}$

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 <= i <= 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: a1 a2 … aN
- Дописать в конец перестановки числа k
  1 <= k <= N +1
- Все ai > 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