Алгоритмы сортировки

- Пузырьковая сортировка
- Чет-нечетная сортировка
- Ранговая сортировка
- Корзиночная сортировка
- Выборочная сортировка
- Быстрая сортировка
- Шелл-сортировка
- Radix-сортировка
- Битоническая сортировка

Bubble sort

```
1.  procedure BUBBLE_SORT(n)
2.  begin
3.      for i := n − 1 downto 1 do
4.          for j := 1 to i do
5.              compare-exchange(a_j, a_{j+1});
6.  end BUBBLE_SORT
```

- Сложность алгоритма

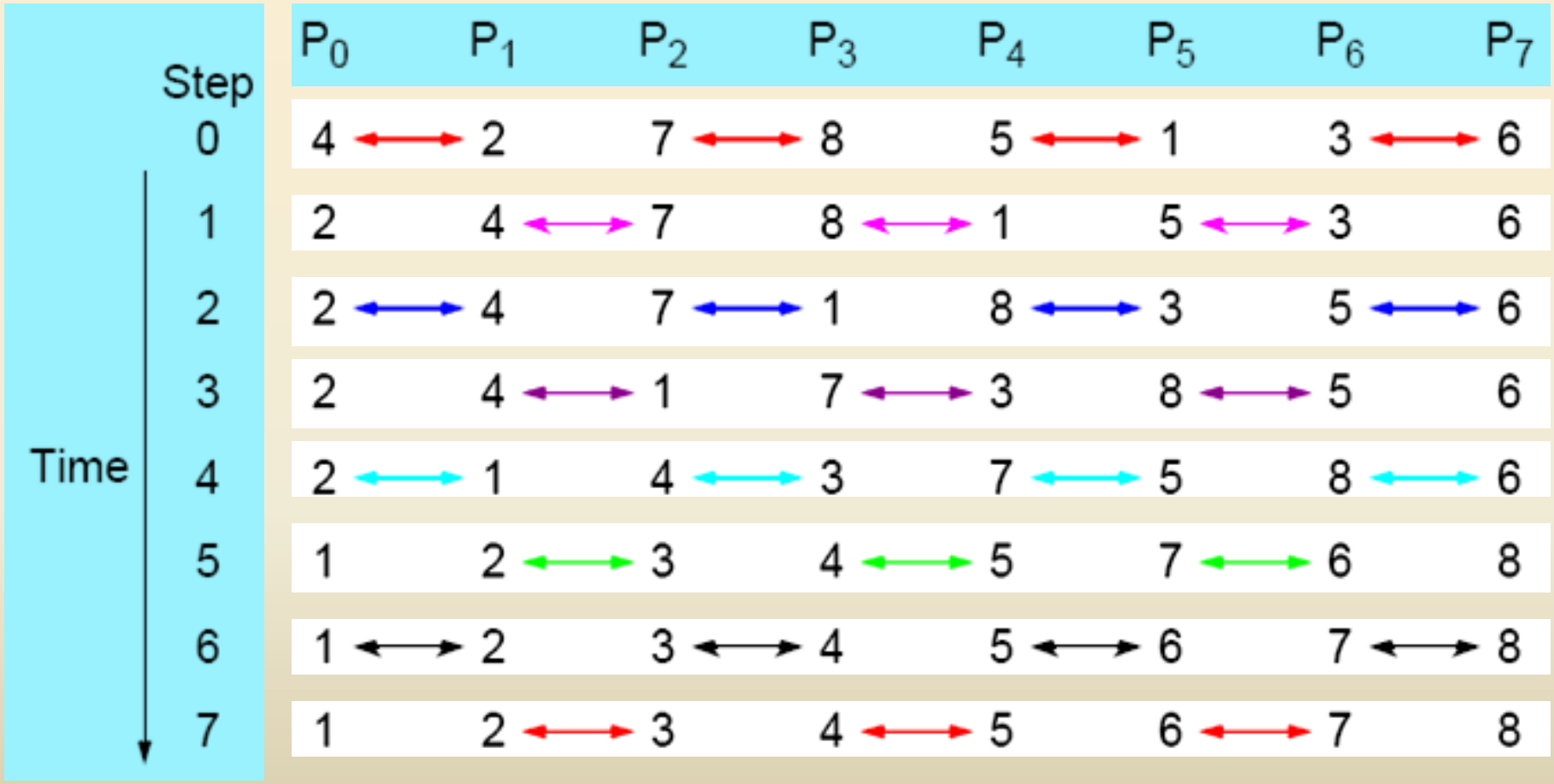
$$T_1(n) = (n-1) + (n-2) + .. + 1 = O(n^2)$$

Odd-even sort

- Переупорядочивание обменов элементов

```
1.   procedure ODD-EVEN(n)
2.   begin
3.       for i := 1 to n do
4.       begin
5.           if i is odd then
6.               for j := 0 to n/2 − 1 do
7.                   compare-exchange(a_{2j+1}, a_{2j+2});
8.           if i is even then
9.               for j := 1 to n/2 − 1 do
10.                  compare-exchange(a_{2j}, a_{2j+1});
11.      end for
12.  end ODD-EVEN
```

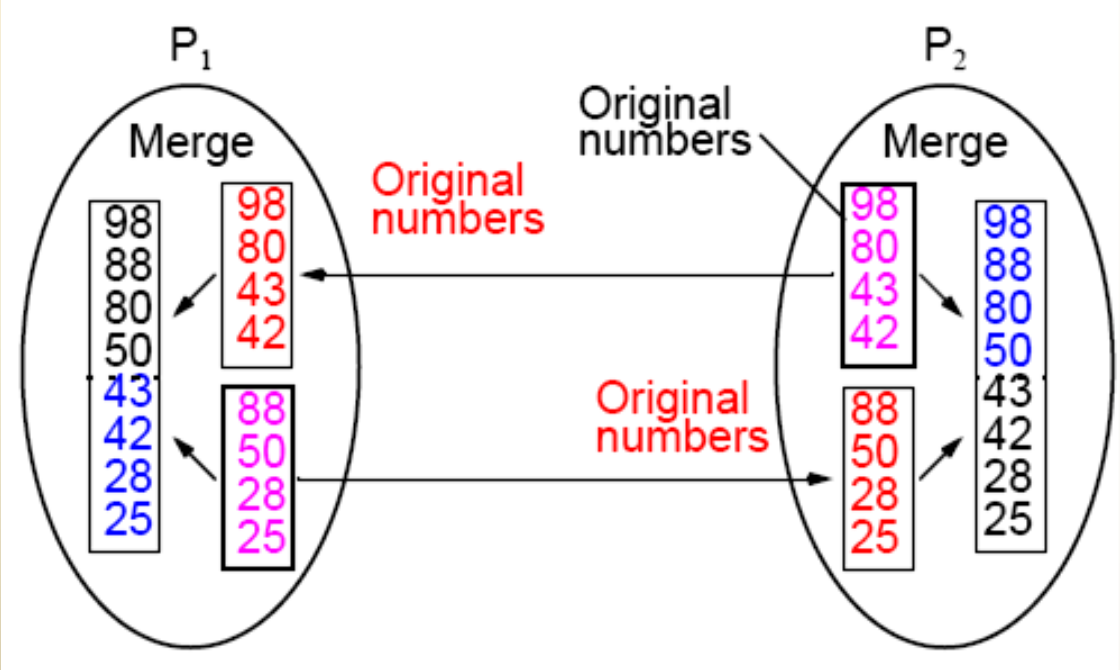
Чет-нечетная сортировка. P=n



Сложность: $T_{par} = \mathbf{O}(n)$ (for P=n)

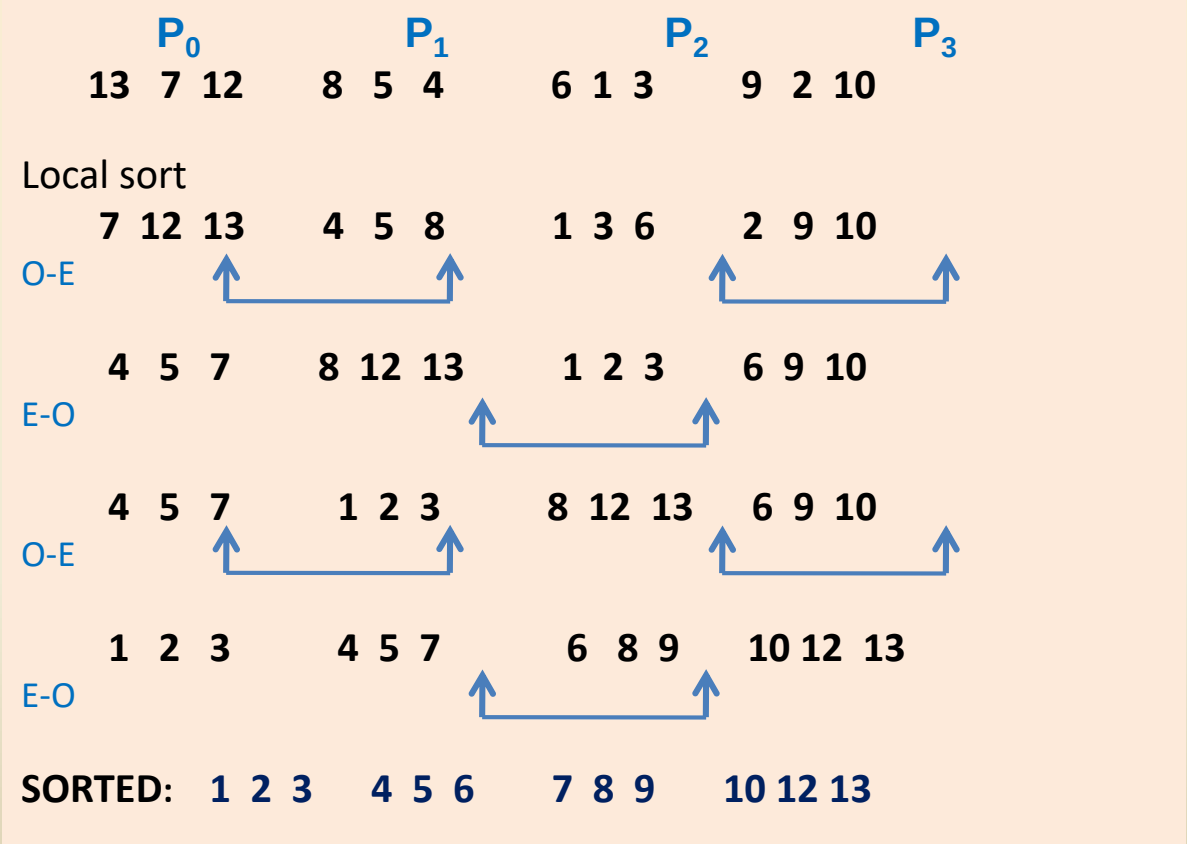
Базовая операция «merge-split»

- Базовая операция в последовательной сортировке – «сравнить-обменять» два элемента
- В параллельной сортировке – «объединить-разделить» для группы элементов



Odd-even sort. P << n

Каждый поток обрабатывает n/p элементов. Предварительно выполняется локальная сортировка элементов. Затем выполняется чет-нечентая перестановка



Сложность: T_{par} = (Local Sort) + (p merge-splits) +(p exchanges)

$$T_{par} = (n/p)\log(n/p) + p*(2 * n/p) = (n/p)\log(n/p) + 2n$$

Ранговая сортировка

- Ранговая сортировка (Enumeration Sort) – основана на вычислении ранга каждого элемента; в результирующей последовательности элементы располагаются в соответствии со своим рангом
- Ранг элемента a_i в Enumeration Sort – число меньших элементов
- Для каждого элемента необходимо выполнить (n-1) сравнений со всеми остальными элементами
- Сложность алгоритма: $T_{seq} = O(n^2)$

Ранговая сортировка

```
// по всем элементам
for (i = 0;     i < n;    i++)
    x = 0;
    //  вычисляем ранг элемента
    for (j = 0; j < n; j++)
        if   a[i] > a[j] OR  (  a[i] == a[j] AND j > i  )
             x++;

    b[x] = a[i];            // записываем в результирующий массив
```

Bucket Sort

- Сортировка осуществляется разделением всех чисел на упорядоченные «корзины»
- Корзины составляются таким образом, чтобы все элементы i-корзины были меньше, чем элементы j-корзины, если $i < j$.
- Алгоритм корзиночной сортировки:

  1. Оценить диапазон числовой последовательности a_{min}, a_{max}.

  2. Распределить элементы по p-корзинам в соответствии с равномерными границами

  3. Выполнить локальную сортировку в каждой корзине

  4. Объединить элементы корзин в общую последовательность

Sample Sort

- Разделение последовательности на части по «глобальным разделителям»; каждый процесс сортирует свою часть
- Выбор глобальных разделителей осуществляется таким образом, чтобы части были относительно одинаковые по размеру
- Основные этапы в Sample Sort:

  - первичное разделение набора на p-частей;

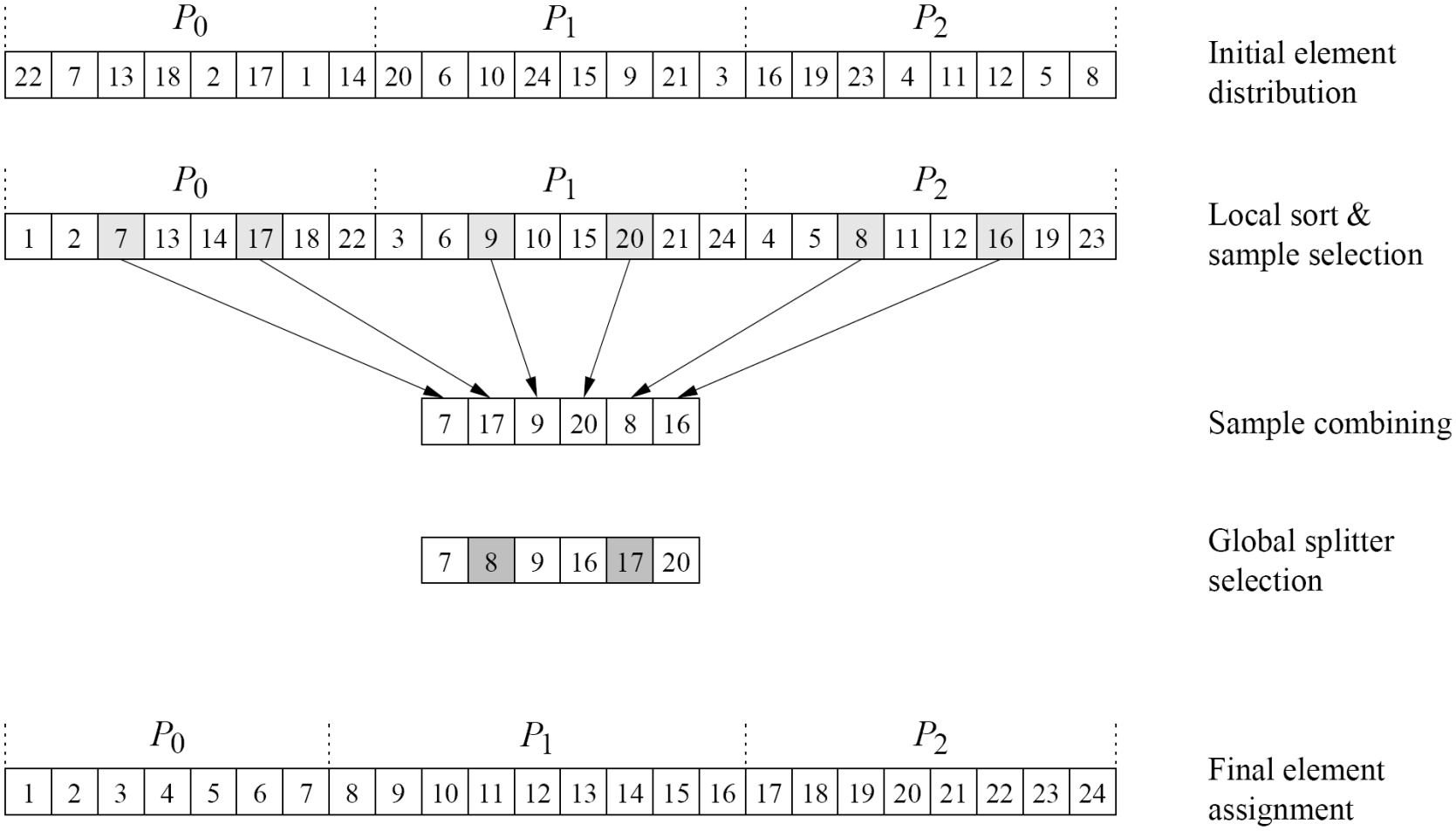
  - выбор локальных разделителей для каждой части;

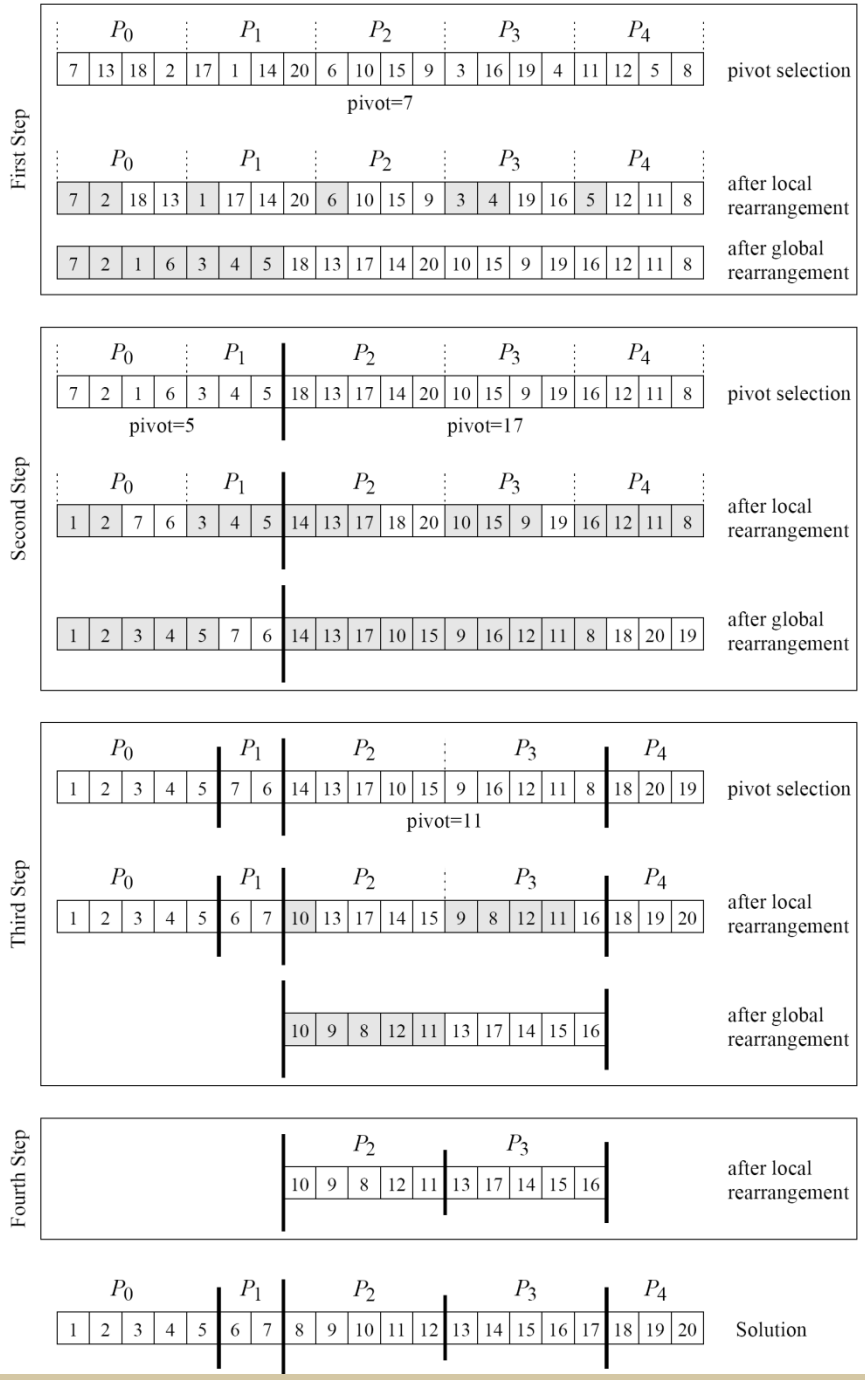
  - выбор глобальных разделителей на базе локальных;

  - разделение набора с учетом глобальных разделителей;
  - сортировка;

  - объединение;

Sample sort


P_0
P_1
P_2
22 7 13 18 2 17 1 14 20 6 10 24 15 9 21 3 16 19 23 4 11 12 5 8
Initial element distribution
P_0
P_1
P_2
1 2 7 13 14 17 18 22 3 6 9 10 15 20 21 24 4 5 8 11 12 16 19 23
Local sort & sample selection
7 17 9 20 8 16
Sample combining
7 8 9 16 17 20
Global splitter selection
P_0
P_1
P_2
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
Final element assignment


First Step
P_0 P_1 P_2 P_3 P_4
7 13 18 2 17 1 14 20 6 10 15 9 3 16 19 4 11 12 5 8
pivot selection
pivot=7
P_0 P_1 P_2 P_3 P_4
7 2 18 13 1 17 14 20 6 10 15 9 3 4 19 16 5 12 11 8
after local rearrangement
7 2 1 6 3 4 5 18 13 17 14 20 10 15 9 19 16 12 11 8
after global rearrangement
Second Step
P_0 P_1 P_2 P_3 P_4
7 2 1 6 3 4 5 18 13 17 14 20 10 15 9 19 16 12 11 8
pivot selection
pivot=5
pivot=17
P_0 P_1 P_2 P_3 P_4
1 2 7 6 3 4 5 14 13 17 18 20 10 15 9 19 16 12 11 8
after local rearrangement
1 2 3 4 5 7 6 14 13 17 10 15 9 16 12 11 8 18 20 19
after global rearrangement
Third Step
P_0 P_1 P_2 P_3 P_4
1 2 3 4 5 7 6 14 13 17 10 15 9 16 12 11 8 18 20 19
pivot selection
pivot=11
P_0 P_1 P_2 P_3 P_4
1 2 3 4 5 6 7 10 13 17 14 15 9 8 12 11 16 18 19 20
after local rearrangement
10 9 8 12 11 13 17 14 15 16
after global rearrangement
Fourth Step
P_2 P_3
10 9 8 12 11 13 17 14 15 16
after local rearrangement
P_0 P_1 P_2 P_3 P_4
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
Solution