Домашняя работа №2 по алгоритмам. Кривенко Андрей. M3107

December 19, 2021

1. Для нахождения числа инверсий в массиве мы можем использовать модифицированную сортировку слиянием. Во время процедуры слияния двух массивов в один, если элемент из левого массива больше элемента из правого массива, то к общему числу инверсий нужно прибавить $m - 1$, где m - указатель на начало правого массива, а i - указатель на рассматриваемый элемент левого массива.

2. Если в сортирвке слиянием использовать нестрогое сравнение при процедуре слияния двух массивов в один, то сортировка будет устойчивой. Например, когда мы во время этой процедуры сравниваем два равных элемента, в финальный массив первым будет добавлен то, который находится в левом массиве, т.е. тот, который, расположен раньше в исходном массиве.

3. Для сортировки слиянием без рекурсии, необходимо разбивать исходный массив через цикл на части, и сразу же их сливать: сначала размера 1, затем 2, затем 4, и т.д. Для этого просто использовать два for:

```
// n - array size
for (int i = 1; i < n; i *= 2) {
  for (int j = 0; j < n; j += 2 * i) {
    // now do merge with: l = j, mid = min(i + j, n), r = min(i * 2 + j, n)
  }
}
```

4. (a) $4, 3, 2, 1, 0$
   (b) $0, 1, 2, 3, 4$
   (c) $1, 3, 4, 2, 0$

5. В среднем случае будет использоваться $\log n$ дополнительной памяти, так как в среднем функция будет вызвана $\log n$ раз, в худшем же - n раз и n дополнительной памяти.

6. Так как быстрая сортировка на каждом шаге вставляет элемент на нужную позицию, достаточно просто брать финальный индекс эл-та на каждом шаге, так как все элементы меньше него будут расположены в левом подмассиве, длинна которого как раз будет зависеть от индекса элемента.

7.

8. Достаточно сначала взять массив размера k, а затем заполнить его числом вхождений элементов в изначальный массив(то есть выполнить часть сортировки подсчетом). А затем выполнить для каждой ячейки массива: $arr[i] = arr[i] + arr[i - 1]$. Это будет работать за $O(n + k)$. Затем сделать для этой структуры функцию, которая принимает a и b и возвращает $arr[b] - arr[a]$

9. Не совсем понятен вопрос. Требуется количество сортировок или? Если же в пределах одной сортировки, то это зависит от количества равных элементов массива.

10. запустить рекурсивнй алгоритм для каждой из точек:

```
int resultDistance = INFTY;
int result;
for (int i = 0; i < n; ++i) {
```

```
  int tmp = 0;
  for (int j = 0; j < n; ++j) {
    tmp += abs(arr[i] - arr[j]);
  }
  if (tmp < resultDistance) {
    result = i;
    resultDistance = tmp;
  }
}
```