Лабораторная 6. Слуцкий Никита, гр. 053501

(3 балла) Найдите 10 чисел, у которых ровно k делителей. Необходимо предоставить эти 10 чисел и описание решения (можно и само решение), с помощью которого вы их находили. Если 10 таких чисел не существует, необходимо это сообщить.

Решение:

Искомые числа составляю из простых множителей. Каждое моё число представляет степень некоторого простого числа. Таким образом можно легко считать количество делителей таких чисел, точно зная, что при факторизации исходное число представляет из себя какой-то простой множитель в какой-то степени.

Число, составленное перемножением *двух* простых чисел, делится на: единицу, само себя, это простое число (*3 делителя*). Число, составленное из *трёх* простых чисел, делится на единицу, само себя, простое число и простое число в квадрате (*4 делителя*) (Например, число 7 * 7 * 7 = 343 делится на [1, 343, 7, 7^2 = 49]). Число, составленное из *четырёх* простых чисел, делится на единицу, само себя, простой множитель, простой множитель во 2 и 3 степенях (*5 делителей*) (Например, число 81 = 3 * 3 * 3 * 3 делится на [1, 81, 3, 9, 27])

Таким образом я могу выявить закономерность, что если число должно иметь ровно k делителей, оно может быть представлено в виде p^(k-1), где p – любое простое число. Сложность задачи заключается в поиске простых чисел и их перемножении. 10 простых чисел можно записать в константный массив, чтобы не просчитывать каждый раз, а можно каждый раз искать новые числа на каком-либо отрезке [L..R], зная, что там гарантированно существуют 10 простых чисел. (Поиск простых чисел занимает (R – L) * Sqrt(R)).

Составление 10 простых чисел:

```
leftBorder = 2
 while (k < 10) {
     i = leftBorder
     while (true) {
         if (isSimple(i) == true) {
             arraySimple.push(i)
             leftBorder = i + 1
             k++
             break
         }
         i++
     }
 }
```

Составление искомых чисел – это возведение полученных простых в степень (k – 1).

Для моего номера в журнале k = 23 сгенерированный **ответ на задачу**:

4194304 ; 31381059609 ; 2384185791015625 ; 3909821048582988049 ; 81402749386839761113321 ; 3211838877954855105157369 ; 1174562876521148458974062689 ; 13569980418174090907801371961 ; 907846434775996175406740561329 ; 148852438543083302439338564577241

2) (2 балла) Постройте строку из 100 символов, которая состоит из k различных символов. Можно описать алгоритм построения или прикрепить файл с решением (объяснения решения тоже необходимы).

Решение:

```
const int k = 23;
for (int i = 0; i < 100; i++)
    cout<<(static_cast<char>(i % k + 40));
```

Иду в цикле от одного до 100 и вывожу символ (преобразовывая счётчик / код в символьный тип). % k обеспечивает, что код символа никогда не станет больше, чем k. И он будет бегать от 0 до k. Также я добавил смещение на 40, например, так как в первых 23 символах ASCII есть символ переноса строки (чтобы убрать путающий фактор). Для моего номера в журнале k = 23 сгенерированный **ответ на задачу**: ()*+,-./0123456789:;<=>()*+,-./0123456789:;<=>()*+,-./0123456789:;<=>()*+,-./0123456789:;<=>()*+,-./

3) (2 балла) Постройте строку из 2k + 20 скобок, являющейся правильной скобочной последовательностью. Можно описать алгоритм построения или прикрепить файл с решением (объяснения решения тоже необходимы).

Решение:

Можно поступить просто и быстро: вывести вначале k + 10 открывающихся скобок, а затем k + 10 закрывающихся скобок => это будет элементарная правильная скобочная последовательность (ПСП) . Если генерировать абсолютно случайную последовательность, воспользуюсь правилами ПСП: 1) любая ПСП в скобках – это ПСП ((ПСП) => ПСП); 2) ПСП + ПСП => ПСП.

Я работаю с массивом символов, начинаю работать с середины (это точка старта, две позиции: L = 0.5 * length -1, R = 0.5 * length + 1). В цикле от 0 до k + 10 рандомом выбираю: построить ПСП (справа или слева, можно решать на этапе выбора) или обернуть существующую ПСП в скобки. L и R будут контролировать это. Например, у меня есть уже строка ( ) и на очередной итерации рандомно выбирается, сделать ли из неё (( )) или ( )( ). Естественно, массив лучше заводить длиной что-то около 2k + 20 + 4 (чтобы по краям хватило в случае чего), а потом сделать .Trim( ) => убрать пробелы в начале и в конце. Таким образом получится очень разнообразная и непредсказуемая ПСП. Для моего номера в журнале k = 23 сгенерированный **ответ на задачу**:

(((((((((((((((((((((((((((((((((())()())()()()())))))))())())()())()())()()))())())())))())))()))))())()()))))))()())()()()
()

Последовательность правильная. Проверено программой из ЛР №4 (проверка описана в файле ЛР №4, если необходимо)

4) (3 балла) Постройте такую перестановку размера n = 10, в которой будет ровно k инверсий. **Количество инверсий (беспорядка) в перестановке** – это количество пар элементов (не обязательно соседних), в которых следующий элемент имеет меньший номер, чем предыдущий. Перестановка размера n - последовательность из n натуральных

чисел, в которой каждое число от 1 до n встречается ровно один раз. Можно описать алгоритм построения или прикрепить файл с решением (объяснения решения тоже необходимы).

Решение:

Количество инверсий считаю за O (N^2 / 2). Следующий код работает по определению.

```
for (int i = 0; i < array.size() - 1; i++) {
    for (int j = i + 1; j < array.size(); j++) {
        if (array[i] > array[j])
            k_inverse++;
    }
}
```

Далее мне необходимо сгенерировать все перестановки и на каждой итерации проверять количество инверсий. Это можно сделать с помощью сортировки. Исходный массив отсортирован по возрастанию. Сортирую его по убыванию, на каждой итерации проверяю количество. Сортирую пузырьком, за N^2, чтобы получить ВСЕ промежуточные стадии между состояниями отсортированности по убыванию и возрастанию, поэтому итоговая сложность O(N^2). Для моего номера в журнале k = 23 ответ на задачу: 4 5 6 7 8 9 3 10 2 1