ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Отчёт по лабораторной работе №7
«Жадные алгоритмы. Задание 321. Create Maximum Number»

Выполнила работу:
Абаянцева Евгения
Академическая группа №J3110
Принято:
...

Санкт-Петербург
2024 г.

Введение

Цель – написать код с использованием жадного алгоритма по задаче с сайта LeetCode.

Задача: вам даны два целочисленных массива nums1 и nums2 длины m и n соответственно. Массивы nums1 и nums2 представляют в себе цифры двух чисел. Вам также дано целое число k.

Создайте максимальный массив из чисел длины k <= m + n из массива цифр двух массивов. Относительный порядок цифр из одного массива должен быть сохранен.

Верните массив цифр k, представляющих ответ.

Теоретическая подготовка

Данная задача предполагает решение через жадные алгоритмы. Задача выбирает на каждом шагу более оптимальный вариант, чем был на предыдущем (в случае задачи 321 – выбор того массива, где находятся более максимальные цифры). Использованные типы данных для этой задачи – целые числа (тип int), массив чисел (тип хранения чисел vector, тип самих чисел int).

Реализация

Для задачи 321 требуются следующие библиотеки:

1. Библиотека iostream – эта основная библиотека C++ для ввода и вывода текста с консоли;
2. Библиотека vector – эта библиотека определяет вектор шаблона класса контейнера и несколько вспомогательных шаблонов;
3. Библиотека algorithm – эта библиотека определяет функции шаблона контейнера стандартной библиотеки C++, которые выполняют алгоритмы.

Реализация была сделана путём создания трёх функций в классе Solution, представленнных ниже: функции maxnumber, функции maxnums и функции mergenums. Каждая из функций использовалась на каждом ходу пробежки цикла.

Рассмотрим каждую из них.

Представленная функция maxnumber была реализована для пробежки цикла, которая на каждом шагу сравнивает массивы чисел и выбирает тот, в котором последовательность чисел больше:

```
vector<int> maxnumber(vector<int>& nums1, vector<int>& nums2, int k, int& coun)
    {
    int m = nums1.size(), n = nums2.size();
    vector<int> nums;
    for (int i = max(0, k - n); i <= min(k, m); i++) {
        nums = max(nums, mergenums(maxnums(nums1, i, coun), maxnums(nums2, k - i
    , coun), coun), coun));
    }

    coun += sizeof(m) + sizeof(n) + sizeof(nums);
    cout << coun << ''bt\n'';
    return nums;
}
```

Листинг 1 – функция maxnumber

Представленная функция maxnums была реализована для создания массива, который добавляет в него числа переданного массива:

```
vector<int> maxnums(vector<int>& nums, int k, int &coun) {
    int i = (int)nums.size() - k;
    vector<int> end_nums;
    for (int num : nums) {
        while (i > 0 && end_nums.size() > 0 && end_nums.back() < num) {
            end_nums.pop_back();
            i--;
        }
        end_nums.push_back(num);
    }
    end_nums.resize(k);

    coun += sizeof(i) + sizeof(end_nums);
    cout << coun << ''bt\n'';
    return end_nums;
}
```

Листинг 2 – функция maxnums

Представленная функция maxnumber была реализована для выбора максимального массива из переданных двух:

```
vector<int> mergenums(vector<int> nums1, vector<int> nums2, int& coun) {
    vector<int> nums;
    while ( nums1.size() > 0 || nums2.size() > 0 ) {
        if (nums1 > nums2) {
            vector<int> &mas = nums1;
            nums.push_back(mas[0]);
            mas.erase(mas.begin());
        }
        else {
            vector<int> &mas = nums2;
            nums.push_back(mas[0]);
            mas.erase(mas.begin());
        }

    }
```

```
16
17      coun += sizeof(nums);
18      cout << coun << ''bt\n'';
19      return nums;
20  }
```

Листинг 3 – функция mergenums

В конце выдаётся массив, который содержит максимальные цифры массива, не нарушая исходного порядка в двух переданных массивах.

Особенность реализации: сравнение двух массивов, происходящая на каждом шагу алгоритма.

Экспериментальная часть

1. Подсчёт памяти: два массива 16 байт, один int – 4 байта. Вызов k раз (на запускаемом тесте пробежка составляет заданное k, которое равняется 5 раз) с подсчётом памяти каждой из трёх функций (в сумме три массива в каждой функции – 24 байта, два инта в двух – 8 байтов). Всего в сумме, учитывая возвращенный массив и сам счётчик занимаемой памяти – 361 байт.

2. Асимптотика: $O(k \cdot (m + n))$.

Заключение

В ходе выполнения этой задачи был реализован жадный алгоритм. Цель работы была достигнута путём проверки кода на сайте LeetCode.

В качестве дальнейшего исследования можно предложить оптимизацию алгоритма с точки зрения уменьшения требуемой памяти и времени на исполнение алгоритма.

Приложения

```
1  #include <iostream>
2  #include <vector>
3  #include <algorithm>
4
5  using namespace std;
6
7  class Solution {
8  public:
9    vector<int> maxnumber(vector<int>& nums1, vector<int>& nums2, int k, int& coun
     ) {
10     int m = nums1.size(), n = nums2.size();
11     vector<int> nums;
12     for (int i = max(0, k - n); i <= min(k, m); i++) {
13       nums = max(nums, mergenums(maxnums(nums1, i, coun), maxnums(nums2, k - i,
```

```
    coun), coun));
    }

    coun += sizeof(m) + sizeof(n) + sizeof(nums);
    cout << coun << "bt\n";
    return nums;
  }

  vector<int> maxnums(vector<int>& nums, int k, int &coun) {
    int i = (int)nums.size() - k;
    vector<int> end_nums;
    for (int num : nums) {
      while (i > 0 && end_nums.size() > 0 && end_nums.back() < num) {
        end_nums.pop_back();
        i--;
      }
      end_nums.push_back(num);
    }
    end_nums.resize(k);

    coun += sizeof(i) + sizeof(end_nums);
    cout << coun << "bt\n";
    return end_nums;
  }

  vector<int> mergenums(vector<int> nums1, vector<int> nums2, int& coun) {
    vector<int> nums;
    while ( nums1.size() > 0 || nums2.size() > 0 ) {
      if (nums1 > nums2) {
        vector<int> &mas = nums1;
        nums.push_back(mas[0]);
        mas.erase(mas.begin());
      }
      else {
        vector<int> &mas = nums2;
        nums.push_back(mas[0]);
        mas.erase(mas.begin());
      }

    }

    coun += sizeof(nums);
    cout << coun << "bt\n";
    return nums;
  }
};

int main() {
  vector<int> nums1 = { 3, 4, 6, 5 };
  vector<int> nums2 = { 9, 1, 2, 5, 8, 3 };
  int k = 5;

  int coun = sizeof(nums1) + sizeof(nums2) + sizeof(k);
  cout << sizeof(nums1) + sizeof(nums2) << "\n";

  Solution x;
  vector<int> result = x.maxnumber(nums1, nums2, k, coun);

  coun += sizeof(result) + sizeof(x);

```

```
73
74    for (int num : result) {
75      cout << num << ” ”;
76    }
77
78    cout << ”\n” << coun + sizeof(coun) << ”bt”;
79    return 0;
80  }
```

Листинг 4 – код файла lab-7.cpp

Прохождение всех тестов на сайте LeetCode:

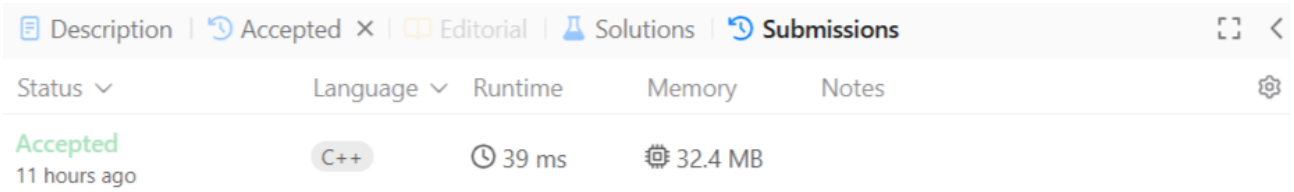

Рис. 1 – основные тесты