Московский авиационный институт (национальный исследовательский университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работа №1 по курсу «Дискретный анализ»

Студент: А. А. Почечура
Преподаватель: А. А. Кухтичев
Группа: М8О-206Б

Дата:
Оценка:
Подпись:

Москва, 2022

Лабораторная работа №1

Задача: Требуется разработать программу, осуществляющую ввод пар «ключ-значение», их упорядочивание по возрастанию ключа указанным алгоритмом сортировки за линейное время и вывод отсортированной последовательности.

Вариант сортировки: Поразрядная сортировка.

Вариант ключа: `телефонные номера, с кодами стран и городов в формате +<код страны> <код города> телефон.`

Вариант значения: `числа от 0 до` $2^{64} - 1$.

1 Описание

Требуется написать реализацию алгоритма поразрядной сортировки. Идея её заключаются в том, что элементы сортируются по разрядам, начиная с первого. Проходя по разрядам элементов (сначала по первому разряду всех элементов, затем по второму и т.д.), мы сортируем цифры элементов текущего разряда, используя сортировку подсчётом. Получив массив, в котором число x на позиции i определяет количество элементов, у которых на текущем разряде стоит цифра, большая или равная i, мы расставляем наши элементы в новый массив согласно правилам сортировки подсчётом. Полученный новый массив «свайпаем» с нашим старым массивом и проходимся по алгоритму заново, пока все разряды каждого элемента не будут пройдены.

2 Исходный код

Элементы я решил хранить в массиве пар $elems$, у которых на первой позиции находится статический массив $char'$ов с размером 31 (с запасом) для ключа, а на второй позиции $unsigned\ long\ long$, который без труда сможет поместить в себя позицию значения элемента. Ключ сначала вводим в строку, определяем её размер и с помощью этого загоняем наш ключ в массив, начиная с конца массива. Для удобства, в первую позицию массива запишем размер ключа элемента. Также при вводе определяем максимальный размер ключа в переменной max_num. После этого подготавливаем всё для самой сортировки: массив tmp, который будет хранить количество элементов с конкретной цифрой на текущем разряде, переменна cnt, в которую мы будем записывать цифру ключа элемента на текущем разряде, и массив $elems1$, в который мы будем расставлять элементы на соответствующие позиции и затем «свайпать» с начальным массивом. Массив tmp изначально обнуляем с помощью функции $PutZeroes$. Проходимся по позициям каждого ключа, начиная с первого разряда и заканчивая разрядом, соответствующим максимальному размеру ключа max_num. Заполняем массив tmp, если вдруг наш ключ на данном разряде уже закончился, или cnt меньше нуля при переводе из $char$ в int (в случаях «-» и «+»), берём cnt как ноль. Складываем постфиксно все элементы. Теперь в массиве tmp число x на позиции i определяет количество элементов, у которых на текущем разряде стоит цифра, большая или равная i. Теперь снова проходимся по массиву $elems$, и расставляем наши элементы с помощью массива tmp и переменной cnt в массив $elems1$ на нужную позицию. «Свайпаем» массивы, проставляем в tmp нули и начинаем итерацию заново. В конце выводим элементы изменённого масиива $elems$.

```
1  #include <iostream>
2  #include <math.h>
3  #include <string>
4  #include <vector>
5  #include <algorithm>
6  #include <map>
7  #include <queue>
8  #include <stack>
9  #include <set>
10 #include <list>
11
12 using namespace std;
13
14 void PutZeroes(int* a, int n){
15     for(int i = 0 ; i < n ; i++){
16         a[i] = 0;
17     }
18 }
19
20 int main() {
```

```
21      ios::sync_with_stdio(false);
22      cin.tie(nullptr); cout.tie(nullptr);
23      int size = 31;
24      pair<char[31], unsigned long long> Pair;
25      vector<pair<char[31], unsigned long long>> elems;
26      string key;
27      auto max_num = key.size();
28      while(cin >> key >> Pair.second){
29          for(int i = size - key.size() ; i < size ; i++){
30              Pair.first[i] = key[key.size() - (size - i)];
31          }
32          Pair.first[0] = key.size();
33          max_num = max(key.size(), max_num);
34          elems.push_back(Pair);
35      }
36      int cnt;
37      vector<pair<char[31], unsigned long long>> elems1(elems.size());
38      int tmp[10];
39      PutZeroes(tmp, 10);
40      for(int i = 30 ; i > 30 - max_num ; i--){
41          for(int j = 0 ; j < elems.size() ; j++){
42              cnt = elems[j].first[i] - '0';
43              if(cnt < 0 || elems[j].first[0] < (31 - i)){
44                 cnt = 0;
45              }
46              tmp[cnt]++;
47          }
48          for(int j = 1 ; j < 10 ; j++){
49              tmp[j] = tmp[j] + tmp[j - 1];
50          }
51          for(int j = elems.size() - 1 ; j > -1 ; j--){
52              cnt = elems[j].first[i] - '0';
53              if(cnt < 0 || elems[j].first[0] < (31 - i)){
54                 cnt = 0;
55              }
56              elems1[tmp[cnt] - 1].first[0] = elems[j].first[0];
57              for(int k = 31 - elems[j].first[0] ; k < 31 ; k++){
58                  elems1[tmp[cnt] - 1].first[k] = elems[j].first[k];
59              }
60              elems1[tmp[cnt] - 1].second = elems[j].second;
61              tmp[cnt]--;
62          }
63          swap(elems, elems1);
64          PutZeroes(tmp, 10);
65      }
66      for(int i = 0 ; i < elems.size() ; i++){
67          for(int j = 31 - elems[i].first[0] ; j < 31 ; j++){
68              cout << elems[i].first[j];
69          }
```

```
70          cout << '\t' << elems[i].second << '\n';
71      }
72  }
```

3 Консоль

```
root@DESKTOP-5HM2HTK:~# g++ -pedantic lab1.cpp
root@DESKTOP-5HM2HTK:~# cat test1
+7-495-1123212  13207862122685464576
+375-123-1234567        7670388314707853312
+7-495-1123212  4588010303972900864
+375-123-1234567        12992997081104908288
root@DESKTOP-5HM2HTK:~# ./a.out <test1
+7-495-1123212  13207862122685464576
+7-495-1123212  4588010303972900864
+375-123-1234567        7670388314707853312
+375-123-1234567        12992997081104908288
```

4 Тест производительности

```
root@DESKTOP-5HM2HTK:~# g++ benchmark.cpp
root@DESKTOP-5HM2HTK:~# ./a.out <tests/1.t
Count of lines is 100000
Radix sort time: 318345us
STL stable sort time: 2570013us
```

Как видно, сортировка подсчётом работает в разы быстрее встроенной сортировки, так как сложность сортировки подсчётом - O(максимальное количесво разрядов ключа * количество элементов), а сложность встроенной сортировки - O(количество элементов * логарифм от количества элементов). При большом количестве элементов максимальное количество разрядов ключа значительно меньше логарифма от количества элементов, что значительно играет роль.

5 Выводы

Выполнив первую лабораторную работу по курсу «Дискретный анализ», я научился писать сортировки за линейное время, что может пригодиться мне при выполнении многих задач. Также, очень полезным была практика написания генератора случайных тестов на языке $C++$, что улучшило мои навыки работы с файлами на данном языке. $Benchmark$ - очень удобный способ для сравнения двух алгоритмов, который тоже мной освоен и в будущем будет использоваться.

Список литературы

[1] Томас Х. Кормен, Чарльз И. Лейзерсон, Рональд Л. Ривест, Клиффорд Штайн. *Алгоритмы: построение и анализ, 2-е издание.* -- Издательский дом «Вильямс», 2007. Перевод с английского: И. В. Красиков, Н. А. Орехова, В. Н. Романов. -- 1296 с. (ISBN 5-8459-0857-4 (рус.))

[2] *Поразрядная сортировка - Википедия.* URL: https://ru.wikipedia.org/wiki/Поразрядная_сортировка (дата обращения: 17.03.2022).