Лабораторная работа № 1 по курсу дискретного анализа: сортировка за линейное время

Выполнил студент группы М8о-207Б-18 МАИ *Токарев Никита*.

Условие

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

2. Поразрядная сортировка.Тип ключа: телефонные номера, с кодами стран и городов в формате +<код страны> <код города> телефон.Тип значения: строки переменной длины (до 2048 символов)..

Метод решения

После инициализации переменных и выделения памяти:

1. Считываю ключ и значение в формат char*.

2. Так как ключ – мобильный номер, а сортировка поразрядная, перевожу ключ в unsigned long long и также подсчитываю максимальное число разрядов.

3. Когда все данные были считаны, вызываю функцию поразрядной сортировки.

4. В функции поразрядной сортировки нахожу последний разряд для каждого ключа и вызываю сортировку подсчетом.

5. Действую так(см п.4), пока не достигну максимального числа разрядов.

6. Печатаю и результат и освобождаю память,выделенную ранее.

Описание программы

Данные хранил в структуре, память выделял с помощью оператора new.

```
struct_data type {
    unsigned long long number;//номер в числовом типе
    char *value;//значение
    char *key;//ключ(мобильный номер)
};
struct table {
    data_type *dt;//мой тип данных
    size_t size;//размер таблицы
    size_t m_ln;//максимальное число разрядов
```

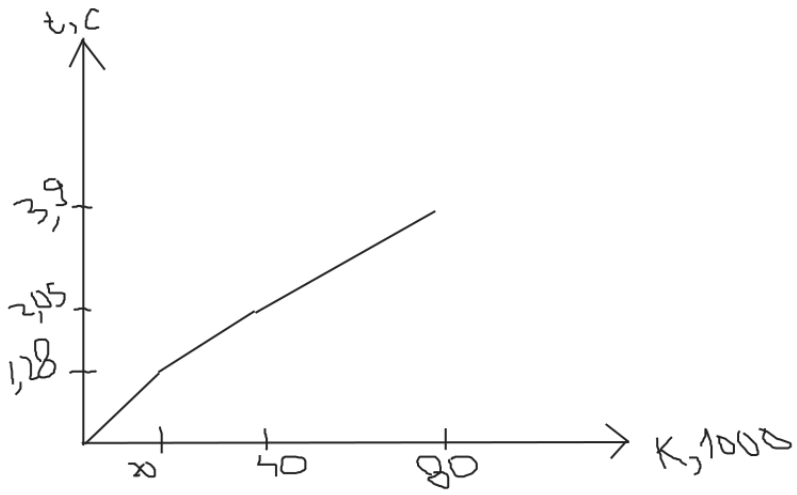
};
В функции int main я сперва выделяю память для структуры table, затем, выделяю 100 ячеек для hesh->dt, а также для своих значений char* value и char *value с нулевым индексом. Потом запускается цикл while с условием: scanf(«%s %s », hesh->dt[i].key, hesh->dt[i].value) != EOF.
В данном цикле считывая нужные значения запускается функция To_full(table *hesh, unsigned long long n,char *c);. В данной функции значение char *key конвертируется в переменную unsigned long long и высчитывается количество разрядов. Максимальное число разрядов сохраняется в переменную m_ln, а полученная числовая переменная в unsigned long long hesh->dt[i].number[n].
Также когда индекс добавляемого элемента является предельным для моей структуры, запускается функция table *Resize(table *hesh,size_t size), которая создает новою структуру с большим размером, чем у сруктуры, являющейся агрументом в int main. Затем происходит копирование значений в новую структуру и возвращается указатель на новую структуру, а структура являющаяся аргументом в int main, удаляется.

Пример вызова функции Resize: hesh = Resize(hesh,hesh->size * 2);
После того,как был достигнут конец файла вызывается функция Radix_sort(table *hesh), а затем печатается результат с последовательным удалением char* value и char* key.
Radix_sort:
for(size_t i = 0;i < hesh->m_ln; i++) {
for(size_t j = 0; j < hesh->size; j++) {
a[j] = hesh->dt[j].number % 10; //добавление разрядов в массив
if(max < a[j]) { //поиск максимального числа,среди добавленных в массив a
max = a[j];
}
hesh->dt[j].number = hesh->dt[j].number / 10; //сокращение номера на 10, чтобы добраться до следующего разряда
}
hesh = Counting_sort(hesh,a,max); //вызов сортировки подсчетом
max = 0;
}
В сортировке подсчетом,я создаю нулевой массив размером max + 1, затем прохожу по массиву a и в позицию c[a[i]](i от нуля до size,где size размер массива a) добавляю единицу. Затем c[i] = c[i] + c[i - 1](i от 1 до max + 1). Далее я создаю новую структуру(new_hesh) и
(от size - 1 по 0) {
new_hesh->dt[c[a[i]] - 1] = hesh->dt[i];
c[a[i]]–;
}
после окончания данного цикла возвращаю указатель на новую структуру(new_hesh).

Дневник отладки

1. Ошибки компиляции -> исправлял согласно стандартам C++;
2. Ошибка выполнения -> изменил размер char *value на [2049],т.к value[2048] – нулевой элемент;
3. Неправильный ответ в 6-м тесте -> начал выводить номер с ведущими нулями;
4. Превышено реальное время работы -> начал использовать char*, scanf(), отключил синхронизацию iostream с stdio;

Тест производительности



Пояснения к графику:

1. За 1.28с сортируется 20000 строк
2. За 2.05с сортируется 40000 строк
3. За 3.9с сортируется 80000 строк

Строка включается в себя ключ + значение. Исходя из графика(см. выше), можно заметить,что реализованная сортировка,является линейной.

Выводы

Пожалуй, начну с оценки сложности моей сортировки. Сложность = O(k*(n + m + m + n + n)) = O(k*(3n)),k – максимальное число разрядов, n – количество сортируемых элементов, также при больших значениях можно пренебречь m, так как максимальное значение m = 10. С помощью поразрядной сортировки можно также сортировать различные типы данных. Например мобильные номера,автомобильные номера и тд. Также

поразрядная сортировка является устойчивой, то есть сортировка, которая не меняет относительный порядок сортируемых элементов, имеющих одинаковые ключи. В ходе работы встречался с такими проблемами,как утечки памяти. Выделяя память для следующего элемента в цикле while с условием: scanf(«%s %s », hesh->dt[i].key, hesh->dt[i].value) != EOF, случалось, что следующего элемента не было. Также какое-то время я представлял элемент hesh->dt[i]->number типом int,от чего происходил выход за пределы int и возвращались неккоректные данные. Также хотелось бы отметить, что в данной работе не использовался класс vector, так как удалось удалось реализовать необходимый тип данных с помощью структуры.