Университет ИТМО

Факультет программной инженерии и компьютерной техники

**Лабораторная работа №2**

по «Алгоритмам и структурам данных»

Базовые задачи

Выполнил:

Студент группы P3233

Анисимов М. Д.

Преподаватели:

Косяков М.С.

Тараканов Д.С.

Санкт-Петербург

2024

**Yandex contest**

**Задача E «Коровы в стойла»**

Пояснение к применённому алгоритму:

Задача решается через бинарный поиск элементов. С начала цикла while мы находим середину координат нашего отрезка со стойлами (значение centerArray). Затем между координатами centerArray и endArray (Правая часть отрезка) ищем доступные стойла, в которую можно поставить корову. С каждой новой итерацией мы пошагово увеличиваем отрезок centerArray-endArray на половину от правого отрезка. Как только все коровы окажутся в стойлах, программа начинает двигать значение startArray, пока условие startArray + 1 <= endArray удовлетворительно. Таким образом startArray и будет являться максимальным расстоянием между минимальными

Код работает за O(n \* log m + n) = O(n \* log m)

Память: О(n)

Код программы

#include <iostream>  
using namespace std;  
  
void cowsInStallsFunc(int stallsNumber, int cowsNumber, int inputStalls[]) {  
 int startArray = 0;  
 int endArray = inputStalls[stallsNumber - 1];  
 int centerArray = 0;  
  
 while (startArray +1 <= endArray) {  
 centerArray = (startArray + endArray) / 2;  
 int filledStalls = 1;  
 int lastFilledStalls = inputStalls[0];  
 for (int i = 0; i < stallsNumber; i++) {  
 if (centerArray + lastFilledStalls > inputStalls[i])  
 continue;  
 else {  
 lastFilledStalls = inputStalls[i];  
 filledStalls++;  
 }  
 }  
 if (filledStalls >= cowsNumber) {  
 startArray = centerArray + 1;  
 } else  
 endArray = centerArray;  
 }  
 cout << startArray - 1;  
}  
int main() {  
 int stallsNumber, cowsNumber;  
 cin >> stallsNumber >> cowsNumber;  
 int inputNumbers[stallsNumber];  
  
 for (int i = 0; i < stallsNumber; i++) {  
 cin >> inputNumbers[i];  
 }  
 cowsInStallsFunc(stallsNumber, cowsNumber, inputNumbers);  
 return 0;  
}

**Задача F «Число»**

Пояснение к применённому алгоритму:

Алгоритм состоит в том, что программа проходится по введённым с клавиатуры числами и проверяет, какое число больше в числовом эквиваленте:

Текущий элемент строки + следующий элемент или следующий элемент + текущий элемент. Если первый вариант оказался меньше, то программа меняет местами текущий и следующий элемент строки. В конце каждой итерации наименьшая цифра оказывается в конце, поэтому с каждым циклом программа проверяет на один элемент меньше, чем в предыдущем цикле

Код работает за O(n \* n + n) = O(n2)

Память: О(n)

Код программы:

#include <iostream>  
#include <fstream>  
#include <deque>  
using namespace std;  
  
void numberFunc(deque<string> inputNumbers){  
 string result = "";  
 int integer\_result = 0;  
 for(int i = inputNumbers.size() - 1; i >= 0; --i){  
 for(int j=0; j < i; ++j){  
 if(inputNumbers.at(j) + inputNumbers.at(j + 1) < inputNumbers.at(j + 1) + inputNumbers.at(j))  
 swap(inputNumbers[j], inputNumbers[j+1]);  
 }  
  
 }  
 for(int i = 0; i< inputNumbers.size(); i++) {  
 result += inputNumbers[i];  
 }  
 cout<<result;  
}  
  
int main() {  
 string enterLine;  
 deque<string> inputNumbers;  
  
 while(cin >> enterLine && enterLine != "#") inputNumbers.push\_back(enterLine);  
 numberFunc(inputNumbers);  
 return 0;  
}

**Задача G «Кошмар в замке»**

Пояснение к применённому алгоритму:

Алгоритм заключается в том, что сперва происходит сортировка веса латинских символов в порядке возрастания. Отсортированный вес всех латинских символов заносится в вектор symbolsWeight, который хранит пару значений – символ и его вес. Далее следует цикл, который подсчитывает сколько раз буквы повторяются в строке. Затем программа выполняет ещё цикл, где пары элементов размещаются по краям строк, а элементы без пар заполняют оставшуюся середину строки. Последним этапом является вывод строки на экран

Код работает за O(n2)

Память: О(n2)

Код программы:

#include <iostream>  
#include <map>  
#include <algorithm>  
#include <vector>  
using namespace std;  
bool comparator (const pair<char, int>& a, const pair<char, int>& b) {  
return a.second > b.second;  
}  
void castleNightmare(string line, int inputNumbers[]) {  
 map<char, int> resultMap;  
 vector<pair<char, int>> symbolsWeight;  
 int firstElement = 0;  
 int lastElement = line.length() - 1;  
 int counterElements = 0;  
 char resultArray[line.length()];  
 symbolsWeight.reserve(26);  
for (int i = 0; i < 26; i++) {  
 symbolsWeight.emplace\_back(char(i + 97), inputNumbers[i]);  
 }  
  
 sort(symbolsWeight.begin(), symbolsWeight.end(), comparator);  
 for (int i = 0; i < line.length(); i++) {  
 resultMap[line.at(i)]++;  
 if (resultMap[line.at(i)] == 2) {  
 counterElements++;  
 }  
 }  
  
 for(int i=0; i<symbolsWeight.size(); i++){  
 if(resultMap[symbolsWeight[i].first] > 1){  
 resultArray[firstElement] = symbolsWeight[i].first;  
 resultArray[lastElement] = symbolsWeight[i].first;  
 firstElement++;  
 lastElement--;  
 resultMap[symbolsWeight[i].first] -= 2;  
 }  
 while(resultMap[symbolsWeight[i].first] !=0){  
 resultArray[counterElements] = symbolsWeight[i].first;  
 resultMap[symbolsWeight[i].first]--;  
 counterElements++;  
 }  
 }  
 for(int i=0; i<sizeof(resultArray); i++){  
 cout<<resultArray[i];  
 }  
}  
int main() {  
 string line;  
 int inputNumbers[26];  
 cin>>line;  
 for(int i=0; i<26; i++){  
 cin>>inputNumbers[i];  
 }  
 castleNightmare(line, inputNumbers);  
 return 0;  
}

**Задача H «Магазин»**

Пояснение к применённому алгоритму:

Алгоритм функционирует при помощи стандартной реализации сортировки слиянием. Цены товаров сортируются в порядке убывания. Далее следует цикл, при помощи которого мы проходимся п отсортированному массиву цен и суммируем стоимость товаров, кроме тех, что идут бесплатно по условию акции

Код работает за O(n \* log n)

Память: О(n)

#include <iostream>  
using namespace std;  
void merge(int list[],int start, int end, int mid, int arraySize);  
  
void mergeSort(int list[], int start, int end, int arraySize)  
{  
 int mid;  
 if (start < end){  
  
 mid=(start+end)/2;  
 mergeSort(list, start, mid, arraySize);  
 mergeSort(list, mid+1, end, arraySize);  
 merge(list,start,end,mid, arraySize);  
 }  
}  
  
void merge(int list[],int start, int end, int mid, int arraySize)  
{  
 int mergedList[arraySize];  
 int i, j, k;  
 i = start;  
 k = start;  
 j = mid + 1;  
  
 while (i <= mid && j <= end) {  
 if (list[i] > list[j]) {  
 mergedList[k] = list[i];  
 k++;  
 i++;  
 }  
 else {  
 mergedList[k] = list[j];  
 k++;  
 j++;  
 }  
 }  
  
 while (i <= mid) {  
 mergedList[k] = list[i];  
 k++;  
 i++;  
 }  
  
 while (j <= end) {  
 mergedList[k] = list[j];  
 k++;  
 j++;  
 }  
  
 for (i = start; i < k; i++) {  
 list[i] = mergedList[i];  
 }  
}  
  
int main() {  
 int summary = 0;  
 int product = 0;  
 int freeProduct = 0;  
 cin >> product >> freeProduct;  
 int inputNumbers[product];  
  
 for(int i=0; i<product; i++) {  
 cin >> inputNumbers[i];  
 }  
 mergeSort(inputNumbers, 0, product - 1, product);  
 for(int i = 0; i< product; i++){  
 if((i + 1) % freeProduct != 0 || (i+1) < freeProduct || ((product - 1) - (i) < freeProduct && (i + 1) % freeProduct != 0)) summary = summary + inputNumbers[i];  
 }  
 cout<< summary;  
 return 0;  
}