Университет ИТМО

Факультет программной инженерии и компьютерной техники

Лабораторная работа №2

по «Алгоритмам и структурам данных»

Базовые задачи

Выполнил:

Студент группы P3233

Фамилия И.О.

Преподаватели:

Косяков М.С.

Тараканов Д.С.

Санкт-Петербург

2024

Задача №1 «E.Коровы в стойла»

Пояснение к примененному алгоритму:

Данный код решает задачу поиска наименьшего возможного значения `ans`, при котором можно разместить `k` объектов на прямой так, чтобы расстояние между любыми двумя объектами было не меньше `ans`.

Алгоритм использует бинарный поиск для нахождения значения `ans`. Изначально устанавливаются границы диапазона поиска `l` (нижняя граница) и `r` (верхняя граница). Внутри цикла бинарного поиска, на каждой итерации берется значение `tmp`, равное середине текущего диапазона `l` и `r`. Функция `check` вызывается для проверки, можно ли разместить `k` объектов с расстоянием не меньше `tmp`. Если размещение возможно, то значение `tmp` становится новой нижней границей `l`, иначе `tmp` становится новой верхней границей `r`.

Функция `check` принимает массив `arr`, число `k` и текущее предполагаемое расстояние `ans`. Начиная с первого элемента массива, она итерируется по всем элементам и считает, сколько объектов можно разместить, если между ними будет расстояние не меньше `ans`. Если количество объектов, которые можно разместить, больше или равно `k`, то возвращается `true`, иначе `false`.

Алгоритм работает за время O(n log m), где n - количество объектов, m - максимальное значение в массиве `arr`.

```
#include <bits/stdc++.h>
using namespace std;

bool check(vector<int> &arr, int k, int ans){
   int tmp = arr[0] + ans;
   int tmpK = 1;
   for(int i = 1; i < arr.size(); i++){
       if(arr[i] >= tmp){
           tmpK++;
           tmp = arr[i] + ans;
       }
   }

   if(tmpK >= k){
       return true;
   }else{
```

```
        return false;
    }
}

int searchBin(vector<int> &arr, int k){
    //int l = 0, r = arr[arr.size() - 1] - arr[0];
    int l = 0, r = arr[arr.size() - 1];

    while(l + 1 < r){
        int tmp = l + (r - l) / 2;

        if(check(arr, k, tmp)){
            l = tmp;
        }else{
            r = tmp;
        }
    }

    return l;
}

int main() {
    int n, k;
    cin >> n >> k;
    vector<int> arr(n);

    for(int i = 0; i < n; i++){
        cin >> arr[i];
    }

    cout << searchBin(arr, k);
    return 0;
}
```

Задача №2 «F.Число»

Пояснение к примененному алгоритму:

Этот алгоритм сортировки строк направлен на упорядочивание их таким образом, чтобы образовывалось минимально возможное число. Он основан на идее сравнивать строки не просто как строки, а как числа, объединенные в разных порядках. Например, для строк "3" и "30" числа будут 330 и 303, и нужно выбрать порядок, при котором число будет меньше.

Алгоритм использует сортировку вставками, начиная с первого элемента и поочередно вставляя его в правильное место в уже отсортированной части массива строк. Функция `sort` сравнивает каждую строку с предыдущей, объединяя их в разных порядках, и выбирает тот, который образует меньшее число.

Временная сложность этого алгоритма в худшем случае составляет O(n^2)), где n - количество строк в массиве. Это происходит из-за вложенного цикла в функции сортировки. Однако в лучшем случае его временная сложность может быть ближе к O(n), если строки уже отсортированы.

```
#include <bits/stdc++.h>

using namespace std;

void sort(string arr[100], int size){
   for (int i = 0; i < size; i++) {
       int j = i;
       string tmp = arr[i];
       while (j >= 0 && tmp + arr[j] <= arr[j] + tmp) {
           arr[j] = arr[j - 1];
           j--;
       }
       arr[j + 1] = tmp;
   }
}

void printArrayStrsAsNumber(string arr[100], int size){
   while (size >= 0) {
       cout << arr[size];
       size--;
```

```
    }
}

int main() {
    string str;
    string arrStrs[100];
    int size = 0;

    while (cin >> str) {
        if (str.empty() || str == "end") {
            break;
        }
        arrStrs[size] = str;
        size++;
    }

    sort(arrStrs, size);
    printArrayStrsAsNumber(arrStrs, size);

    return 0;
}
```

Задача №3 «G.Кошмар в замке»

Пояснение к примененному алгоритму:

Алгоритм имеет временную сложность O(n log(n)), где n - количество товаров. Это связано с сортировкой массива цен в порядке убывания с использованием функций `sort` и `reverse`. После сортировки происходит итерация по всем элементам массива, что занимает линейное время O(n).Память, требуемая для выполнения алгоритма, также линейно зависит от размера входных данных. Она используется для хранения массива цен и других переменных, что дает временную сложность O(n).Алгоритм эффективен в том, что он выполняет задачу оптимизации расходов, вычисляя сумму цен на основе сортированного списка товаров и пропуская самые дорогие товары в каждой группе. Он хорошо подходит для решения задач, где необходимо найти оптимальное решение среди набора данных.Поскольку сортировка производится в порядке убывания, алгоритм не гарантирует стабильность при сортировке равных элементов. Однако, в данной задаче это не критично, так как не требуется сохранение порядка элементов с равными значениями.

В целом, данный алгоритм обладает хорошей эффективностью и может быть эффективным инструментом для решения задач оптимизации расходов.

```
#include <bits/stdc++.h>

using namespace std;

struct Chr {
    char symbol;
    int value;
};

bool comp(Chr l, Chr r) {
    return l.value > r.value;
}

string getResByBinSearch(string str, vector<Chr> chrPair){
    int l = 0;
    int r = str.size() - 1;

    for (Chr pair: chrPair) {
```

```
        int posL = str.find(pair.symbol);
        if (str[l] != pair.symbol) swap(str[l], str[posL]);
        int posR = str.substr(l + 1).find(pair.symbol) + l + 1;
        if (str[r] != pair.symbol) swap(str[r], str[posR]);
        l++;
        r--;
    }

    return str;
}

int main() {
    string str;
    cin >> str;
    char symbol = 'a';
    vector<Chr> all;

    while (symbol <= 'z') {
        Chr pair;
        cin >> pair.value;
        pair.symbol = symbol;
        all.push_back(pair);
        symbol++;
    }

    sort(all.begin(), all.end(), comp);

    vector<Chr> chrPair;
    for (Chr pair: all) {
        if (count(str.begin(), str.end(), pair.symbol) >= 2) {
            chrPair.push_back(pair);
        }
    }
```

```
    cout << getResByBinSearch(str, chrPair) << endl;
    return 0;
}
```

Задача №4 «Н.Магазин»

Пояснение к примененному алгоритму:
Этот код реализует алгоритм с временной сложностью O(n log(n)), где n - количество товаров. Алгоритм сначала сортирует цены товаров в порядке убывания, используя функцию `sort`. Затем он итерируется по отсортированному вектору и суммирует цены (k-1) самых дешевых товаров в каждой группе, пропуская самые дорогие товары. Сумма этих цен сохраняется в переменной и выводится на экран.

Этот подход обеспечивает эффективную суммацию цен на основе сортированного списка товаров. В конечном итоге, алгоритм решает задачу оптимизации расходов, выбирая самые дешевые товары в каждой группе и пропуская более дорогие.

```
#include <bits/stdc++.h>
using namespace std;

int calc(vector<int> a, int k){
    int ans = 0;
    for(int i = 0; i < a.size(); i++){
        if((i + 1) % k == 0){
            ans = ans + a[i];
        }
    }

    return ans;
}
```

```cpp
int main() {
    int n, k;
    cin >> n >> k;
    vector<int> a(n);

    int tmpSum = 0;

    for(int i = 0; i < n; i++){
        cin >> a[i];
        tmpSum = tmpSum + a[i];
    }

    sort(a.begin(), a.end());
    reverse(a.begin(), a.end());

    cout << tmpSum - calc(a, k);
}
```