Университет ИТМО

Факультет программной инженерии и компьютерной техники

Лабораторная работа №2

по «Алгоритмам и структурам данных»

Базовые задачи

Выполнила:

Студентка группы P3215

Павличенко Софья Алексеевна

Преподаватели:

Косяков М.С.

Тараканов Д.С.

Санкт-Петербург

2025

Задача E «Коровы в стойла»

Код:

```cpp
#include <iostream>
#include <vector>

using namespace std;
#define ll long long

ll canPlace(ll m, vector<ll>& cows) {
  int k = 1, last = 0;
  for (size_t i = 1; i < cows.size(); ++i) {
    if (cows[i] - cows[last] >= m) {
      k++;
      last = i;
    }
  }
  return k;
}

int main() {
  ll n, k;
  cin >> n >> k;
  vector<ll> cows(n);
  for (int i = 0; i < n; ++i)
    cin >> cows[i];
  ll l = 0, r = cows[n - 1] - cows[0] + 1, m;
  while (r - l > 1) {
    m = (l + r) / 2;
    if (canPlace(m, cows) >= k)
      l = m;
    else
      r = m;
  }
  cout << l;
  return 0;
}
```

Пояснение к примененному алгоритму:

Алгоритм использует бинарный поиск по ответу: ищем максимальное расстояние между коровами. Функция canPlace проверяет, можно ли поставить хотя бы k коров с заданным минимальным расстоянием. Если можно, перемещаем l вправо, уменьшая отрезок поиска в два раза, иначе r влево. Так как функция монотонна (чем больше расстояние, тем меньше коров помещается), бинарный поиск работает корректно.

Время: O(NlogD), где N — число стойл, D — разница между крайними стойлами.
Память: O(N) — для хранения координат.

Задача F «Число»

Код:

```
#include <algorithm>
#include <iostream>
#include <vector>

using namespace std;

bool comp(string a, string b) {
  return a + b > b + a;
}

int main() {
  string s;
  vector<string> ns;
  while (cin >> s) {
    if (!s.empty())
      ns.push_back(s);
  }
  sort(ns.begin(), ns.end(), comp);
  string res;
  for (size_t i = 0; i < ns.size(); ++i)
    cout << ns[i];
  return 0;
}

```

Пояснение к примененному алгоритму:

Алгоритм читает все части числа, сортирует их с использованием компаратора, который сравнивает строки как a + b и b + a, чтобы выбрать порядок, дающий максимальное число. Затем соединяет и выводит отсортированные строки. Алгоритм работает, потому что такая сортировка гарантирует, что наиболее "выгодные" комбинации окажутся впереди.

Время: O(nlog(n*m)), где n — количество частей, m — максимальная длина строки.
Память: O(n*m) — для хранения всех частей.

Задача G «Кошмар в замке»

Код:

```cpp
#include <algorithm>
#include <iostream>
#include <vector>

using namespace std;

vector<int> c(26);
vector<int> cnt(26, 0);

bool comp(char a, char b) {
  if ((cnt[a - 'a'] == 1) != (cnt[b - 'a'] == 1))
    return !(cnt[a - 'a'] == 1);
  return c[a - 'a'] > c[b - 'a'];
}

int main() {
  string s;
  cin >> s;
  for (int i = 0; i < 26; ++i)
    cin >> c[i];
  vector<char> unique;
  for (char ch : s) {
    cnt[ch - 'a']++;
    if (cnt[ch - 'a'] == 1)
      unique.push_back(ch);
  }
  sort(unique.begin(), unique.end(), comp);
  vector<char> res(s.size());
  int l = 0, r = res.size() - 1;
  for (size_t i = 0; i < unique.size(); ++i) {
    if (cnt[unique[i] - 'a'] > 1) {
      res[l++] = unique[i];
      res[r--] = unique[i];
      cnt[unique[i] - 'a'] -= 2;
    } else
      break;
  }
  for (size_t i = 0; i < unique.size(); ++i) {
    for (int j = 0; j < cnt[unique[i] - 'a']; ++j)
      res[l++] = unique[i];
  }
  for (char ch : res)
    cout << ch;
  return 0;
}

```

Пояснение к примененному алгоритму:

Сначала мы считываем строку и веса букв, затем подсчитываем частоту каждой буквы в строке. Для того, чтобы максимизировать вес строки, символы сортируются: сначала те, которые встречаются более одного раза, идут в начало, а среди них сортируются по убыванию их веса. Затем буквы с частотой больше одного размещаются симметрично, чтобы максимизировать расстояние между одинаковыми буквами, а оставшиеся символы размещаются в центре. В результате получается строка с максимально возможным весом.

Время: сначала мы выполняем подсчёт частоты символов, что занимает O(n). Затем сортировка уникальных символов происходит за O(26log26), что является константой, так как количество уникальных символов всегда ограничено 26. Формирование результата также занимает O(n). В итоге, сложность — O(n), где n — длина строки.
Память: O(n) — для хранения строки и вспомогательных массивов (частот и результатов). Память для весов символов фиксирована и занимает O(26), что также можно считать константой.

Задача H «Магазин»

Код:

```cpp
#include <algorithm>
#include <iostream>
#include <vector>

using namespace std;

int main() {
  int n, k, s = 0;
  cin >> n >> k;
  vector<int> a(n);
  for (int i = 0; i < n; ++i) {
    cin >> a[i];
    s += a[i];
  }

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

  for (int i = n - 1; i >= 0; i -= k) {
    if (i - k + 1 >= 0)
      s -= a[i - k + 1];
  }
  cout << s;
  return 0;
}

```

Пояснение к примененному алгоритму:

Товары сортируются по цене, и после сортировки товары разбиваются на блоки по k товаров. В каждом блоке самый дешевый товар становится бесплатным, и его цена вычитается из общей суммы. Таким образом, для каждого блока из k товаров вычитается стоимость самого дешевого товара.

Время: O(nlogn), где n — количество товаров.
Память: O(n) — для хранения списка цен.