Факультет программной инженерии и компьютерной техники

Алгоритмы и структуры данныхка

Лабораторная работа №4

Сортировка

Преподаватель: Косяков Михаил Сергеевич
Выполнил: Ле Чонг Дат
Группа: P3231

2021 г.

1 Задание

Вариант: 1604. В Стране Дураков

Укажите k знаков и количество каждого типа $n_1, n_2, ..., n_k$. $n_1 + n_2 + ... + n_k = n$. Расставьте n знаков так, чтобы количество разных пар знаков i, $i+1$, где $i < n$, было максимальным.

2 Решение

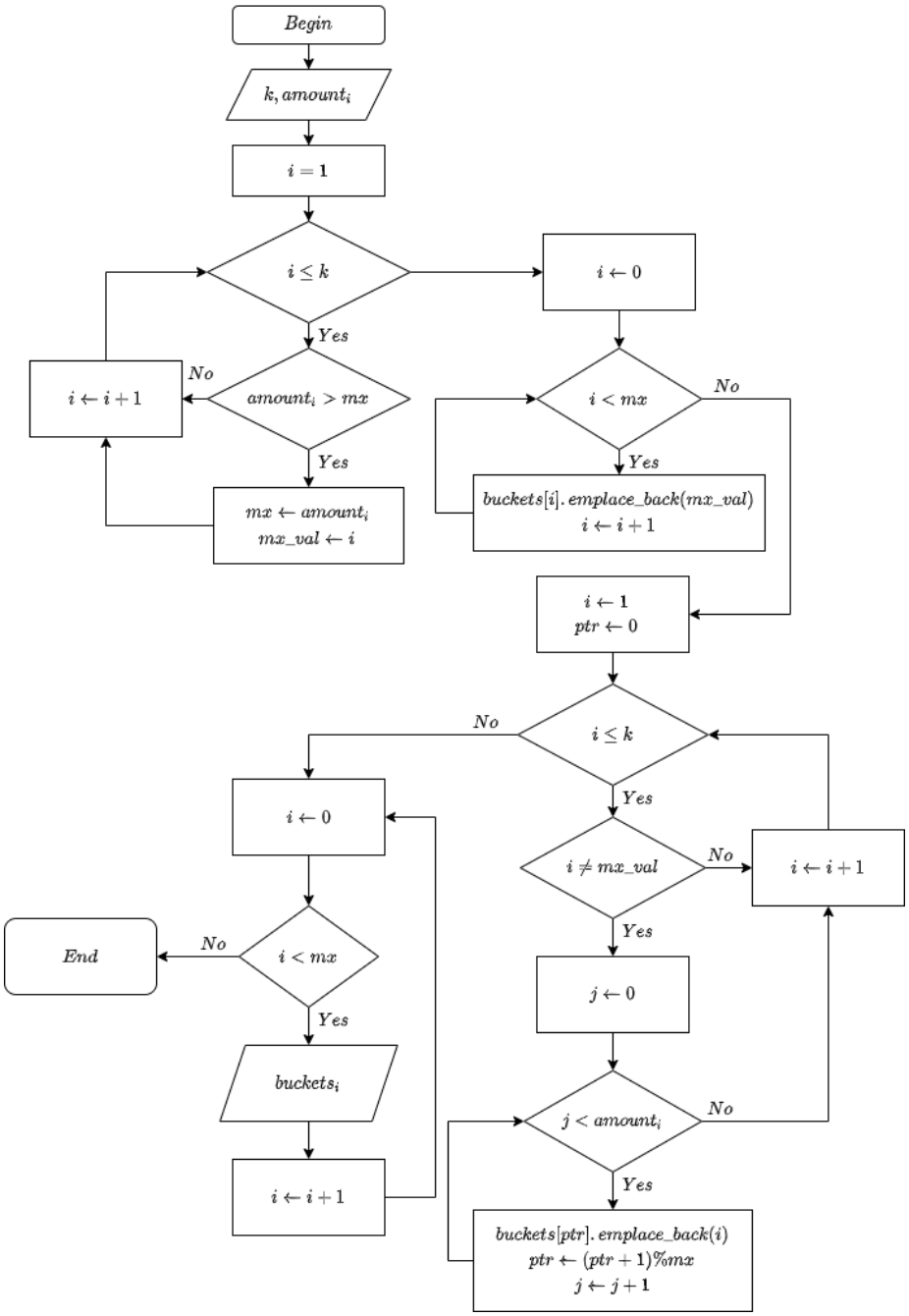
Вызов max - это тип с наибольшим количеством знаков.

Если $n_{max} > \ sum_{i=1}^{i \leq n; i \neq max} n_i + 1$, очевидно будет одна и та же пара знаков должна быть смежной. Предположим, у нас есть n_{max} bucket, изначально каждая корзина имеет знак max. Пусть ptr будет позицией текущего сегмента, для остальных знаков со знаком i мы добавляем каждый из знаков i в корзину $(ptr + 1)\%n_{max}$. В одной и той же корзине не будет двух одинаковых элементов, фактически, если есть два одинаковых знака x в одной и той же корзине, это означает, что количество элементов x больше, чем n_{max}.

3 Схема програмы

Глобальные переменные:

- k - количество различных типов знаков с ограничением скорости
- mx - n_{max}
- mx_val - тип с наибольшим количеством знаков.
- $amount_i$ - n_i



4 Реализация

```
#include <bits/stdc++.h>
using namespace std;
const int K = 100010;
int k;
```

```
int mx, mx_val;
int amount[K];
int main() {
    ios::sync_with_stdio(0);
    cin.tie(0); cout.tie(0);

    cin >> k;
    for(int i = 1; i <= k; ++i) {
        cin >> amount[i];
        if (amount[i] > mx) {
            mx = amount[i];
            mx_val = i;
        }
    }
    vector<int> buckets[mx];
    for(int i = 0; i < mx; ++i)
        buckets[i].emplace_back(mx_val);
    int ptr = 0;
    for(int i = 1; i <= k; ++i) if (i != mx_val)
        for(int j = 0; j < amount[i]; ++j) {
            buckets[ptr].emplace_back(i);
            ptr = (ptr + 1) % mx;
        }
    for(int i = 0; i < mx; ++i)
        for(auto x: buckets[i]) cout << x << ' ';
}
```

5 Вывод

Изначально я использовал жадный алгоритм для решения проблемы, выбрав следующий элемент с наибольшим количеством и отличный от предыдущего. После многих улучшений и наблюдений я придумал алгоритм оптимальной сложности.