Описание задачи

Состояние N фотонов в M модах можно описать как суперпозицию фоковских состояний с комплексными коэффициентами. Фоковское состояние – это состояние с определённым числом фотонов в каждой моде, поэтому его можно хранить как вектор длины M, каждый элемент которого целочисленный и равен числу заполнения соответствующей моды. Произвольное состояние фотонов можно хранить в ассоциативном контейнере std::map, где ключом является фоковское состояние, а значением – комплексная амплитуда данного фоковского состояния.

Над состоянием возможны различные операции, например, сложение, вычитание, умножение на константу, скалярное произведение и т. д. Для простоты вам требуется реализовать сложение двух состояний (см. прототип функции `add()` ниже), представленных как std::map. Также необходимо реализовать функцию `print()` для вывода состояния на экран.

Учитывая, что ключи в std::map хранятся в отсортированном виде (по умолчанию для векторов используется лексикографическая сортировка по возрастанию), алгоритм сложения можно реализовать за линейное время, точнее за O(l1 + l2), где l1, l2 – размеры контейнеров. Общая идея метода следующая. Сначала сравниваются первые элементы двух состояний. Если ключи (фоковские состояния) одинаковые, то их амплитуды складываются. Если, например, ключ для первого состояния больше, чем для второго, то нужно итерировать второе состояние до нахождения совпадения ключей и т.д. Таким образом, последовательно будут просмотрены все элементы как первого, так и второго состояний, что ведёт к сложности O(l1 + l2).

state.h

```
#include <complex>
#include <vector>
#include <map>

using Fock = std::vector<int>;
using Amp = std::complex<double>;
using State = std::map<Fock, Amp>;

State add(const State &s1, const State &s2); // Складывает два состояния
и возвращает результат
void print(const State &s); // Распечатывает состояние в stdout
```

main.c

```
#include “state.h”

int main()
{
      State s1;
      s1[{1, 0, 0}] = 1.1;  // Значения коэффициентов, очевидно, нарушают
      s1[{0, 1, 0}] = 2.2;  // нормировку, но алгоритм работы от этого
      s1[{0, 0, 1}] = 3.3;  // не меняется

      State s2;
      s2[{0, 0, 1}] = 4.0;
```

```
    s2[{1, 0, 0}] = Amp(5.0, 6.0);
    s2[{2, 1, 0}] = 7.0;

    State s3 = add(s1, s2);
    print(s3); // Пример вывода:
               // {0, 0, 1} = 7.3 + 0.0i
               // {0, 1, 0} = 2.2 + 0.0i
               // {1, 0, 0} = 6.1 + 6.0i
               // {2, 1, 0} = 7.0 + 0.0i
    return 0;
}
```

Дополнительные вопросы

1. Можно ли реализовать операцию сложения двух состояний за линейное время, представив состояние в виде несортированного вектора (см. ниже)?

```
struct Element
{
    Fock f;
    Amp a;
}
using State = std::vector<Element>;
```

2. Можно ли реализовать операцию сложения двух состояний за линейное время, заменив std::map на хэш-таблицу std::unordered_map, в которой операции вставки, удаления и доступа по ключу выполняются за константное время?