МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждения высшего образования

«МИРЭА - Российский технологический университет»

РТУ МИРЭА

Отчет по выполнению практической работы № 8.2

Тема:

«Алгоритмические стратегии или методы разработки алгоритмов. Перебор и методы его сокращения.»

Дисциплина: «Структуры и алгоритмы обработки данных»

Выполнил студент: Туляшева А.Т.

Группа: ИКБО-42-23

Москва - 2024

Содержание

ЦЕЛЬ РАБОТЫ

Цель: получить навыки применения методов, позволяющих сократить число переборов в задачах, которые могут быть решены только методом перебора всех возможных вариантов решения.

ЗАДАНИЕ 1

Ответьте на вопросы.

1. В чём суть метода грубой силы (прямолинейное решение, метод решения «в лоб»)?

Метод грубой силы предполагает полный перебор всех возможных решений задачи для выбора оптимального. Этот подход прост в реализации, но часто неэффективен с точки зрения времени выполнения, так как сложность задачи может быть очень высокой.

Примеры задач:

- Поиск всех подстрок строки и сравнение их с искомой строкой.
- Перебор всех возможных маршрутов в задаче коммивояжера (без оптимизации).

Преимущества:

- Простота реализации.
- Гарантированное нахождение решения (если оно существует).

Недостатки:

- Высокая вычислительная сложность.

2. В чём суть метода «разделяй и властвуй» (декомпозиция)? Приведите примеры задач, которые решаются этим способом? Какие алгоритмы можно отнести к этому методы?

Этот метод заключается в разбиении задачи на несколько меньших подзадач, решения которых объединяются для получения общего результата. Метод эффективен, если подзадачи проще исходной и могут решаться независимо друг от друга.

Примеры задач:

- Сортировка слиянием: массив делится на две части, каждая сортируется, затем объединяется.
- Быстрая сортировка: массив делится на элементы, меньшие и большие опорного элемента.
- Поиск в массиве (бинарный поиск): массив делится пополам, и поиск осуществляется только в одной из частей.

Алгоритмы, относящиеся к методу:

- Быстрая сортировка.
- Сортировка слиянием.
- Алгоритм FFT (быстрое преобразование Фурье).

Преимущества:

- Ускорение работы за счёт рекурсивного деления задач.
- Возможность использования параллельных вычислений.

Недостатки:

- Не подходит для задач, где подзадачи сложно разъединить.

3. В чём суть метода динамического программирования? Приведите примеры задач, которые решаются этим способом?

Метод заключается в разбиении задачи на перекрывающиеся подзадачи и запоминании результатов этих подзадач для повторного использования. Это помогает избежать избыточных вычислений.

Примеры задач:

- -Задача о рюкзаке (0/1): выбор предметов для максимизации ценности при ограничении веса.
- Поиск наибольшей общей подпоследовательности (LCS).
- Задача о размене монет.

Пример алгоритмов:

- Алгоритм Флойда-Уоршелла (нахождение кратчайших путей).
- Поиск оптимального пути (алгоритм Беллмана-Форда).

Преимущества:

- Эффективность за счёт устранения повторяющихся вычислений.
- Часто обеспечивает решение за полиномиальное время.

Недостатки:

- Увеличение потребления памяти.
- Требуется глубокое понимание структуры задачи.

4. В чём суть «жадного метода»? Приведите примеры задач, которые решаются этим способом?

Этот метод предполагает принятие локально оптимальных решений на каждом шаге в надежде, что это приведёт к глобально оптимальному решению. Жадный метод работает корректно, если задача обладает свойством "жадности" (локально оптимальное решение гарантирует глобальную оптимальность).

Примеры задач:

- Алгоритм Крускала или Прима для построения минимального остовного дерева.
- Задача о размене монет (если монеты имеют "жадную" структуру).
- Задача о покрытии множеств.

Преимущества:

- Простота и скорость выполнения.
- Низкие требования к памяти.

Недостатки:

- Не всегда приводит к оптимальному решению.
- Требуется проверка задачи на наличие свойства жадности.

5. В чём суть «метода ветвей и границ»? Приведите примеры задач, которые решаются этим способом?

Метод основан на разбиении задачи на подзадачи (ветвление) с одновременной оценкой границ решения для отсечения заведомо невыгодных подзадач.

Примеры задач:

- Задача коммивояжера: поиск оптимального маршрута с отсечением невыгодных вариантов.
- Задача о рюкзаке: перебор с отсечением невозможных комбинаций.

Основные этапы:

1. Построение дерева решений.
2. Оценка границ каждой ветви.
3. Отсечение ветвей, не ведущих к оптимальному решению.

Преимущества:

- Экономия вычислительных ресурсов за счёт отсечения.
- Применение к задачам с комбинаторной сложностью.

Недостатки:

- Неэффективен, если не удаётся эффективно оценивать границы.
- Может потребовать большого объёма памяти.

ЗАДАНИЕ 2

2.1 Постановка задачи

1. Разработать алгоритм решения задачи с применением метода, указанного в варианте и реализовать программу.

2. Оценить количество переборов при решении задачи стратегией «в лоб» - грубой силы. Сравнить с числом переборов при применении метода.

Вариант – 1:

№_	Задача	Метод
1	Посчитать число последовательностей нулей и единиц длины *n*, в которых не встречаются две идущие подряд единицы.	Динамическое программирование

2.2 Математическая модель решения (описание алгоритма)

1) Динамическое программирование (ДП):

- Обозначим dp[i] как количество последовательностей длины i, которые не содержат двух подряд идущих единиц.
- Базовые случаи:
  - dp[1] = 2: последовательности длины 1 — "0", "1".
  - dp[2] = 3: последовательности длины 2 — "00", "01", "10".
- Рекуррентное соотношение:
  - dp[i] = dp[i − 1] + dp[i − 2].
  - Это основано на том, что последовательность длины i может быть:
    - Дополнила последовательность длины i−1 символом "0".
    - Дополнила последовательность длины i−2 символами "10".
- Итоговое решение для длины n: dp[n].

2) Решение "в лоб" (перебор):

- Перебираются все возможные последовательности длины n (2^n вариантов).
- Каждая последовательность проверяется на наличие подряд идущих единиц.

2.3 Код программы

На рисунке 1 представлено решение задачи методом динамического программирования и перебором.

Метод динамического программирования (функция countSequencesDP):

- Вычисляем dp[i] последовательно для всех i от 1 до n.
- Количество операций — O(n), так как каждый элемент таблицы вычисляется за O(1).

Метод перебора (функция countSequencesBruteForce):

- Перебираем 2^n вариантов.
- Для каждого варианта проверяем, есть ли подряд идущие единицы.
- Количество операций — $O(2^n \times n)$.

```cpp
#include ...
using namespace std;

int countSequencesDP(int n) { //кол-во после-тей методом дп
    if (n == 1) return 2;
    if (n == 2) return 3;
    vector<int> dp(n + 1);
    dp[1] = 2; //для длины 1
    dp[2] = 3; //2
    //таблица дп
    for (int i = 3; i <= n; i++) {
        dp[i] = dp[i - 1] + dp[i - 2];
    }
    return dp[n];
}

int countSequencesBruteForce(int n) { //перебор всех посл-тей длины n
    int total = 0;
    int combinations = pow(2, n); //всего 2^n последовательностей
    //перебор всех возможных комбинаций
    for (int i = 0; i < combinations; i++) {
        bool valid = true;
        for (int j = 0; j < n - 1; j++) {
            //проверка на наличие двух подряд идущих единиц
            if ((i & (1 << j)) && (i & (1 << (j + 1)))) {
                valid = false;
                break;
            }
        }
        if (valid) total++;
    }
    return total;
}

int main() {
    setlocale(LC_ALL, "rus");
    int n;
    cout << "Введите длину последовательности n: ";
    cin >> n;
    int resultDP = countSequencesDP(n);
    cout << "\nКоличество последовательностей (метод ДП): " << resultDP << endl;
    int resultBruteForce = countSequencesBruteForce(n);
    cout << "Количество последовательностей (перебор): " << resultBruteForce << endl;
    cout << "\nКоличество переборов для ДП: " << n << endl;
    cout << "Количество переборов для грубой силы: " << pow(2, n) << endl;
}
```

Рисунок 1 – Программа

2.4. Тестирование

Было проведено тестирование при различных значениях n – длина последовательности (рис. 2-3).

```
Введите длину последовательности n: 3

Количество последовательностей (метод ДП): 5
Количество последовательностей (перебор): 5

Количество переборов для ДП: 3
Количество переборов для грубой силы: 8
```

Рисунок 2 – Тестирование

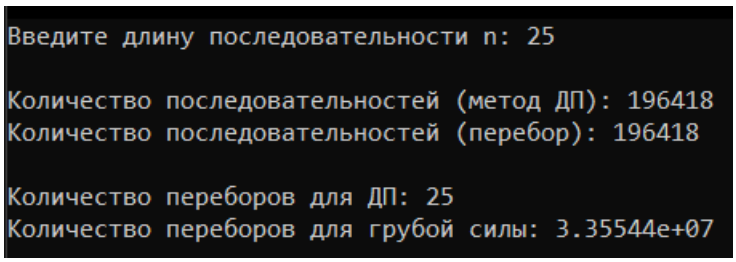

Рисунок 3 – Тестирование

Метод ДП выполняется намного быстрее. Метод динамического программирования значительно более эффективный, так как использует ранее вычисленные результаты, что сокращает количество операций и время выполнения

ВЫВОД

В ходе работы были решены задачи поиска минимального времени пути с использованием двух методов: динамического программирования и метода грубой силы. Метод динамического программирования оказался значительно более эффективным, так как использует ранее вычисленные результаты, что сокращает количество операций и время выполнения. Метод грубой силы требует перебора всех возможных путей, что приводит к экспоненциальному росту вычислений и долгому времени работы при больших размерах поля.

Метод динамического программирования предпочтительнее для решения данной задачи, так как он значительно быстрее и эффективнее.

ИНФОРМАЦИОННЫЕ ИСТОЧНИКИ

1. Вирт Н. Алгоритмы и структуры данных. Новая версия для Оберона.
2. Практические занятия Сорокин А.В. - РТУ МИРЭА 2024
3. Лекции Бузыкова Ю.С. – РТУ МИРЭА 2024
4. Практические занятия преподавателя Муравьевой Е. А., 2024.
5. Лекции Лозовский В.В. – РТУ МИРЭА 2024