Міністерство освіти і науки України

Національний університет «Запорізька Політехніка»

Кафедра програмних засобів

ЗВІТ

з лабораторної роботи №4

з дисципліни «Алгоритми та Структури Даних» на тему:

«Динамічне програмування»

Виконав:

Студент групи КНТ-122 О. А. Онищенко

Прийняли:

Старший викладач: Л. Ю. Дейнега

2023

Динамічне програмування

Мета роботи

Вивчити основні принципи динамічного програмування.

Навчитися використовувати динамічне програмування для розв’язання практичних завдань.

Завдання до роботи

- Розробити програмне забезпечення, що розв’язує задачу у відповідності з індивідуальним завданням з пункту із використанням принципів динамічного програмування.
- Розроблюваний програмний проєкт має складатися з класу, що описує задачу, сформульовану в індивідуальному завданні, а також має містити окремий модуль, що забезпечує інтерфейсну взаємодію з користувачем.
- Клас вхідних даних задачі має дозволяти:
  - задавати початкові дані;
  - вводити нові параметри;
  - коригувати та видаляти існуючі;
  - розв ’язувати відповідну задачу.
- Індивідуальне завдання, що складається з 2 задач, наведених нижче, обрати з таблиці 4.1 у відповідності з варіантом.
- Під час експериментального тренування роботів їх було посаджено на велосипеди. Уся група роботів відправляється на велосипедах вузькою велосипедною доріжкою в однаковому напрямку з деякими проміжками. Кожен робот керує велосипедом на деякій заданій швидкості, яка змінюється тільки у тому випадку, якщо робот наздожене велосипедиста з меншою швидкістю: не маючи змоги обігнати більш

повільного велосипедиста, він знизить швидкість до швидкості велосипедиста, який їде попереду. Таким чином, через деякий час роботи будуть розбиті на групи, кожна з яких буде рухатись зі своєю сталою швидкістю. Завдання програміста полягає в тому, щоб розбити роботів на деяку задану кількість груп. Визначити, скільки існує способів старту роботів (тобто порядку, в якому кожен робот почне рух велосипедною доріжкою), які в результаті сформують задану кількість груп.

- На одній з вулиць містечка будинки класифіковано за трьома типами: перший – звичайні житлові споруди, другий – промислові споруди, а третій – міські заклади (лікарні, школи тощо). У результаті вулиця схематично зображена набором літер, кожна з яких визначає тип будинку. У процесі збору інформації про місто була створена ма триця – таблиця, в якій кожен стовпчик і рядок відповідають одному з типів будівель. Відповідно клітинка такої таблиці визначає, чи розташовані на даній вулиці міста поруч будівлі заданого типу. Матриця симетрична. Визначити, скільки існує способів взаємного розташування будинків даних типів між собою за заданою матрицею для заданої кількості будинків на вулиці, тобто кількість можливих наборів літер заданої довжини, що відповідають заданій матриці.

- Виконати аналіз розроблених алгоритмів для розв’язання індивідуального завдання щодо часу їх роботи та кількості використаної пам’яті.

- Виконати тестування розробленого програмного забезпечення.

- Порівняти одержані результати виконаних тестів, провести аналіз вірності, коректності та адекватності роботи розробленого пр ограмного забезпечення і використаних методі.

Результати виконання роботи

```
1. Robots
2. Building Arrangements
3. Exit
Enter your choice: 1

Enter the number of robots: 5
Enter the speeds of the robots, separated by spaces: 2 3 1 4 2
Enter the number of groups: 3

Number of ways to start the robots: 6
```

```
1. Robots
2. Building Arrangements
3. Exit
Enter your choice: 2

Can building type A be next to building type A? (yes/no): yes
Can building type A be next to building type B? (yes/no): yes
Can building type A be next to building type C? (yes/no): no
Can building type B be next to building type A? (yes/no): yes
Can building type B be next to building type B? (yes/no): yes
Can building type B be next to building type C? (yes/no): yes
Can building type C be next to building type A? (yes/no): no
Can building type C be next to building type B? (yes/no): yes
Can building type C be next to building type C? (yes/no): yes
Enter the number of buildings: 5

Number of arrangements: 70
```

Код

```python
class RobotGroup:
    def __init__(self, num_robots, speeds, num_groups):
        self.num_robots = num_robots
        self.speeds = speeds
        self.num_groups = num_groups
```

```
        self.dp = [[-1] * (num_robots + 1) for _ in range(num_groups +
1)]

    def count_ways(self):
        return self._count_ways(self.num_groups, self.num_robots)

    def _count_ways(self, groups, robots):
        if groups == 0 and robots == 0:
            return 1
        if groups == 0 or robots == 0:
            return 0
        if self.dp[groups][robots] != -1:
            return self.dp[groups][robots]

        ways = 0
        for i in range(1, robots + 1):
            ways += self._count_ways(groups - 1, i - 1)
        self.dp[groups][robots] = ways
        return ways


class BuildingArrangement:
    def __init__(self, matrix):
        self.matrix = matrix
        self.memo = {}

    def count_arrangements(self, n):
        return self._count_arrangements(n, "A")

    def _count_arrangements(self, n, building_type):
        if n == 0:
            return 1

        if (n, building_type) in self.memo:
            return self.memo[(n, building_type)]

        count = 0
        for prev_building_type in ["A", "B", "C"]:
            if self.matrix[prev_building_type][building_type]:
                count += self._count_arrangements(n - 1,
prev_building_type)

        self.memo[(n, building_type)] = count
        return count


def main_menu():
    while True:
        print("\n1. Robots\n2. Building Arrangements\n3. Exit")
```

```
        choice = input("Enter your choice: ")
        print()

        if choice == "1":
            num_robots = int(input("Enter the number of robots: "))
            speeds = list(
                map(
                    int,
                    input(
                        "Enter the speeds of the robots, separated by
spaces: "
                    ).split(),
                )
            )
            num_groups = int(input("Enter the number of groups: "))
            robot_group = RobotGroup(num_robots, speeds, num_groups)
            num_ways = robot_group.count_ways()
            print(f"\nNumber of ways to start the robots: {num_ways}")

        elif choice == "2":
            matrix = {}
            for building_type in ["A", "B", "C"]:
                matrix[building_type] = {}
                for other_building_type in ["A", "B", "C"]:
                    can_be_next_to = input(
                        f"Can building type {building_type} be next to
building type {other_building_type}? (yes/no): "
                    )
                    matrix[building_type][other_building_type] = (
                        can_be_next_to.lower() == "yes"
                    )
            num_buildings = int(input("Enter the number of buildings: "))
            arrangement = BuildingArrangement(matrix)
            num_arrangements =
arrangement.count_arrangements(num_buildings)
            print(f"\nNumber of arrangements: {num_arrangements}")

        else:
            break


if __name__ == "__main__":
    main_menu()
```

Висновки

Таким чином, ми вивчили основні принципи динамічного програмування, а також навчилися використовувати динамічне програмування для розв’язання практичних завдань.

Контрольні питання

У чому полягає динамічне програмування?

Динамічне програмування - це метод, який використовується для вирішення оптимізаційних задач шляхом розбиття складної задачі на простіші підзадачі, пошуку оптимального рішення цих підзадач, зберігання результатів підзадач (запам'ятовування), їх повторного використання і, нарешті, обчислення результату складної задачі.

Для розв’язання яких задач можна використовувати динамічне програмування?

Динамічне програмування можна використовувати для розв'язання широкого кола задач, включаючи, але не обмежуючись ними:

Знаходження найкоротшого шляху у графі

Знаходження найдовшої спільної підпослідовності між двома рядками

Знаходження максимальної суми підмасиву

Знаходження мінімальної кількості монет, необхідних для здійснення певної кількості змін

Знаходження оптимального способу розрізати прут на частини, щоб максимізувати його вартість

З яких етапів складається процес розроблення алгоритмів динамічного програмування?

Етапи процесу розробки алгоритмів динамічного програмування наступні:

Визначення підзадач та структури оптимального розв'язку.

Визначити значення рішення рекурсивно через значення менших підзадач.

Обчислити вартість оптимального рішення за методом "знизу-вгору", вирішуючи підпроблеми в порядку зростання їх розміру.

Побудувати оптимальний розв'язок на основі обчисленої інформації