Міністерство освіти і науки України

Національний університет «Запорізька Політехніка»

Кафедра програмних засобів

ЗВІТ

з лабораторної роботи №3

з дисципліни «Спортивне програмування» на тему:

«Рекурсивні алгоритми»

Виконав:

Студент групи КНТ-122 О. А. Онищенко

Прийняли:

Викладач: С. Д. Леощенко

2023

Рекурсивні алгоритми

Мета роботи

Вивчити основні можливості та принципи роботи рекурсивних алгоритмів.

Завдання до роботи

Обрати та виконати двізадачі із запропонованого переліку

- Є N осіб і цілі числа A1,..., AN; людину і необхідно познайомити з Ai людьми. Чи можна це зробити?

- Дано дві цілочисельних таблиці A[1:10] і B[1:15]. Розробити алгоритм і написати програму, яка перевіряє, чи є ці таблиці схожими. Дві таблиці називаються схожими, якщо збігаються множини чисел, що зустрічаються в цих таблицях.

- Задано сімейство множин літер. Знайти таке k, для якого можна побудувати безліч, що складається з k літер, причому кожна з них належить рівно k множинам заданого сімейства.

- Впорядкувати за не зростанням 5 чисел за 7 операцій порівняння.

- Дано цілі M і N та масив дійсних чисел X[1..N]. Знайти ціле число для якого сума x[i] + ... + x[i+M] найближче до нуля.

- Є два відсортованих за не зростанням масиви A[1,N] і B[1,M]. Отримати відсортований за не зростанням масив C[1, N+M], що складається з елементів масивів A і B ("злити" разом масиви A і B).

- Дано масив X[1..N]. Необхідно циклічно зрушити його на k елементів вправо (тобто елемент X[i] після зсуву повинен стояти на місці X[i+k]; тут ми вважаємо, що після X[N] йде X[1]). Дозволяється використовувати тільки кілька додаткових слотів пам'яті (додаткового масиву заводити не можна!).

- Маємо N каменів ваги A1, A2,..., AN. Необхідно розбити їх на дві купи таким чином, щоб ваги Куп відрізнялися не більше ніж в 2 рази. Якщо цього зробити не можна, то вказати це.

- Є 2N чисел. Відомо що їх можна розбити на пари таким чином, що добутки чисел в парах рівні. Зробити розбиття, якщо числа натуральні.

- Є 2N чисел. Відомо що їх можна розбити на пари таким чином, що добутки чисел в парах рівні. Зробити розбиття, якщо числа цілі.

Результати виконання

```
MAIN MENU
1. Determine if tables are similar
2. Merge two arrays in descending order
3. Exit
: 1

TASK ONE
1. Generate random tables
2. Run tests
3. Go back
: 1


6 9 8 4
8 5 8 5
6 6 2 3
9 7 7 3

8 4 3 8 5 2 6 6
6 3 3 1 7 8 3 6
4 9 3 3 1 9 8 4
8 6 3 2 6 4 9 7
9 3 6 3 6 6 9 1
4 1 3 4 1 7 3 5
7 2 5 4 5 3 9 9
7 8 2 3 4 2 4 7

The two tables are not similar

MAIN MENU
1. Determine if tables are similar
2. Merge two arrays in descending order
3. Exit
: 2

TASK TWO
1. Enter custom data
2. Run tests
3. Go back
: 1

Enter the elements of the first array: 9 6 3 1
Enter the elements of the second array: 5 4 2 1

Merged arrays: [9, 6, 5, 4, 3, 2, 1, 1]

MAIN MENU
1. Determine if tables are similar
2. Merge two arrays in descending order
3. Exit
: 1

TASK ONE
1. Generate random tables
2. Run tests
3. Go back
: 2

All tests have passed

MAIN MENU
1. Determine if tables are similar
2. Merge two arrays in descending order
3. Exit
: 2

TASK TWO
1. Enter custom data
2. Run tests
3. Go back
: 2

All tests have passed
```

Програмний код

```
"""
- Дано дві цілочисельних таблиці A[1:10] і B[1:15]. Розробити алгоритм і написати програму, яка перевіряє, чи є ці таблиці схожими. Дві таблиці називаються схожими, якщо збігаються множини чисел, що зустрічаються в цих таблицях
- Є два відсортованих за не зростанням масиви A[1,N] і B[1,M]. Отримати відсортований за не зростанням масив C[1, N+M], що складається з елементів масивів A і B ("злити" разом масиви A і B).
"""


def compareTables(A: [[int]], B: [[int]]):
    # Дано дві цілочисельних таблиці A[1:10] і B[1:15]. Розробити алгоритм і написати програму, яка перевіряє, чи є ці таблиці схожими. Дві таблиці називаються схожими, якщо збігаються множини чисел, що зустрічаються в цих таблицях

    def isSubList(childList, parentList):
        # taking each element in a child list
        for element in childList:
            # checking if the element is not in parent list
            if element not in parentList:
                # if so, return false
                return False
        # if we didnt return in the loop, then the child list is a sublist of parent list
        return True

    # taking two tables and iterating over each one
    for subList, List in zip(A, B):
        # checking if current sublist is not a sublist of list from larger table
        if not isSubList(subList, List):
            # if not, return false
            return False
    # if we exited out of the loop and didnt return False, this means that each list in table one is a sublist of table two
    return True


def mergeArrays(A, B, C):
    # Є два відсортованих за не зростанням масиви A[1,N] і B[1,M]. Отримати відсортований за не зростанням масив C[1, N+M], що складається з елементів масивів A і B ("злити" разом масиви A і B).
```

```
    # if the first array becomes empty
    if not A:
        # we just append our second array to result
        return C + B
    # if the second array becomes empty
    if not B:
        # same deal, appending the first array to our result
        return C + A

    # compare the two largest elements in the arrays, which are the first
ones
    if A[0] > B[0]:
        # now recursively merge the two arrays while removing the larger
element from array A and appending it to array result
        return mergeArrays(A[1:], B, C + [A[0]])
    # if an element from the second array is larger
    else:
        # then recursively merge the arrays, but remove the first element
from array B and append it to our result
        return mergeArrays(A, B[1:], C + [B[0]])


def testCompareTables():
    assert (
        compareTables(
            [
                [1, 2, 3],
                [3, 2, 1],
                [4, 4, 4],
            ],
            [[1, 3, 4, 1, 2, 6], [3, 5, 8, 1, 2, 2], [4, 7, 5, 9, 4, 4]],
        )
        == True
    ), "Test 1 failed"
    assert (
        compareTables(
            [
                [1, 2],
                [3, 4],
            ],
            [
                [1, 1, 1],
                [2, 2, 2],
            ],
        )
        == False
    ), "Test 2 failed"
    assert (
        compareTables(
```

```
            [
                [9],
                [4],
            ],
            [
                [3, 2, 9],
                [4, 1, 1],
            ],
        )
        == True
    ), "Test 3 failed"
    assert (
        compareTables(
            [
                [9, 6, 2, 1],
                [6, 1, 7, 8],
                [5, 3, 7, 1],
            ],
            [
                [8, 3, 6, 1, 5, 9],
                [1, 6, 9, 2, 5, 6],
                [2, 9, 5, 7, 2, 7],
            ],
        )
        == False
    ), "Test 4 failed"
    print("All tests have passed")


def testMergeArrays():
    assert mergeArrays([7, 5, 3, 1], [8, 6, 4, 2], []) == [
        8,
        7,
        6,
        5,
        4,
        3,
        2,
        1,
    ], "Test 1 failed"
    assert mergeArrays([3, 2, 1], [6, 5, 4], []) == [6, 5, 4, 3, 2, 1],
"Test 2 failed"
    assert mergeArrays([1, 1, 1], [1, 1, 1], []) == [1, 1, 1, 1, 1, 1],
"Test 3 failed"
    assert mergeArrays([3, 2, 1], [], []) == [3, 2, 1], "Test 4 failed"
    assert mergeArrays([], [6, 5, 4], []) == [6, 5, 4], "Test 5 failed"
    assert mergeArrays([965, 240, 120, 90, 30, 20], [], []) == [
        965,
        240,
```

```
        120,
        90,
        30,
        20,
    ], "Test 6 failed"
    assert mergeArrays([], [], []) == [], "Test 7 failed"
    assert mergeArrays([50, 40, 30, 20, 10], [55, 45, 35, 25, 15], []) ==
[
        55,
        50,
        45,
        40,
        35,
        30,
        25,
        20,
        15,
        10,
    ], "Test 8 failed"
    print("All tests have passed")


import random


def menu():
    while True:
        print("\nMAIN MENU")
        print("1. Determine if tables are similar")
        print("2. Merge two arrays in descending order")
        print("3. Exit")
        choice = int(input(": "))

        if choice == 1:
            print("\nTASK ONE")
            print("1. Generate random tables")
            print("2. Run tests")
            print("3. Go back")
            localChoice = int(input(": "))
            print()

            if localChoice == 1:
                tableA = [[random.randint(1, 9) for _ in range(4)] for _
in range(4)]
                tableB = [[random.randint(1, 9) for _ in range(8)] for _
in range(8)]
                res = compareTables(tableA, tableB)

                print()
```

```
                for array in tableA:
                    for element in array:
                        print(element, end=" ")
                    print()

                print()
                for array in tableB:
                    for element in array:
                        print(element, end=" ")
                    print()

                print(f"\nThe two tables are {'similar' if res else 'not
similar'}")
            elif localChoice == 2:
                testCompareTables()

        elif choice == 2:
            print("\nTASK TWO")
            print("1. Enter custom data")
            print("2. Run tests")
            print("3. Go back")
            localChoice = int(input(": "))
            print()

            if localChoice == 1:
                A = list(
                    map(int, input("Enter the elements of the first
array: ").split())
                )
                B = list(
                    map(int, input("Enter the elements of the second
array: ").split())
                )
                if A != sorted(A, reverse=True) or B != sorted(B,
reverse=True):
                    A = sorted(A, reverse=True)
                    B = sorted(B, reverse=True)
                res = mergeArrays(A, B, [])
                print(f"\nMerged arrays: {res}")
            elif localChoice == 2:
                testMergeArrays()

        else:
            break


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

Висновки

Таким чином, ми вивчили основні можливості та принципи роботи із рекурсивними алгоритмами.