Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа №1

По дисциплине «Теоретические и множественные основы интеллектуальных систем»

Тема: «Множества. Работа с битовой маской»

Выполнил:

Студент 1 курса

Группы ИИ-21(1)

Кирилович А.А.

Проверил:

Глущенко Т. А.

Брест 2022

**Цель:** изучить некоторые алгоритмы на множествах, в частности алгоритм построения булеана (генерации всех подмножеств), алгоритм «слияние» для объединения, алгоритм «слияние» для сортировки.

**Ход работы**

Вариант 5

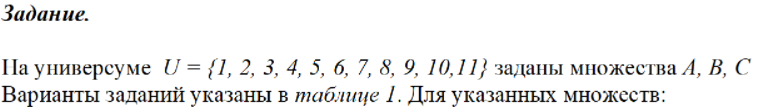


Таблица 1:



1. Для заданного множества A построить булеан.

A = [1, 3, 4, 7]

# создаем список для записи всех элементов булеана

C = [[] for i in range(2\*\*len(A))]

# создаем список двоичных чисел размеров 2 в степени n

B = [bin(x) for x in range(0, 2\*\*len(A))]

# проходим по всем двоичным числам

for i in range(0, 2\*\*len(A)):

    # форматирование двоичных чисел

    B[i] = B[i].replace('0b', '0'\*(len(A)-(len(B[i])-2)))

    # записываем элемент булеана в список

    for j in range(0, len(B[i])):

        if B[i][j] == '1':

            C[i].append(A[j])

print(C)

Out[1]: [], [7], [4], [4, 7], [3], [3, 7], [3, 4], [3, 4, 7], [1], [1, 7],

[1, 4], [1, 4, 7], [1, 3], [1, 3, 7], [1, 3, 4], [1, 3, 4, 7]

1. Реализовать на выбор сортировку слиянием или объединение слиянием. При сортировке слиянием на вход подается неупорядоченное множество (неотсортированный массив). Выход – упорядоченное множество (отсортированный по возрастанию массив). При объединении сливаем два упорядоченных массива.

Сортировка слиянием:

from typing import List

def merge\_two\_list(a:List, b:List):

    # инициализируем список для слияния

    c = []

    i = j = 0

    # сравниваем значение элемента с одного списка

    # со значением элемента с другого списка

    while i < len(a) and j < len(b):

        if a[i] < b[j]:

            c.append(a[i])

            i += 1

        else:

            c.append(b[j])

            j += 1

    # если списки были неодинаковой длины,

    # попадаем в одно из следующих условий

    if i < len(a):

        c += a[i:]

    if j < len(b):

        c += b[j:]

    return c

def merge\_sort(list):

    if len(list) == 1:

        return list

    # находим середину

    middle = len(list) // 2

    # делим список

    left = merge\_sort(list[:middle])

    right = merge\_sort(list[middle:])

    # сливаем списки и возвращаем их

    return merge\_two\_list(left, right)

# вводим список

list = [int(i) for i in input('Введите список: ').split(' ')]

# выводим список

print(merge\_sort(list))

Out[1]: Введите список:

In[1]: 34 3 5 1 6 7 89 44 4

Out[2]: [1, 3, 4, 5, 6, 7, 34, 44, 89]

1. Реализовать все операции над множествами: объединение, пересечение, разность, симметрическую разность, дополнение, используя битовую маску.
2. Вычислить программно, используя битовую маску, выражение согласно варианту, проиллюстрировать результат диаграммой Эйлера-Венна.

from typing import List

def bitmask(A:List):

    '''Создание битовой маски множества'''

    bitmask = [0 for i in range(len(U))]

    for i in range(len(A)):

        if U.index(A[i]) != -1:

            bitmask[U.index(A[i])] = 1

    return bitmask

def antibitmask(bitmask:List):

    '''Расшифровка битовой маски'''

    set = []

    for i in range(len(bitmask)):

        if bitmask[i] == 1:

            set.append(U[i])

    return set

def union(A:List, B:List):

    '''Объединение'''

    union = [0 for i in range(len(U))]

    a\_ = bitmask(A)

    b\_ = bitmask(B)

    for i in range(len(a\_)):

        union[i] = a\_[i] | b\_[i]

    union = antibitmask(union)

    return union

def intersection(A:List, B:List):

    '''Пересечение'''

    intersection = [0 for i in range(len(U))]

    a\_ = bitmask(A)

    b\_ = bitmask(B)

    for i in range(len(a\_)):

        intersection[i] = a\_[i] & b\_[i]

    intersection = antibitmask(intersection)

    return intersection

def diff(A:List, B:List):

    '''Разность'''

    diff = [0 for i in range(len(U))]

    a\_ = bitmask(A)

    b\_ = bitmask(B)

    for i in range(len(a\_)):

        diff[i] = a\_[i] - b\_[i]

    diff = antibitmask(diff)

    return diff

def sym\_diff(A:List, B:List):

    '''Симметрическая разность'''

    sym\_diff = [0 for i in range(len(U))]

    a\_ = bitmask(A)

    b\_ = bitmask(B)

    for i in range(len(a\_)):

        sym\_diff[i] = a\_[i] ^ b\_[i]

    sym\_diff = antibitmask(sym\_diff)

    return sym\_diff

def addition(A:List):

    '''"Нe"'''

    a\_ = bitmask(A)

    for i in range(len(a\_)):

        if a\_[i] == 0:

            a\_[i] = 1

        else:

            a\_[i] = 0

    A = antibitmask(a\_)

    return A

U = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]

A = [1, 3, 4, 7]

B = [3, 5, 6, 7, 8]

C = [2, 4, 5, 7]

print('\nОбъединение: ')

print(union(A, B))

print('\nПересечение: ')

print(intersection(A, B))

print('\nРазность А и В: ')

print(diff(A, B))

print('\nРазность В и А: ')

print(diff(B, A))

print('\nСимметрическая разность: ')

print(sym\_diff(A, B))

print('\nДополнение А: ')

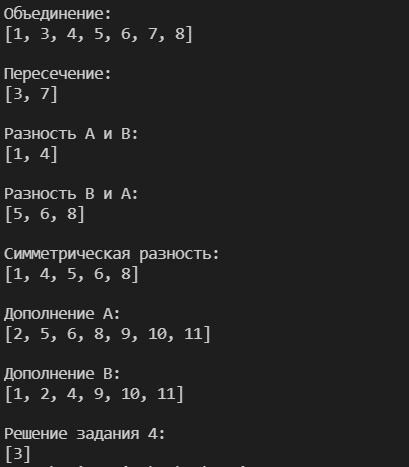
print(addition(A))

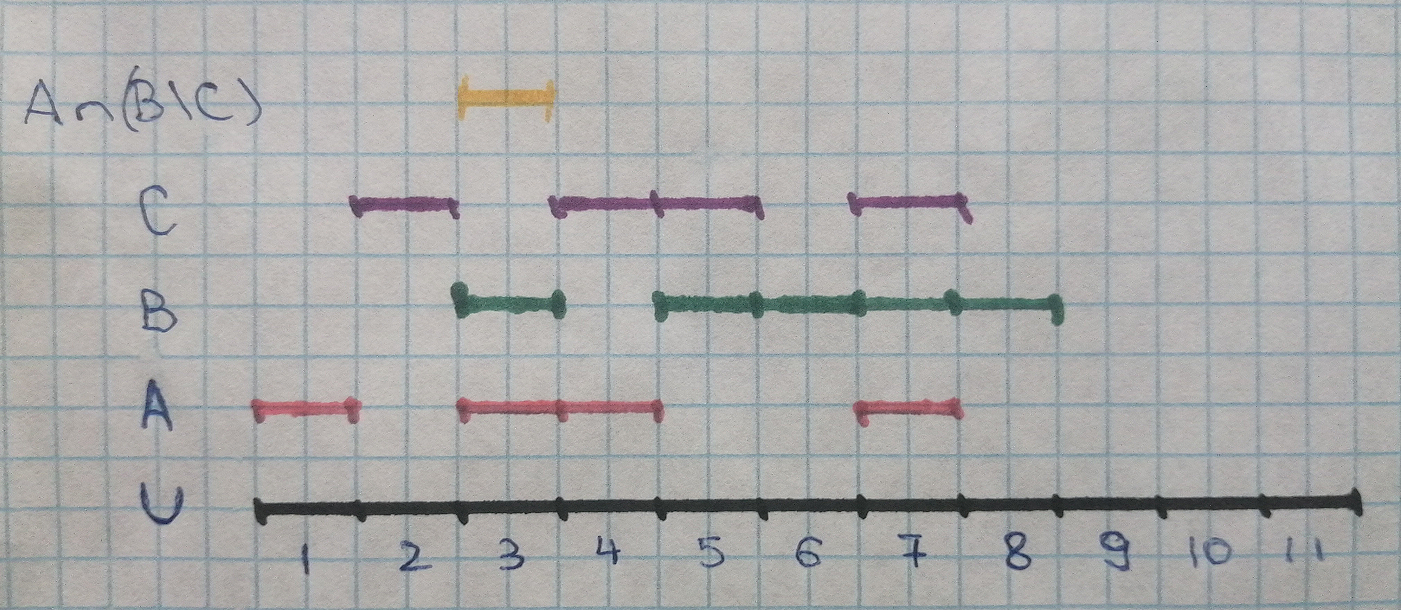
print('\nДополнение В: ')

print(addition(B))

print('\nРешение задания 4: ')

print(intersection(A, diff(B, C)))





1. Указать для номера вашего варианта (порядок генерации) бинарный код Грея.

101

010

--- XOR

111

Вывод: изучил алгоритмы «слияние» и построения булеана.