**СОДЕРЖАНИЕ**

[1 Реализация Алгоритма 2](#_Toc125590389)

[1.1 Постановка задачи 2](#_Toc125590390)

[1.1.1 Выбранные алгоритмы 2](#_Toc125590391)

[1.1.2 Задача о рюкзаке 2](#_Toc125590392)

[1.1.3 Генетический алгоритм 3](#_Toc125590393)

[**Приложение А Программная реализация алгоритма** 4](#_Toc125590394)

# Реализация Алгоритма

## Постановка задачи

Реализовать алгоритм для решения задачи распределения функций релейной защиты по терминалам с поставленными ограничениями.

Распределять по основной вычислительной нагрузке и количеству SV потоков.

В одном терминале не должно быть двух основных функций одного оборудования, в одном терминале не должно быть и основной и резервной функций одного оборудования.

Распределяться функции должны с максимальной заполняемостью терминала.

### Выбранные алгоритмы

Реализуемы алгоритм решения строится на решении задачи о рюкзаке.

В качестве используемого готового алгоритма для проверки выбран генетический алгоритм.

### Задача о рюкзаке

Задача о рюкзаке (англ. Knapsack problem) — дано N предметов, ni предмет имеет массу wi>0 и стоимость pi>0. Необходимо выбрать из этих предметов такой набор, чтобы суммарная масса не превосходила заданной величины W (вместимость рюкзака), а суммарная стоимость была максимальна.

В рамках поставленной задачи вместимость это вычислительная нагрузка и количество SV потоков, а стоимость будет вычислять по количеству добавленных функций.

### Генетический алгоритм

**Генетический алгоритм** (англ. genetic algorithm) — это эвристический алгоритм поиска, используемый для решения задач оптимизации и моделирования путём последовательного подбора, комбинирования и вариации искомых параметров с использованием механизмов, напоминающих биологическую эволюцию. Является разновидностью эволюционных вычислений (англ. evolutionary computation). Отличительной особенностью генетического алгоритма является акцент на использование оператора «скрещивания», который производит операцию рекомбинации решений-кандидатов, роль которой аналогична роли скрещивания в живой природе.

В качестве гена в нашей задаче будет функция, а хромосомой терминал.

Используется готовый программный модуль DEAP.

**Программная реализация алгоритма**

class Terminal:  
  
 def \_\_init\_\_(self, capacity, streams):  
 self.capacity = capacity  
 self.streams = streams  
 self.functions\_streams = set()  
 self.functions\_capacity = []  
 self.functions = []

class Function:  
  
 def \_\_init\_\_(self, capacity, streams, appointment, equipment):  
 self.capacity = capacity  
 self.streams = set(streams)  
 self.appointment = appointment  
 self.equipment = equipment

def process\_pack(terminals, functions):  
 functions\_list = [i for i in functions if isinstance(i, Function)]  
 terminals\_list = [i for i in terminals if isinstance(i, Terminal)]  
 calculations = dict()  
 while len(functions\_list) > 0:  
 for i in range(len(functions\_list)):  
 for j in range(len(terminals\_list)):  
 if sum(terminals\_list[j].functions\_capacity) + functions\_list[i].capacity <= \  
 terminals\_list[j].capacity and len(terminals\_list[j].functions\_streams) + len(  
 functions\_list[i].streams - terminals\_list[j].functions\_streams) <= \  
 terminals\_list[j].streams:  
 calculations[str(j) + "\_" + str(i)] = functions\_list[i].capacity + 5 \* len(  
 functions\_list[i].streams - terminals\_list[j].functions\_streams) + 100 \* (  
 [k.equipment for k in terminals\_list[j].functions if  
 isinstance(k, Function)].count(  
 functions\_list[i].equipment))  
 para = sorted(calculations.items(), key=lambda group: (int(group[1]), int(group[0])))[0]  
 calculations.clear()  
 now\_function = functions\_list.pop(int(para[0].split("\_")[1]))  
 now\_terminal = terminals\_list[int(para[0].split("\_")[0])]  
 now\_terminal.functions.append(now\_function)  
 now\_terminal.functions\_capacity.append(now\_function.capacity)  
 now\_terminal.functions\_streams.update(now\_function.streams - now\_terminal.functions\_streams)

Term1 = Terminal(100, 20)  
Term2 = Terminal(100, 20)  
Term3 = Terminal(100, 20)  
Term4 = Terminal(100, 20)  
Term5 = Terminal(100, 20)  
Term6 = Terminal(100, 20)  
Term7 = Terminal(100, 20)  
Term8 = Terminal(100, 20)  
Term9 = Terminal(100, 20)  
Term10 = Terminal(100, 20)  
Term11 = Terminal(100, 20)  
Term12 = Terminal(100, 20)  
Term13 = Terminal(100, 20)  
Term14 = Terminal(100, 20)  
Term15 = Terminal(100, 20)  
Term16 = Terminal(100, 20)  
Term17 = Terminal(100, 20)  
Term18 = Terminal(100, 20)  
Term19 = Terminal(100, 20)  
Term20 = Terminal(100, 20)  
Term21 = Terminal(100, 20)  
  
listFunctions = []  
listTerminals = []

# BDP = Function(20, ["IW2E", "IW2'E", "IW1E", "IT1E", "IT2E", "I1QCE", "I2QCE", "TV1E", "TV2E"], "main", "BUSE")  
# DZL1 = Function(15, ["IW2E", "DZL1'"], "main", "W2E")  
# DZL2 = Function(15, ["IW2'E", "DZL2'"], "main", "W2'E")  
# DZL3 = Function(15, ["IW1E", "DZL3'"], "main", "W1E")  
# KSZ1 = Function(10, ["IW2E", "TV1E"], "secondary", "W2E")  
# KSZ2 = Function(10, ["IW2'E", "TV2E"], "secondary", "W2'E")  
# KSZ3 = Function(10, ["IW1E", "TV1E"], "secondary", "W1E")  
# KSZ4 = Function(10, ["I1QCE", "TV1E", "TV2E"], "secondary", "QCE")  
# KSZ5 = Function(10, ["IT1E", "TV1E", "IT1K", "TV1K"], "secondary", "T1")  
# KSZ6 = Function(10, ["IT2E", "TV2E", "IT2K", "TV2K"], "secondary", "T2")  
# TDP1 = Function(15, ["IT1E", "IT1K"], "main", "T1")  
# TDP2 = Function(15, ["IT2E", "IT2K"], "main", "T2")  
# KSZ7 = Function(5, ["I1QCK", "TV1K", "TV2K"], "main", "QCK")  
# KSZ8 = Function(5, ["IW1K", "TV1K"], "main", "W1K")  
# KSZ9 = Function(5, ["IW2K", "TV1K"], "main", "W2K")  
# KSZ10 = Function(5, ["IW3K", "TV1K"], "main", "W3K")  
# KSZ11 = Function(5, ["IW4K", "TV1K"], "main", "W4K")  
# KSZ12 = Function(5, ["IW5K", "TV1K"], "main", "W5K")  
# KSZ13 = Function(5, ["IW6K", "TV1K"], "main", "W6K")  
# KSZ14 = Function(5, ["IW7K", "TV2K"], "main", "W7K")  
# KSZ15 = Function(5, ["IW8K", "TV2K"], "main", "W8K")  
# KSZ16 = Function(5, ["IW9K", "TV2K"], "main", "W9K")  
# KSZ17 = Function(5, ["IW10K", "TV2K"], "main", "W10K")  
# KSZ18 = Function(5, ["IW11K", "TV2K"], "main", "W11K")  
# KSZ19 = Function(5, ["IW12K", "TV2K"], "main", "W12K")  
# KSZ20 = Function(5, ["IT1K", "TV1K"], "main", "T1K")  
# KSZ21 = Function(5, ["IT2K", "TV2K"], "main", "T2K")  
#  
# listFunctions.append(BDP)  
# listFunctions.append(DZL1)  
# listFunctions.append(DZL2)  
# listFunctions.append(DZL3)  
# listFunctions.append(KSZ1)  
# listFunctions.append(KSZ2)  
# listFunctions.append(KSZ3)  
# listFunctions.append(KSZ4)  
# listFunctions.append(KSZ5)  
# listFunctions.append(KSZ6)  
# listFunctions.append(TDP1)  
# listFunctions.append(TDP2)  
# listFunctions.append(KSZ7)  
# listFunctions.append(KSZ8)  
# listFunctions.append(KSZ9)  
# listFunctions.append(KSZ10)  
# listFunctions.append(KSZ11)  
# listFunctions.append(KSZ12)  
# listFunctions.append(KSZ13)  
# listFunctions.append(KSZ14)  
# listFunctions.append(KSZ15)  
# listFunctions.append(KSZ16)  
# listFunctions.append(KSZ17)  
# listFunctions.append(KSZ18)  
# listFunctions.append(KSZ19)  
# listFunctions.append(KSZ20)  
# listFunctions.append(KSZ21)  
  
DZHK1C1 = Function(17, ["IW1C", "IW2C", "IAT1C"], "main", "K1C")  
DZHK1C2 = Function(17, ["IW1C", "IW2C", "IAT1C"], "secondary", "K1C")  
DZHK2C1 = Function(17, ["IW3C", "IW4C", "IAT2C"], "main", "K2C")  
DZHK2C2 = Function(17, ["IW3C", "IW4C", "IAT2C"], "secondary", "K2C")  
DZLW1C = Function(13, ["IW1C", "W1C-TV2"], "main", "W1C")  
DZLW2C = Function(13, ["IW2C", "W2C-TV2"], "main", "W2C")  
DZLW3C = Function(13, ["IW3C", "W3C-TV2"], "main", "W3C")  
DZLW4C = Function(13, ["IW4C", "W4C-TV2"], "main", "W4C")  
KSZRCW1C1 = Function(11, ["IW1W2C", "IW1C", "W1C-TV1"], "secondary", "W1C")  
KSZRCW1C2 = Function(11, ["IW1W2C", "IW1C", "W1C-TV2"], "secondary", "W1C")  
KSZRCW2C1 = Function(11, ["IW1W2C", "IW2C", "W2C-TV1"], "secondary", "W2C")  
KSZRCW2C2 = Function(11, ["IW1W2C", "IW2C", "W2C-TV2"], "secondary", "W2C")  
KSZRCW3C1 = Function(11, ["IW3W4C", "IW3C", "W3C-TV1"], "secondary", "W3C")  
KSZRCW3C2 = Function(11, ["IW3W4C", "IW3C", "W3C-TV2"], "secondary", "W3C")  
KSZRCW4C1 = Function(11, ["IW3W4C", "IW4C", "W4C-TV1"], "secondary", "W4C")  
KSZRCW4C2 = Function(11, ["IW3W4C", "IW4C", "W4C-TV2"], "secondary", "W4C")  
DZTAT1C1 = Function(12, ["IAT1C", "IAT1K", "IAT1E"], "main", "AT1")  
DZTAT1C2 = Function(12, ["IAT1C", "IAT1K", "IAT1E"], "main", "AT1")  
KSZAT1 = Function(9, ["IAT1C", "IAT1E", "K1C-TV1", "TV1E"], "secondary", "AT1")  
DZOAT1 = Function(9, ["IAT1K", "TV1K"], "secondary", "AT1")  
DZTAT2C1 = Function(12, ["IAT2C", "IAT2K", "IAT2E"], "main", "AT2")  
DZTAT2C2 = Function(12, ["IAT2C", "IAT2K", "IAT2E"], "main", "AT2")  
KSZAT2 = Function(9, ["IAT2C", "IAT2E", "K1C-TV2"], "secondary", "AT2")  
DZOAT2 = Function(9, ["IAT2K", "TV2K"], "secondary", "AT2")  
DZHK1E = Function(14, ["IQC4E", "IW8E", "IAT1E", "IW6E", "IQC1E"], "main", "K1E")  
DZHK2E = Function(14, ["IQC2E", "IW4E", "IAT2E", "IW3E", "IQC3E"], "main", "K2E")  
DZHK3E = Function(14, ["IQC1E", "IW2E", "IAT3E", "IW1E", "IQC3E"], "main", "K3E")  
DZHK4E = Function(14, ["IQC4E", "IW7E", "IAT4E", "IW5E", "IQC2E"], "main", "K4E")  
KSZQC1E = Function(9, ["IQC1E", "TV1E", "TV3E"], "main", "QC1E")  
KSZQC2E = Function(9, ["IQC2E", "TV4E", "TV2E"], "main", "QC2E")  
KSZQC3E = Function(9, ["IQC3E", "TV3E", "TV2E"], "main", "QC3E")  
KSZQC4E = Function(9, ["IQC4E", "TV1E", "TV4E"], "main", "QC4E")  
DFZW1E = Function(16, ["IW1E", "TV3E"], "main", "W1E")  
DFZW2E = Function(16, ["IW2E", "TV3E"], "main", "W2E")  
DFZW3E = Function(16, ["IW3E", "TV2E"], "main", "W3E")  
DFZW4E = Function(16, ["IW4E", "TV2E"], "main", "W4E")  
DFZW5E = Function(16, ["IW5E", "TV4E"], "main", "W5E")  
DFZW6E = Function(16, ["IW6E", "TV1E"], "main", "W6E")  
DFZW7E = Function(16, ["IW7E", "TV4E"], "main", "W7E")  
DFZW8E = Function(16, ["IW8E", "TV1E"], "main", "W8E")  
KSZRCW1E = Function(11, ["IW1E", "TV3E"], "main", "W1E")  
DZLW2E = Function(13, ["IW2E", "TV3E"], "main", "W2E")  
DZLW3E = Function(13, ["IW3E", "TV2E"], "main", "W3E")  
KSZRCW4E = Function(11, ["IW4E", "TV2E"], "main", "W4E")  
KSZRCW5E = Function(11, ["IW5E", "TV4E"], "main", "W5E")  
KSZRCW6E = Function(11, ["IW6E", "TV1E"], "main", "W6E")  
KSZRCW7E = Function(11, ["IW7E", "TV4E"], "main", "W7E")  
KSZRCW8E = Function(11, ["IW8E", "TV1E"], "main", "W8E")  
DZTAT3C1 = Function(12, ["IAT3E", "IAT3G", "IAT3K"], "main", "AT3")  
DZTAT3C2 = Function(12, ["IAT3E", "IAT3G", "IAT3K"], "main", "AT3")  
DZTAT4C1 = Function(12, ["IAT4E", "IAT4G", "IAT4K"], "main", "AT4")  
DZTAT4C2 = Function(12, ["IAT4E", "IAT4G", "IAT4K"], "main", "AT4")  
KSZAT3 = Function(9, ["IAT3E", "IAT3G", "TV3E", "TV1G"], "secondary", "AT3")  
DZOAT3 = Function(9, ["IAT3K", "TV3K"], "secondary", "AT3")  
KSZAT4 = Function(9, ["IAT4E", "IAT4G", "TV4E", "TV2G"], "secondary", "AT4")  
DZOAT4 = Function(9, ["IAT4K", "TV4K"], "secondary", "AT4")  
DZHK1G = Function(14, ["IAT3G", "IW1G", "IW3G", "IW5G", "IQC1G"], "main", "K1G")  
DZHK2G = Function(14, ["IAT4G", "IW2G", "IW4G", "IW5G", "IQC1G"], "main", "K2G")  
KSZQC1G = Function(9, ["IQC1G", "TV1G", "TV2G"], "main", "QC1G")  
DFZW1G1 = Function(16, ["IW1G", "TV1G"], "main", "W1G")  
DFZW2G1 = Function(16, ["IW2G", "TV2G"], "main", "W2G")  
DFZW3G1 = Function(16, ["IW3G", "TV1G"], "main", "W3G")  
DFZW4G1 = Function(16, ["IW4G", "TV2G"], "main", "W4G")  
DFZW5G1 = Function(16, ["IW5G", "TV1G"], "main", "W5G")  
DFZW6G1 = Function(16, ["IW6G", "TV2G"], "main", "W6G")  
DFZW1G2 = Function(16, ["IW1G", "TV1G"], "secondary", "W1G")  
DFZW2G2 = Function(16, ["IW2G", "TV2G"], "secondary", "W2G")  
DFZW3G2 = Function(16, ["IW3G", "TV1G"], "secondary", "W3G")  
DFZW4G2 = Function(16, ["IW4G", "TV2G"], "secondary", "W4G")  
DFZW5G2 = Function(16, ["IW5G", "TV1G"], "secondary", "W5G")  
DFZW6G2 = Function(16, ["IW6G", "TV2G"], "secondary", "W6G")  
KSZTN1 = Function(6, ["ITN1", "TV1K"], "main", "TN1")  
KSZTN2 = Function(6, ["ITN2", "TV2K"], "main", "TN2")  
KSZTN3 = Function(6, ["ITN3", "TV1K"], "main", "TN3")  
KSZTN4 = Function(6, ["ITN4", "TV2K"], "main", "TN4")  
KSZQC1K = Function(9, ["IQC1K", "TV1K", "TV2K"], "main", "QC1K")  
KSZQC4K = Function(9, ["IQC4K", "TV3K", "TV4K"], "main", "QC4K")  
KSZW412K = Function(7, ["IW412K", "TV4K"], "main", "W412K")  
KSZW410K = Function(7, ["IW410K", "TV4K"], "main", "W410K")  
KSZW409K = Function(7, ["IW409K", "TV4K"], "main", "W409K")  
KSZW408K = Function(7, ["IW408K", "TV4K"], "main", "W408K")  
KSZW407K = Function(7, ["IW407K", "TV4K"], "main", "W407K")  
KSZW405K = Function(7, ["IW405K", "TV4K"], "main", "W405K")  
KSZW404K = Function(7, ["IW404K", "TV4K"], "main", "W404K")  
KSZW303K = Function(7, ["IW303K", "TV3K"], "main", "W303K")  
KSZW304K = Function(7, ["IW304K", "TV3K"], "main", "W304K")  
KSZW305K = Function(7, ["IW305K", "TV3K"], "main", "W305K")  
KSZW306K = Function(7, ["IW306K", "TV3K"], "main", "W306K")  
KSZW307K = Function(7, ["IW307K", "TV3K"], "main", "W307K")  
KSZW309K = Function(7, ["IW309K", "TV3K"], "main", "W309K")  
KSZW310K = Function(7, ["IW310K", "TV3K"], "main", "W310K")  
KSZW312K = Function(7, ["IW312K", "TV3K"], "main", "W312K")  
KSZAT1K = Function(9, ["IAT1K", "TV1K"], "secondary", "AT1")  
KSZAT2K = Function(9, ["IAT2K", "TV2K"], "secondary", "AT2")  
KSZAT3K = Function(9, ["IAT3K", "TV3K"], "secondary", "AT3")  
KSZAT4K = Function(9, ["IAT4K", "TV4K"], "secondary", "AT4")  
  
listTerminals.append(Term1)  
listTerminals.append(Term2)  
listTerminals.append(Term3)  
listTerminals.append(Term4)  
listTerminals.append(Term5)  
listTerminals.append(Term6)  
listTerminals.append(Term7)  
listTerminals.append(Term8)  
listTerminals.append(Term9)  
listTerminals.append(Term10)  
listTerminals.append(Term11)  
listTerminals.append(Term12)  
listTerminals.append(Term13)  
listTerminals.append(Term14)  
listTerminals.append(Term15)  
listTerminals.append(Term16)  
listTerminals.append(Term17)  
listTerminals.append(Term18)  
listTerminals.append(Term19)  
listTerminals.append(Term20)  
listTerminals.append(Term21)  
  
listFunctions.append(DZHK1C1)  
listFunctions.append(DZHK1C2)  
listFunctions.append(DZHK2C1)  
listFunctions.append(DZHK2C2)  
listFunctions.append(DZLW1C)  
listFunctions.append(DZLW2C)  
listFunctions.append(DZLW3C)  
listFunctions.append(DZLW4C)  
listFunctions.append(KSZRCW1C1)  
listFunctions.append(KSZRCW1C2)  
listFunctions.append(KSZRCW2C1)  
listFunctions.append(KSZRCW2C2)  
listFunctions.append(KSZRCW3C1)  
listFunctions.append(KSZRCW3C2)  
listFunctions.append(KSZRCW4C1)  
listFunctions.append(KSZRCW4C2)  
listFunctions.append(DZTAT1C1)  
listFunctions.append(DZTAT1C2)  
listFunctions.append(KSZAT1)  
listFunctions.append(DZOAT1)  
listFunctions.append(DZTAT2C1)  
listFunctions.append(DZTAT2C2)  
listFunctions.append(KSZAT2)  
listFunctions.append(DZOAT2)  
listFunctions.append(DZHK1E)  
listFunctions.append(DZHK2E)  
listFunctions.append(DZHK3E)  
listFunctions.append(DZHK4E)  
listFunctions.append(KSZQC1E)  
listFunctions.append(KSZQC2E)  
listFunctions.append(KSZQC3E)  
listFunctions.append(KSZQC4E)  
listFunctions.append(DFZW1E)  
listFunctions.append(DFZW2E)  
listFunctions.append(DFZW3E)  
listFunctions.append(DFZW4E)  
listFunctions.append(DFZW5E)  
listFunctions.append(DFZW6E)  
listFunctions.append(DFZW7E)  
listFunctions.append(DFZW8E)  
listFunctions.append(KSZRCW1E)  
listFunctions.append(DZLW2E)  
listFunctions.append(DZLW3E)  
listFunctions.append(KSZRCW4E)  
listFunctions.append(KSZRCW5E)  
listFunctions.append(KSZRCW6E)  
listFunctions.append(KSZRCW7E)  
listFunctions.append(KSZRCW8E)  
listFunctions.append(DZTAT3C1)  
listFunctions.append(DZTAT3C2)  
listFunctions.append(DZTAT4C1)  
listFunctions.append(DZTAT4C2)  
listFunctions.append(KSZAT3)  
listFunctions.append(DZOAT3)  
listFunctions.append(KSZAT4)  
listFunctions.append(DZOAT4)  
listFunctions.append(DZHK1G)  
listFunctions.append(DZHK2G)  
listFunctions.append(KSZQC1G)  
listFunctions.append(DFZW1G1)  
listFunctions.append(DFZW2G1)  
listFunctions.append(DFZW3G1)  
listFunctions.append(DFZW4G1)  
listFunctions.append(DFZW5G1)  
listFunctions.append(DFZW6G1)  
listFunctions.append(DFZW1G2)  
listFunctions.append(DFZW2G2)  
listFunctions.append(DFZW3G2)  
listFunctions.append(DFZW4G2)  
listFunctions.append(DFZW5G2)  
listFunctions.append(DFZW6G2)  
listFunctions.append(KSZTN1)  
listFunctions.append(KSZTN2)  
listFunctions.append(KSZTN3)  
listFunctions.append(KSZTN4)  
listFunctions.append(KSZQC1K)  
listFunctions.append(KSZQC4K)  
listFunctions.append(KSZW412K)  
listFunctions.append(KSZW410K)  
listFunctions.append(KSZW409K)  
listFunctions.append(KSZW408K)  
listFunctions.append(KSZW407K)  
listFunctions.append(KSZW405K)  
listFunctions.append(KSZW404K)  
listFunctions.append(KSZW303K)  
listFunctions.append(KSZW304K)  
listFunctions.append(KSZW305K)  
listFunctions.append(KSZW306K)  
listFunctions.append(KSZW307K)  
listFunctions.append(KSZW309K)  
listFunctions.append(KSZW310K)  
listFunctions.append(KSZW312K)  
listFunctions.append(KSZAT1K)  
listFunctions.append(KSZAT2K)  
listFunctions.append(KSZAT3K)  
listFunctions.append(KSZAT4K)  
  
process\_pack(listTerminals, listFunctions)  
  
for j in listTerminals:  
 print([i.equipment + " " + i.appointment for i in j.functions])  
 print(sum(j.functions\_capacity))  
 print(len(j.functions\_streams))  
  
print(len(listTerminals))  
o = 0  
for i in listTerminals:  
 if sum(i.functions\_capacity) > 0:  
 o = o + 1  
print(o)  
print(sum([len(k.functions) for k in listTerminals]))  
print(len(listFunctions))  
print(sum([k.capacity for k in listFunctions]))  
print(sum([sum(m.functions\_capacity) for m in listTerminals]))

IND\_INIT\_SIZE = 5  
MAX\_ITEM = 50  
MAX\_WEIGHT = 50  
NBR\_ITEMS = 20  
  
items = {0: (17, {"IW1C", "IW2C", "IAT1C"}, "main", "K1C"),  
 1: (17, {"IW1C", "IW2C", "IAT1C"}, "secondary", "K1C"),  
 2: (17, {"IW3C", "IW4C", "IAT2C"}, "main", "K2C"),  
 3: (17, {"IW3C", "IW4C", "IAT2C"}, "secondary", "K2C"),  
 4: (13, {"IW1C", "W1C-TV2"}, "main", "W1C"),  
 5: (13, {"IW2C", "W2C-TV2"}, "main", "W2C"),  
 6: (13, {"IW3C", "W3C-TV2"}, "main", "W3C"),  
 7: (13, {"IW4C", "W4C-TV2"}, "main", "W4C"),  
 8: (11, {"IW1W2C", "IW1C", "W1C-TV1"}, "secondary", "W1C"),  
 9: (11, {"IW1W2C", "IW1C", "W1C-TV2"}, "secondary", "W1C"),  
 10: (11, {"IW1W2C", "IW2C", "W2C-TV1"}, "secondary", "W2C"),  
 11: (11, {"IW1W2C", "IW2C", "W2C-TV2"}, "secondary", "W2C"),  
 12: (11, {"IW3W4C", "IW3C", "W3C-TV1"}, "secondary", "W3C"),  
 13: (11, {"IW3W4C", "IW3C", "W3C-TV2"}, "secondary", "W3C"),  
 14: (11, {"IW3W4C", "IW4C", "W4C-TV1"}, "secondary", "W4C"),  
 15: (11, {"IW3W4C", "IW4C", "W4C-TV2"}, "secondary", "W4C"),  
 16: (12, {"IAT1C", "IAT1K", "IAT1E"}, "main", "AT1"),  
 17: (12, {"IAT1C", "IAT1K", "IAT1E"}, "main", "AT1"),  
 18: (9, {"IAT1C", "IAT1E", "K1C-TV1", "TV1E"}, "secondary", "AT1"),  
 19: (9, {"IAT1K", "TV1K"}, "secondary", "AT1"),  
 20: (12, {"IAT2C", "IAT2K", "IAT2E"}, "main", "AT2"),  
 21: (12, {"IAT2C", "IAT2K", "IAT2E"}, "main", "AT2"),  
 22: (9, {"IAT2C", "IAT2E", "K1C-TV2"}, "secondary", "AT2"),  
 23: (9, {"IAT2K", "TV2K"}, "secondary", "AT2"),  
 24: (14, {"IQC4E", "IW8E", "IAT1E", "IW6E", "IQC1E"}, "main", "K1E"),  
 25: (14, {"IQC2E", "IW4E", "IAT2E", "IW3E", "IQC3E"}, "main", "K2E"),  
 26: (14, {"IQC1E", "IW2E", "IAT3E", "IW1E", "IQC3E"}, "main", "K3E"),  
 27: (14, {"IQC4E", "IW7E", "IAT4E", "IW5E", "IQC2E"}, "main", "K4E"),  
 28: (9, {"IQC1E", "TV1E", "TV3E"}, "main", "QC1E"),  
 29: (9, {"IQC2E", "TV4E", "TV2E"}, "main", "QC2E"),  
 30: (9, {"IQC3E", "TV3E", "TV2E"}, "main", "QC3E"),  
 31: (9, {"IQC4E", "TV1E", "TV4E"}, "main", "QC4E"),  
 32: (16, {"IW1E", "TV3E"}, "main", "W1E"),  
 33: (16, {"IW2E", "TV3E"}, "main", "W2E"),  
 34: (16, {"IW3E", "TV2E"}, "main", "W3E"),  
 35: (16, {"IW4E", "TV2E"}, "main", "W4E"),  
 36: (16, {"IW5E", "TV4E"}, "main", "W5E"),  
 37: (16, {"IW6E", "TV1E"}, "main", "W6E"),  
 38: (16, {"IW7E", "TV4E"}, "main", "W7E"),  
 39: (16, {"IW8E", "TV1E"}, "main", "W8E"),  
 40: (11, {"IW1E", "TV3E"}, "main", "W1E"),  
 41: (13, {"IW2E", "TV3E"}, "main", "W2E"),  
 42: (13, {"IW3E", "TV2E"}, "main", "W3E"),  
 43: (11, {"IW4E", "TV2E"}, "main", "W4E"),  
 44: (11, {"IW5E", "TV4E"}, "main", "W5E"),  
 45: (11, {"IW6E", "TV1E"}, "main", "W6E"),  
 46: (11, {"IW7E", "TV4E"}, "main", "W7E"),  
 47: (11, {"IW8E", "TV1E"}, "main", "W8E"),  
 48: (12, {"IAT3E", "IAT3G", "IAT3K"}, "main", "AT3"),  
 49: (12, {"IAT3E", "IAT3G", "IAT3K"}, "main", "AT3"),  
 50: (12, {"IAT4E", "IAT4G", "IAT4K"}, "main", "AT4"),  
 51: (12, {"IAT4E", "IAT4G", "IAT4K"}, "main", "AT4"),  
 52: (9, {"IAT3E", "IAT3G", "TV3E", "TV1G"}, "secondary", "AT3"),  
 53: (9, {"IAT3K", "TV3K"}, "secondary", "AT3"),  
 54: (9, {"IAT4E", "IAT4G", "TV4E", "TV2G"}, "secondary", "AT4"),  
 55: (9, {"IAT4K", "TV4K"}, "secondary", "AT4"),  
 56: (14, {"IAT3G", "IW1G", "IW3G", "IW5G", "IQC1G"}, "main", "K1G"),  
 57: (14, {"IAT4G", "IW2G", "IW4G", "IW5G", "IQC1G"}, "main", "K2G"),  
 58: (9, {"IQC1G", "TV1G", "TV2G"}, "main", "QC1G"),  
 59: (16, {"IW1G", "TV1G"}, "main", "W1G"),  
 60: (16, {"IW2G", "TV2G"}, "main", "W2G"),  
 61: (16, {"IW3G", "TV1G"}, "main", "W3G"),  
 62: (16, {"IW4G", "TV2G"}, "main", "W4G"),  
 63: (16, {"IW5G", "TV1G"}, "main", "W5G"),  
 64: (16, {"IW6G", "TV2G"}, "main", "W6G"),  
 65: (16, {"IW1G", "TV1G"}, "secondary", "W1G"),  
 66: (16, {"IW2G", "TV2G"}, "secondary", "W2G"),  
 67: (16, {"IW3G", "TV1G"}, "secondary", "W3G"),  
 68: (16, {"IW4G", "TV2G"}, "secondary", "W4G"),  
 69: (16, {"IW5G", "TV1G"}, "secondary", "W5G"),  
 70: (16, {"IW6G", "TV2G"}, "secondary", "W6G"),  
 71: (6, {"ITN1", "TV1K"}, "main", "TN1"),  
 72: (6, {"ITN2", "TV2K"}, "main", "TN2"),  
 73: (6, {"ITN3", "TV1K"}, "main", "TN3"),  
 74: (6, {"ITN4", "TV2K"}, "main", "TN4"),  
 75: (9, {"IQC1K", "TV1K", "TV2K"}, "main", "QC1K"),  
 76: (9, {"IQC4K", "TV3K", "TV4K"}, "main", "QC4K"),  
 77: (7, {"IW412K", "TV4K"}, "main", "W412K"),  
 78: (7, {"IW410K", "TV4K"}, "main", "W410K"),  
 79: (7, {"IW409K", "TV4K"}, "main", "W409K"),  
 80: (7, {"IW408K", "TV4K"}, "main", "W408K"),  
 81: (7, {"IW407K", "TV4K"}, "main", "W407K"),  
 82: (7, {"IW405K", "TV4K"}, "main", "W405K"),  
 83: (7, {"IW404K", "TV4K"}, "main", "W404K"),  
 84: (7, {"IW303K", "TV3K"}, "main", "W303K"),  
 85: (7, {"IW304K", "TV3K"}, "main", "W304K"),  
 86: (7, {"IW305K", "TV3K"}, "main", "W305K"),  
 87: (7, {"IW306K", "TV3K"}, "main", "W306K"),  
 88: (7, {"IW307K", "TV3K"}, "main", "W307K"),  
 89: (7, {"IW309K", "TV3K"}, "main", "W309K"),  
 90: (7, {"IW310K", "TV3K"}, "main", "W310K"),  
 91: (7, {"IW312K", "TV3K"}, "main", "W312K"),  
 92: (9, {"IAT1K", "TV1K"}, "secondary", "AT1"),  
 93: (9, {"IAT2K", "TV2K"}, "secondary", "AT2"),  
 94: (9, {"IAT3K", "TV3K"}, "secondary", "AT3"),  
 95: (9, {"IAT4K", "TV4K"}, "secondary", "AT4")  
 }  
  
items1 = items.copy()  
  
creator.create("Fitness", base.Fitness, weights=(1.0, 1.0, 0.00000001, 0.00000001))  
creator.create("Individual", set, fitness=creator.Fitness)  
  
toolbox = base.Toolbox()  
  
toolbox.register("attr\_item", random.choice, list(items1.keys()))  
  
toolbox.register("individual", tools.initRepeat, creator.Individual,  
 toolbox.attr\_item, IND\_INIT\_SIZE)  
toolbox.register("population", tools.initRepeat, list, toolbox.individual)  
  
  
def evalKnapsack(individual):  
 eqpfl = False  
 capacity = 0  
 equipment = []  
 streams = set()  
 for item in individual:  
 if item in items1:  
 capacity += int(items1[item][0])  
 if items1[item][3] in equipment:  
 eqpfl = True  
 break  
 else:  
 equipment.append(items1[item][3])  
 streams.update(items1[item][1])  
 if len(streams) > 20 or capacity > 100 or len(individual) > 14 or eqpfl:  
 return 0, 0, 0, 0  
 return capacity, len(streams), 0, 0  
  
  
def cxSet(ind1, ind2):  
 temp = set(ind1)  
 ind1 &= ind2  
 ind2 ^= temp  
 return ind1, ind2  
  
  
def mutSet(individual):  
 if random.random() < 0.5:  
 if len(individual) > 0:  
 individual.remove(random.choice(sorted(tuple(individual))))  
 else:  
 individual.add(random.choice(list(items1.keys())))  
 return individual,  
  
  
toolbox.register("evaluate", evalKnapsack)  
toolbox.register("mate", cxSet)  
toolbox.register("mutate", mutSet)  
toolbox.register("select", tools.selNSGA2)  
  
  
def main():  
 # random.seed(32)  
 NGEN = 50  
 MU = 50  
 LAMBDA = 100  
 CXPB = 0.7  
 MUTPB = 0.2  
  
 pop = toolbox.population(n=MU)  
 hof = tools.ParetoFront()  
  
 population, logbook = algorithms.eaMuPlusLambda(pop, toolbox, MU, LAMBDA, CXPB, MUTPB, NGEN,  
 halloffame=hof)  
  
 return population, logbook, hof  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 result = []  
 while len(items1) > 0:  
 capacity = 0  
 streams = set()  
 pop, logbook, hof = main()  
 for item in hof.items[0]:  
 if item in items:  
 capacity += int(items[item][0])  
 streams.update(items[item][1])  
 if capacity > 100 or len(streams) > 20:  
 continue  
 result.append(hof.items[0])  
 for i in hof.items[0]:  
 if i in items1:  
 del items1[i]  
 print(len(result))  
 for i in result:  
 capacity = 0  
 streams = set()  
 functions = []  
 print(i)  
 for item in i:  
 if item in items:  
 functions.append(items[item][3] + " " + items[item][2])  
 capacity += int(items[item][0])  
 streams.update(items[item][1])  
 print(functions)  
 print(capacity)  
 print(len(streams))