Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования «Пермский национальный исследовательский политехнический университет» (ПНИПУ)

Кафедра вычислительной математики и механики

**Лабораторная работа № 4**

**по дисциплине: «Интеллектуальные ИСИТ»**

Выполнил

студент группы ИСТ-19-2б

Кошкаров П.С.

Проверил

ассистент кафедры ВММБ

Нетбай Г.В.

Пермь, 2022

**Методы голосования при принятии коллективных решений:** Правило голосования решает задачу принятия коллективного решения: несколько агентов (выборщиков) осуществляют выбор из нескольких исходов (кандидатов). Наиболее популярные модели, решающие задачу принятия коллективного решения:

1. Модель Кондорсе. Победителем по Кондорсе называется кандидат a, который побеждает любого другого кандидата при попарном сравнении по правилу большинства (т.е. число выборщиков, считающих, что a лучше b, больше, чем число выборщиков, считающих, что b лучше a). Однако победитель по Кондорсе может не существовать.

Для определения победителя может применяться одно из возможных правил:

• Правило Копленда. Сравним кандидата a с любым другим кандидатом x. Начислим ему +1, если для большинства a лучше x, −1, если для большинства x лучше a, и 0 при равенстве. Суммируя общее количество очков по всем x, x не равно a, получаем оценку Копленда для a. Избирается кандидат с наивысшей из таких оценок.

• Правило Симпсона. Рассмотрим кандидата a, любого другого кандидата x и обозначим через N(a, x) число выборщиков, для которых a лучше x. Оценкой Симпсона для a называется минимальное из чисел N(a, x) по всем x, x не равно a. Избирается кандидат с наивысшей из таких оценок.

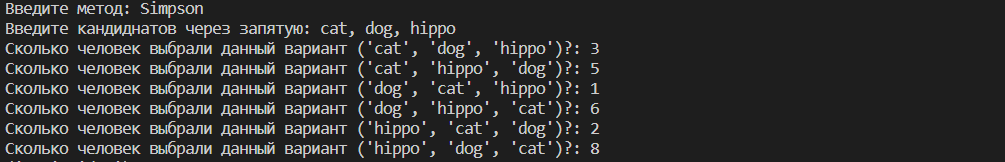
1. Модель Борда. Каждый выборщик объявляет свои предпочтения, ранжируя p кандидатов от лучшего к худшему (безразличие запрещается). Кандидат не получает очков за последнее место, получает p − 1 очков за первое место. Побеждает кандидат с наибольшей суммой очков.

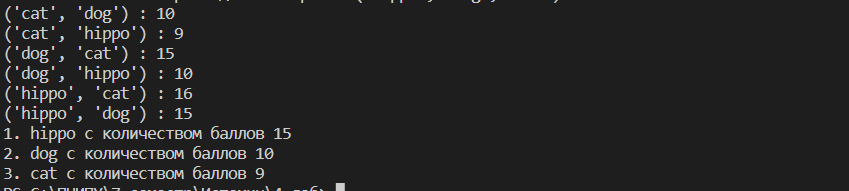
**Цель:** построить ЭС поддержки принятия коллективных решений.

**Описание:** в рамках данной работы необходимо реализовать систему, включающую вышеописанные модели принятия коллективных решений.

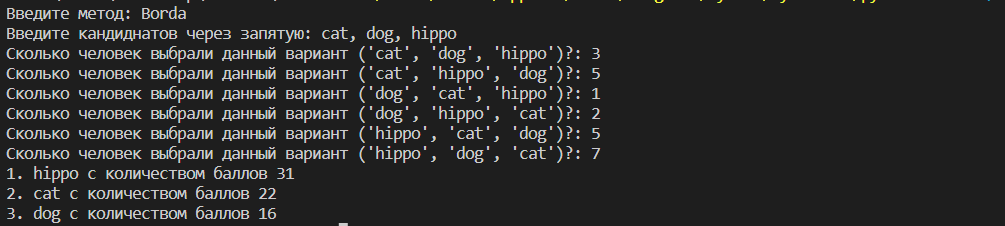
**Пример работы программы:**

1. Правило симпсона:





1. Метод Борда:



# **Листинг кода Python:**

**Файл main.py.**

import itertools

import Simpson

import Borda

def main():

method = input("Введите метод: ")

if method == "Simpson":

candidates, options, votes = read\_candidates\_votes()

pairs, points = Simpson.count\_pairs(candidates, options, votes)

score = Simpson.calculate\_score(candidates, pairs, points)

elif method == "Borda":

candidates, options, votes = read\_candidates\_votes()

score = Borda.count\_points(candidates, options, votes)

else:

print("Выход из программы...")

print\_final\_answer(score)

def read\_candidates\_votes():

user\_input = input("Введите кандиднатов через запятую: ")

candidates = [candidate for candidate in user\_input.split(', ')]

kolvo\_candidates = len(candidates)

options = list(itertools.permutations(candidates))

votes = []

for option in options:

text = f"Сколько человек выбрали данный вариант {option}?: "

votes.append(int(input(text)))

return (candidates, options, votes)

def print\_final\_answer(score):

score = dict(sorted(score.items(), key=lambda item: item[1], reverse=True))

for i in range(len(score)):

record = list(score.items())[i]

print(f"{(i+1)}. {record[0]} с количеством баллов {record[1]}")

if \_\_name\_\_=='\_\_main\_\_':

main()

**Файл Simpson.py.**

import itertools

def count\_pairs(candidates, options, votes):

pairs = list(itertools.permutations(candidates, 2))

points = [0 for \_ in pairs]

for i in range(len(pairs)):

for j in range(len(options)):

if (options[j].index(pairs[i][0]) < options[j].index(pairs[i][1])):

points[i] += votes[j]

for i in range(len(points)):

print(f"{pairs[i]} : {points[i]}")

return (pairs, points)

def calculate\_score(candidates, pairs, points):

candidate\_score = {}

for candidate in candidates:

match\_pairs = list(filter(lambda pair: pair[0]==candidate, pairs))

match\_indexes = [pairs.index(i) for i in match\_pairs]

match\_points = [points[i] for i in match\_indexes]

value = min(match\_points)

candidate\_score[candidate] = value

return candidate\_score

**Файл Borda.py.**

def count\_points(candidates, options, votes):

score = {candidate: 0 for candidate in candidates}

for candidate in candidates:

for i in range(len(options)):

value = (len(candidates) - options[i].index(candidate) - 1)\*votes[i]

score[candidate] += value

score = dict(sorted(score.items(), key=lambda item: item[1], reverse=True))

return score