Міністерство освіти і науки України
Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського"
Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Звіт

з лабораторної роботи № 4 з дисципліни
«Проектування алгоритмів»
„Проектування і аналіз алгоритмів для вирішення NP-складних задач ч.1”
Варіант 25

Виконав(ла) *ІП-15, Тонконог Владислав* _____
(шифр, прізвище, ім'я, по батькові)

Перевірив *Ахаладзе Ілля Елдарійович* _____
(прізвище, ім'я, по батькові)

Київ 2022

ЗМІСТ

1 МЕТА ЛАБОРАТОРНОЇ РОБОТИ

Мета роботи – вивчити основні підходи формалізації метаевристичних алгоритмів і вирішення типових задач з їхньою допомогою.

2 ЗАВДАННЯ

Згідно варіанту, розробити алгоритм вирішення задачі і виконати його програмну реалізацію на будь-якій мові програмування.

Задача, алгоритм і його параметри наведені в таблиці 2.1.

Зафіксувати якість отриманого розв'язку (значення цільової функції) після кожних 20 ітерацій до 1000 і побудувати графік залежності якості розв'язку від числа ітерацій.

Зробити узагальнений висновок.

Таблиця 2.1 – Варіанти алгоритмів

№	**Задача і алгоритм**
1	Задача про рюкзак (місткість P=250, 100 предметів, цінність предметів від 2 до 20 (випадкова), вага від 1 до 10 (випадкова)), генетичний алгоритм (початкова популяція 100 осіб кожна по 1 різному предмету, оператор схрещування одноточковий по 50 генів, мутація з ймовірністю 5% змінюємо тільки 1 випадковий ген). Розробити власний оператор локального покращення.
2	Задача комівояжера (100 вершин, відстань між вершинами випадкова від 5 до 50), мурашиний алгоритм ($\alpha = 2$, $\beta = 4$, $\rho = 0{,}4$, Lmin знайти жадібним алгоритмом, кількість мурах M = 30, починають маршрут в різних випадкових вершинах).
3	Задача розфарбовування графу (200 вершин, степінь вершини не більше 20, але не менше 1), бджолиний алгоритм ABC (число бджіл 30 із них 2 розвідники).
4	Задача про рюкзак (місткість P=200, 100 предметів, цінність предметів від 2 до 20 (випадкова), вага від 1 до 10 (випадкова)), генетичний алгоритм (початкова популяція 100 осіб кожна по 1 різному предмету, оператор схрещування двоточковий порівну генів, мутація з ймовірністю 10% змінюємо тільки 1 випадковий ген). Розробити

	власний оператор локального покращення.
5	Задача комівояжера (150 вершин, відстань між вершинами випадкова від 5 до 50), мурашиний алгоритм (α = 2, β = 3, ρ = 0,4, Lmin знайти жадібним алгоритмом, кількість мурах M = 35, починають маршрут в різних випадкових вершинах).
6	Задача розфарбовування графу (250 вершин, степінь вершини не більше 25, але не менше 2), бджолиний алгоритм ABC (число бджіл 35 із них 3 розвідники).
7	Задача про рюкзак (місткість P=150, 100 предметів, цінність предметів від 2 до 10 (випадкова), вага від 1 до 5 (випадкова)), генетичний алгоритм (початкова популяція 100 осіб кожна по 1 різному предмету, оператор схрещування рівномірний, мутація з ймовірністю 5% два випадкові гени міняються місцями). Розробити власний оператор локального покращення.
8	Задача комівояжера (200 вершин, відстань між вершинами випадкова від 0(перехід заборонено) до 50), мурашиний алгоритм (α = 3, β = 2, ρ = 0,3, Lmin знайти жадібним алгоритмом, кількість мурах M = 45, починають маршрут в різних випадкових вершинах).
9	Задача розфарбовування графу (150 вершин, степінь вершини не більше 30, але не менше 1), бджолиний алгоритм ABC (число бджіл 25 із них 3 розвідники).
10	Задача про рюкзак (місткість P=150, 100 предметів, цінність предметів від 2 до 10 (випадкова), вага від 1 до 5 (випадкова)), генетичний алгоритм (початкова популяція 100 осіб кожна по 1 різному предмету, оператор схрещування рівномірний, мутація з ймовірністю 10% два випадкові гени міняються місцями). Розробити власний оператор локального покращення.
11	Задача комівояжера (250 вершин, відстань між вершинами випадкова від 0(перехід заборонено) до 50), мурашиний алгоритм (α = 2, β = 4, ρ =

	0,6, Lmin знайти жадібним алгоритмом, кількість мурах M = 45, починають маршрут в різних випадкових вершинах).
12	Задача розфарбовування графу (300 вершин, степінь вершини не більше 30, але не менше 1), бджолиний алгоритм ABC (число бджіл 60 із них 5 розвідники).
13	Задача про рюкзак (місткість P=250, 100 предметів, цінність предметів від 2 до 30 (випадкова), вага від 1 до 25 (випадкова)), генетичний алгоритм (початкова популяція 100 осіб кожна по 1 різному предмету, оператор схрещування одноточковий 30% і 70%, мутація з ймовірністю 5% два випадкові гени міняються місцями). Розробити власний оператор локального покращення.
14	Задача комівояжера (250 вершин, відстань між вершинами випадкова від 1 до 40), мурашиний алгоритм (α = 4, β = 2, ρ = 0,3, Lmin знайти жадібним алгоритмом, кількість мурах M = 45 (10 з них дикі, обирають випадкові напрямки), починають маршрут в різних випадкових вершинах).
15	Задача розфарбовування графу (100 вершин, степінь вершини не більше 20, але не менше 1), класичний бджолиний алгоритм (число бджіл 30 із них 3 розвідники).
16	Задача про рюкзак (місткість P=250, 100 предметів, цінність предметів від 2 до 30 (випадкова), вага від 1 до 25 (випадкова)), генетичний алгоритм (початкова популяція 100 осіб кожна по 1 різному предмету, оператор схрещування двоточковий 30%, 40% і 30%, мутація з ймовірністю 10% два випадкові гени міняються місцями). Розробити власний оператор локального покращення.
17	Задача комівояжера (200 вершин, відстань між вершинами випадкова від 1 до 40), мурашиний алгоритм (α = 2, β = 4, ρ = 0,7, Lmin знайти жадібним алгоритмом, кількість мурах M = 45 (15 з них дикі, обирають випадкові напрямки), починають маршрут в різних випадкових

	вершинах).
18	Задача розфарбовування графу (300 вершин, степінь вершини не більше 50, але не менше 1), класичний бджолиний алгоритм (число бджіл 60 із них 5 розвідники).
19	Задача про рюкзак (місткість P=250, 100 предметів, цінність предметів від 2 до 30 (випадкова), вага від 1 до 25 (випадкова)), генетичний алгоритм (початкова популяція 100 осіб кожна по 1 різному предмету, оператор схрещування триточковий 25%, мутація з ймовірністю 5% два випадкові гени міняються місцями). Розробити власний оператор локального покращення.
20	Задача комівояжера (200 вершин, відстань між вершинами випадкова від 1 до 40), мурашиний алгоритм ($\alpha = 3$, $\beta = 2$, $\rho = 0,7$, Lmin знайти жадібним алгоритмом, кількість мурах M = 45 (10 з них елітні, подвійний феромон), починають маршрут в різних випадкових вершинах).
21	Задача розфарбовування графу (200 вершин, степінь вершини не більше 30, але не менше 1), класичний бджолиний алгоритм (число бджіл 40 із них 2 розвідники).
22	Задача про рюкзак (місткість P=250, 100 предметів, цінність предметів від 2 до 30 (випадкова), вага від 1 до 25 (випадкова)), генетичний алгоритм (початкова популяція 100 осіб кожна по 1 різному предмету, оператор схрещування триточковий 25%, мутація з ймовірністю 5% змінюємо тільки 1 випадковий ген). Розробити власний оператор локального покращення.
23	Задача комівояжера (300 вершин, відстань між вершинами випадкова від 1 до 60), мурашиний алгоритм ($\alpha = 3$, $\beta = 2$, $\rho = 0,6$, Lmin знайти жадібним алгоритмом, кількість мурах M = 45 (15 з них елітні, подвійний феромон), починають маршрут в різних випадкових вершинах).

24	Задача розфарбовування графу (400 вершин, степінь вершини не більше 50, але не менше 1), класичний бджолиний алгоритм (число бджіл 70 із них 10 розвідники).
25	Задача про рюкзак (місткість P=250, 100 предметів, цінність предметів від 2 до 20 (випадкова), вага від 1 до 10 (випадкова)), генетичний алгоритм (початкова популяція 100 осіб кожна по 1 різному предмету, оператор схрещування одноточковий по 50 генів, мутація з ймовірністю 5% змінюємо тільки 1 випадковий ген). Розробити власний оператор локального покращення.
26	Задача комівояжера (100 вершин, відстань між вершинами випадкова від 5 до 50), мурашиний алгоритм ($\alpha = 2$, $\beta = 4$, $\rho = 0,4$, Lmin знайти жадібним алгоритмом, кількість мурах M = 30, починають маршрут в різних випадкових вершинах).
27	Задача розфарбовування графу (200 вершин, степінь вершини не більше 20, але не менше 1), бджолиний алгоритм ABC (число бджіл 30 із них 2 розвідники).
28	Задача про рюкзак (місткість P=200, 100 предметів, цінність предметів від 2 до 20 (випадкова), вага від 1 до 10 (випадкова)), генетичний алгоритм (початкова популяція 100 осіб кожна по 1 різному предмету, оператор схрещування двоточковий порівну генів, мутація з ймовірністю 10% змінюємо тільки 1 випадковий ген). Розробити власний оператор локального покращення.
29	Задача комівояжера (150 вершин, відстань між вершинами випадкова від 5 до 50), мурашиний алгоритм ($\alpha = 2$, $\beta = 3$, $\rho = 0,4$, Lmin знайти жадібним алгоритмом, кількість мурах M = 35, починають маршрут в різних випадкових вершинах).
30	Задача розфарбовування графу (250 вершин, степінь вершини не більше 25, але не менше 2), бджолиний алгоритм ABC (число бджіл 35 із них 3 розвідники).

31	Задача про рюкзак (місткість P=250, 100 предметів, цінність предметів від 2 до 20 (випадкова), вага від 1 до 10 (випадкова)), генетичний алгоритм (початкова популяція 100 осіб кожна по 1 різному предмету, оператор схрещування одноточковий по 50 генів, мутація з ймовірністю 5% змінюємо тільки 1 випадковий ген). Розробити власний оператор локального покращення.
32	Задача комівояжера (100 вершин, відстань між вершинами випадкова від 5 до 50), мурашиний алгоритм ($\alpha = 2$, $\beta = 4$, $\rho = 0,4$, Lmin знайти жадібним алгоритмом, кількість мурах M = 30, починають маршрут в різних випадкових вершинах).
33	Задача розфарбовування графу (200 вершин, степінь вершини не більше 20, але не менше 1), бджолиний алгоритм ABC (число бджіл 30 із них 2 розвідники).
34	Задача про рюкзак (місткість P=200, 100 предметів, цінність предметів від 2 до 20 (випадкова), вага від 1 до 10 (випадкова)), генетичний алгоритм (початкова популяція 100 осіб кожна по 1 різному предмету, оператор схрещування двоточковий порівну генів, мутація з ймовірністю 10% змінюємо тільки 1 випадковий ген). Розробити власний оператор локального покращення.
35	Задача комівояжера (150 вершин, відстань між вершинами випадкова від 5 до 50), мурашиний алгоритм ($\alpha = 2$, $\beta = 3$, $\rho = 0,4$, Lmin знайти жадібним алгоритмом, кількість мурах M = 35, починають маршрут в різних випадкових вершинах).

3 ВИКОНАННЯ

3.1 Програмна реалізація алгоритму

3.1.1 Вихідний код

Повна версія програми розміщена на GitHub

Код алгоритму:

```
public class Genetic
    {
        readonly int[] cost;
        readonly int[] weight;
        readonly int limit;
        readonly double mutationChance;
        readonly int point;
        readonly int iterations;
        List<Member> members = new List<Member>();
        public Genetic(int[] cost, int[] weight, Member[] members, int limit, double
mutationChance, int point, int iterations)
        {
            this.cost = cost;
            this.weight = weight;
            foreach (Member member in members)
            {
                WorkWithList.Add(this.members, member, CalculateCost);
            }
            this.limit = limit;
            this.mutationChance = mutationChance;
            this.point = point;
            this.iterations = iterations;
        }
        public (List<Member>, int, int) Run()
        {
            for (int i = 0; i < iterations; i++)
            {
                var (firstParent, secondParent) = ChoseParents();
                var newChild = MergeParents(firstParent, secondParent);
                Mutation(newChild);
                LocalOptimization(newChild);
                if (CalculateWeight(newChild) <= limit && CalculateCost(newChild) >=
CalculateCost(members[0]))
                {
                    ChangeMin(newChild);
                }
            }
            return (members, CalculateCost(members.Last()),
CalculateWeight(members.Last()));
        }
        (Member, Member) ChoseParents()
        {
            Random random = new Random();
            var firstParent = members.Last();
            var secondParent = FindSecondParent();
            return (firstParent, secondParent);
        }
        Member FindSecondParent()
        {
            Random rand = new Random();
            var randomValue = rand.Next(members.Sum(m => CalculateCost(m)) + 1);
```

```
            int sum = 0;
            for (int i = members.Count - 1; i >= 0; i--)
            {
                sum += CalculateCost(members[i]);
                if (sum >= randomValue)
                {
                    return members[i];
                }
            }
            throw new Exception();
        }
        Member MergeParents(Member first, Member second)
        {
            Random rand = new Random();

                Member firstChild = new Member((bool[])first.Things.Clone());
                for (int i = point; i < second.Things.Length; i++)
                {
                firstChild[i] = second[i];
                }

                Member secondChild = new Member((bool[])second.Things.Clone());
                for (int i = point; i < first.Things.Length; i++)
                {
                    secondChild[i] = first[i];
                }
                return CalculateCost(firstChild) > CalculateCost(secondChild) ?
firstChild : secondChild;
        }
        void Mutation(Member member)
        {
            Random random = new Random();
            if (random.NextDouble() < mutationChance)
            {
                int randomValue = random.Next(member.Things.Length);
                member[randomValue] = !member[randomValue];
                if (CalculateWeight(member) > limit)
                {
                    member[randomValue] = !member[randomValue];
                }
            }
        }
        void LocalOptimization(Member member)
        {
            if (CalculateWeight(member) < limit)
            {
                int min = int.MaxValue;
                int minindex = int.MaxValue;
                for (int i = 0; i < member.Things.Length; i++)
                {
                    if (min > weight[i] && !member[i])
                    {
                        min = weight[i];
                        minindex = i;
                    }
                }
                if (0 <= minindex && minindex < member.Things.Length)
                {
                    member[minindex] = true;
                }
            }

            if(CalculateWeight(member) > limit)
            {
                int min = int.MaxValue;
```

```
            int minindex = int.MaxValue;
            for (int i = 0; i < member.Things.Length; i++)
            {
                if (min > weight[i] && member[i])
                {
                    min = weight[i];
                    minindex = i;
                }
            }
            if (0 <= minindex && minindex < member.Things.Length)
            {
                member[minindex] = false;
            }
        }
    }
    void ChangeMin(Member member)
    {
        members.RemoveAt(0);
        WorkWithList.Add(members, member, CalculateCost);
    }
    int CalculateCost(Member member)
    {
        int result = 0;
        for (int i = 0; i < member.Things.Length; i++)
        {
            if (member[i])
            {
                result += cost[i];
            }
        }
        return result;
    }
    int CalculateWeight(Member member)
    {
        int result = 0;
        for (int i = 0; i < member.Things.Length; i++)
        {
            if (member[i])
            {
                result += weight[i];
            }
        }
        return result;
    }
      }
```

3.1.2 Приклади роботи

На рисунку 3.1 показані співвідношення ціни та вартості предметів. На рисунках 3.2 і 3.3 показані приклади роботи програми.

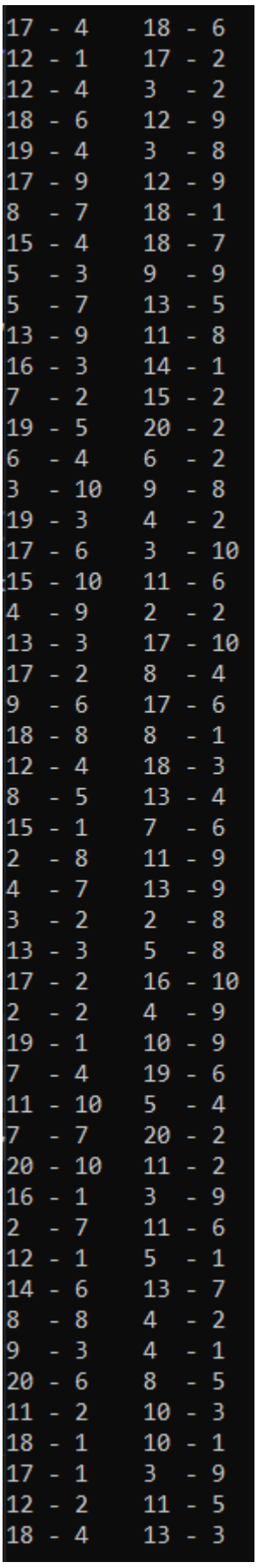

Рисунок 3.1 – Співвідношення ціни та вартості предметів

Рисунок 3.2 – Запуск програми на 50000 ітерацій

Рисунок 3.3 – Запуск програми на 100000 ітерацій

3.1.3 Значення цільової функції зі збільшенням кількості ітерацій

У таблиці 3.1 наведено значення цільової функції зі збільшенням кількості ітерацій.

Кількість ітерацій	Ціна зібраного рюкзаку	Вага зібраного рюкзаку
20	130	14
40	223	26
60	270	38
80	342	52
100	419	71
120	463	80
140	482	86
160	577	110
180	603	122
200	636	132
220	669	150
240	716	167
260	753	185
280	796	203
300	833	223
320	849	244
340	849	244
360	849	244
380	849	244
400	849	244
420	849	244
440	849	244

Таблиця 3.1 – Результати виконання алгоритму в залежності від кількості ітерацій

Продовження таблиці 3.1

Кількість ітерацій	Ціна зібраного рюкзаку	Вага зібраного рюкзаку
460	849	244
480	849	244
500	849	244
520	849	244
540	849	244
560	849	244
580	849	244
600	849	244
620	849	244
640	849	244
660	849	244
680	849	244
700	849	244
720	849	244
740	849	244
760	849	244
780	849	244
800	849	244
820	849	244
840	849	244
860	849	244
880	849	244
900	849	244
920	849	244
940	849	244
960	849	244
980	849	244

Продовження таблиці 3.1

Кількість ітерацій	Ціна зібраного рюкзаку	Вага зібраного рюкзаку
1000	849	244

З таблиці можемо помітити, що піся 320 ітерацій, було досягнуто одного з локальних максимумів – 849. Загалом значення росли досить швидко до 320 ітерації. Це означає, що алгоритм досить швидко знаходить близке до оптимального рішення.

3.1.4 Графіки залежності розв'язку від числа ітерацій

На рисунку 3.3 наведений графік, який показує якість отриманого розв'язку.

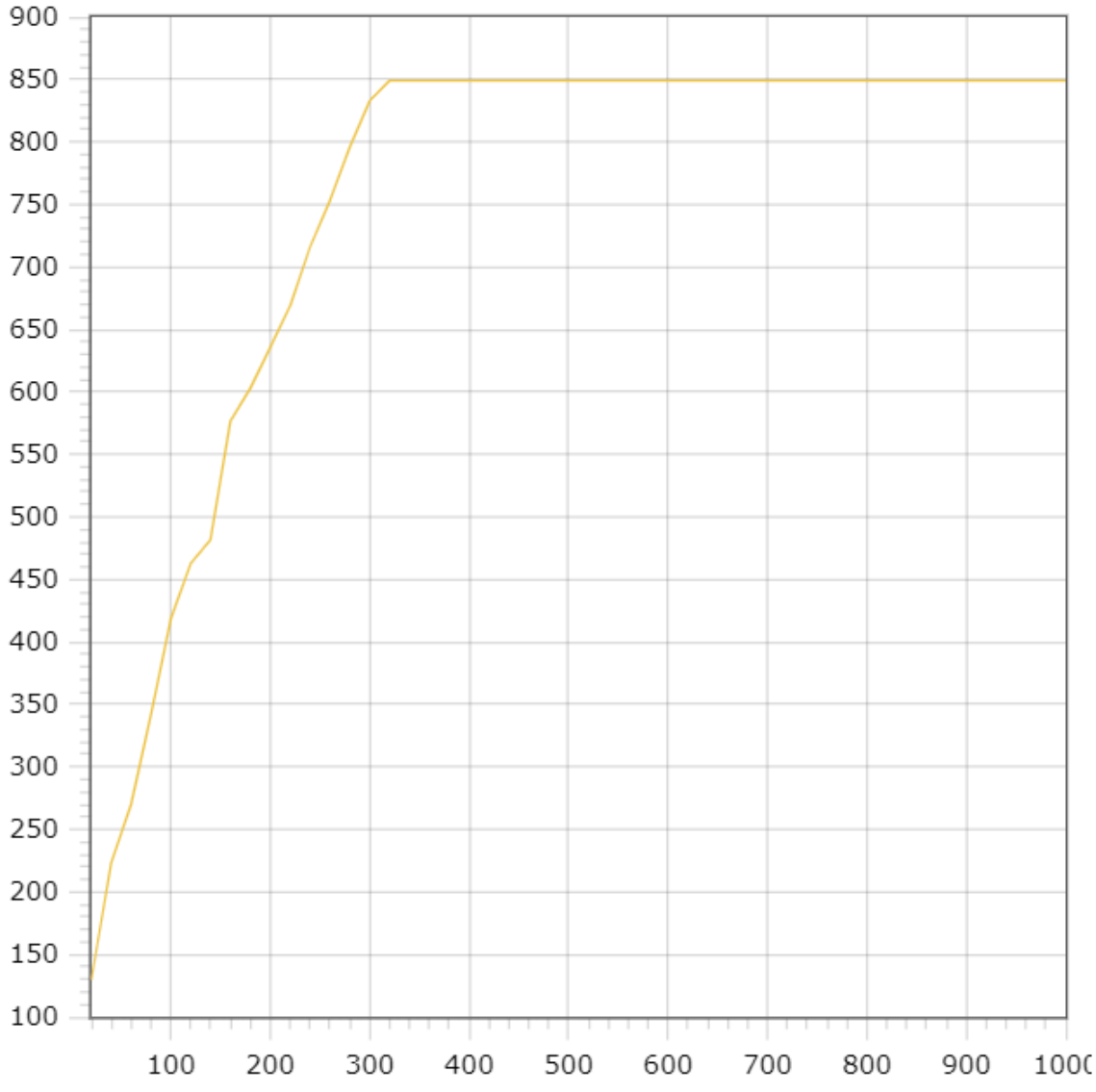

Рисунок 3.3 – Графіки залежності розв'язку від числа ітерацій

ВИСНОВОК

В рамках даної лабораторної роботи реалізували генетичний алгоритм для задачі розміщення рюкзака, створили локальну оптимізацію, яка покращила результати, приблизно, в півтора рази. Досягли наступних результатів: алгоритм за 320 ітерацій досягає локального максимуму, який він зміг подолати тільки на 100000-й ітерації, тобто він швидко знаходить близьке до оптимального рішення. Данний алгоритм потребує сталу, невелику кількість додаткової пам’яті і досить швидко знаходить близьке до оптимального рішення, тому варто про нього пам’ятати в задачах оптимізації.

КРИТЕРІЇ ОЦІНЮВАННЯ

При здачі лабораторної роботи до 27.11.2021 включно максимальний бал дорівнює – 5. Після 27.11.2021 максимальний бал дорівнює – 1.

Критерії оцінювання у відсотках від максимального балу:

- програмна реалізація алгоритму – 75%;
- тестування алгоритму– 20%;
- висновок – 5%.