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 № 1 з дисципліни
«Проектування алгоритмів»

«Неінформативний, інформативний та локальний пошук»

Виконав(ла) *ІТ-03 Чабан Антон Євгенович* _____
(шифр, прізвище, ім'я, по батькові)

Перевірив *Головченко М.М.* _____
(прізвище, ім'я, по батькові)

Київ 2021

1 МЕТА ЛАБОРАТОРНОЇ РОБОТИ

Мета роботи – розглянути та дослідити алгоритми неінформативного, інформативного та локального пошуку. Провести порівняльний аналіз ефективності використання алгоритмів.

2 ЗАВДАННЯ

Записати алгоритм розв’язання задачі у вигляді псевдокоду, відповідно до варіанту (таблиця 2.1).

Реалізувати програму, яка розв’язує поставлену задачу згідно варіанту (таблиця 2.1) за допомогою алгоритму неінформативного пошуку **АНП**, алгоритму інформативного пошуку **АІП,** що використовує задану евристичну функцію Func, або алгоритму локального пошуку **АЛП та бектрекінгу,** що використовує задану евристичну функцію Func.

Програму реалізувати на довільній мові програмування.

Увага! Алгоритм неінформативного пошуку **АНП,** реалізовується за принципом «AS IS», тобто так, як є, без додаткових модифікацій (таких як перевірка циклів, наприклад).

Провести серію експериментів для вивчення ефективності роботи алгоритмів. Кожний експеримент повинен відрізнятись початковим станом. Серія повинна містити не менше 20 експериментів для кожного алгоритму. За проведеними серіями необхідно визначити:

– середню кількість етапів (кроків), які знадобилось для досягнення розв’язку (ітерації);

– середню кількість випадків, коли алгоритм потрапляв в глухий кут (не міг знайти оптимальний розв’язок) – якщо таке можливе;

– середню кількість згенерованих станів під час пошуку;

– середню кількість станів, що зберігаються в пам’яті під час роботи програми.

Передбачити можливість обмеження виконання програми за часом (30 хвилин) та використання пам’яті (512 Мб)

Використані позначення:

– **8-puzzle** – гра, що складається з 8 однакових квадратних пластинок з нанесеними числами від 1 до 8. Пластинки поміщаються в квадратну коробку, довжина сторони якої в три рази більша довжини сторони пластинок,

відповідно в коробці залишається незаповненим одне квадратне поле. Мета гри – переміщаючи пластинки по коробці досягти впорядковування їх по номерах, бажано зробивши якомога менше переміщень.

- **IDS** – Пошук вглиб з ітеративним заглибленням.
- **RBFS** – Рекурсивний пошук за першим найкращим співпадінням.
- **H2** – Манхетенська відстань.

Таблиця 2.1 – Варіанти алгоритмів

№	**Задача**	**АНП**	**АІП**	**АЛП**	**Func**
24	8-puzzle	IDS	RBFS		H2

3 ВИКОНАННЯ

3.1 Псевдокод алгоритмів

IDS:

```
function IDS(root) is
    for depth from 0 to ∞ do
        found, remaining ← DLS(root, depth)
        if found ≠ null then
            return found
        else if not remaining then
            return null

function DLS(node, depth) is
    if depth = 0 then
        if node is a goal then
            return (node, true)
        else
            return (null, true)     (Not found, but may have children)

    else if depth > 0 then
        any_remaining ← false
        foreach child of node do
            found, remaining ← DLS(child, depth−1)
            if found ≠ null then
                return (found, true)
            if remaining then
                any_remaining ← true    (At least one node found at depth, let
IDDFS deepen)
        return (null, any_remaining)
```

RBFS:

```
function Recursive-Best-First-Search(problem) returns решение result
  или индикатор неудачи failure
    RBFS(problem, Make-Node(Initial-State[problem] ) , ∞)

function RBFS(problem, node, f_limit) returns решение result
  или индикатор неудачи failure и новый предел f-стоимости f_limit
    if Goal-Test[problem](State[node]) then return узел node
    successors ← Expand(node, problem)
    if множество узлов-преемников successors пустое
      then return failure, ∞
    for each s in successors do
       f[s] ← max(g(s)+h(s) , f[node] )
repeat
      best ← узел с наименьшим f-значением во множестве successors
      if f[best] > f_limit then return failure, f[best]
      alternative ← второе после наименьшего f-значение во множестве successors
      result, f[best] ← RBFS(problem, best, min{f_limit, alternative))
      if result ≠ failure then return result
```

3.2 Програмна реалізація

3.2.1 Вихідний код

[Код знаходиться на GitHub]

3.2.2 Приклади роботи

На рисунках 3.1 і 3.2 показані приклади роботи програми для різних алгоритмів пошуку.

Рисунок 3.1 – Алгоритм IDS

Рисунок 3.2 – Алгоритм RBFS

3.3 Дослідження алгоритмів

Таблиця 3.1 – Характеристики оцінювання алгоритму IDS

Початкові стани	Ітерації	Всього станів	Глибина
715234086	121961214	121961227	18
013425786	29	31	4
810543276	43606200	43606219	18
580342761	+-3.5млрд	+-3.5млрд	24
285417036	INT.Max_VAL*3+-	INT.Max_VAL*3+-	FAILED
150362478	164815	164825	12
016382475	3715207	3715222	16
421835067	260417134	260417150	20
813264570	-	-	TIMEOUT(20+)
517302486	5289291	5289306	16
520136478	33743	33751	12

123406758	16	16	2
172803465	371232	371240	12
026385147	138146273	138146284	18
187452630	100485014	100485029	20
136285470	164904	164913	12
156703824	15496682	15496694	16
413805762	671046	671058	14
123854076	725052	725064	14
026371548	351808721	351808740	20

Таблиця 3.3 – Характеристики оцінювання алгоритму RBFS

Початкові стани	Ітерації	Всього станів	Глибина
715234086	-	264	18
013425786	-	34	4
810543276	-	693	18
580342761	-	3141	24
285417036	-	545	20
150362478	-	41	12
016382475	-	274	16
421835067	-	748	20
813264570	-	1126	20
517302486	-	137	16
520136478	-	80	12
123406758	-	6	2
172803465	-	31	12
026385147	-	273	18
187452630	-	205	20
136285470	-	58	12
156703824	-	599	16
413805762	-	140	14
123854076	-	263	14
026371548	-	1650	20

ВИСНОВОК

При виконанні даної лабораторної роботи було розглянуто алгоритми пошуку. Один з яких є неінформативним – IDS (пошук в глибину з ітеративним заглибленням), а другий – інформативний RBFS (рекурсивний пошук за першим найкращим значенням). Також було проведено дослідження алгоритмів на 20-ти початкових станах.

КРИТЕРІЇ ОЦІНЮВАННЯ

За умови здачі лабораторної роботи до 25.09.2021 включно максимальний бал дорівнює – 5. Після 25.09.2021 максимальний бал дорівнює – 1.

Критерії оцінювання у відсотках від максимального балу:

- псевдокод алгоритму – 10%;
- програмна реалізація алгоритму – 60%;
- дослідження алгоритмів – 25%;
- висновок – 5%.