**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)**

(шифр, прізвище, ім'я, по батькові)

*ІП-01Черпак Андрій Вадимович*

**Перевірив**

(прізвище, ім'я, по батькові)

*Головченко М.М.*

Київ 2021

Зміст

[1 Мета лабораторної роботи 3](#_Toc81070688)

[2 ЗаВдання 4](#_Toc81070689)

[3 Виконання 8](#_Toc81070690)

[3.1 Псевдокод алгоритмів 8](#_Toc81070691)

[3.2 Програмна реалізація 8](#_Toc81070692)

[3.2.1 Вихідний код 8](#_Toc81070693)

[3.2.2 Приклади роботи 8](#_Toc81070694)

[3.3 Дослідження алгоритмів 8](#_Toc81070695)

[Висновок 11](#_Toc81070696)

[Критерії оцінювання 12](#_Toc81070697)

# Мета лабораторної роботи

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

# ЗаВдання

Записати алгоритм розв’язання задачі у вигляді псевдокоду, відповідно до варіанту .

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

Програму реалізувати на довільній мові програмування.

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:**

* **COLOR** – Задача розфарбування карти самостійно обраної країни, не менше 20 регіонів (областей). Необхідно розфарбувати карту не більше ніж у 4 різні кольори. Мається на увазі приписування кожному регіону власного кольору так, щоб кольори сусідніх регіонів відрізнялись. Використовувати евристичну функцію, яка повертає кількість пар суміжних вузлів, що мають однаковий колір (тобто кількість конфліктів). Реалізувати алгоритм пошуку із поверненнями (backtracking) для розв’язання поставленої задачі. Для підвищення швидкодії роботи алгоритму використати евристичну функцію, а початковим станом вважати випадкову вершину.
* **ANNEAL** – Локальний пошук із симуляцією відпалу. Робоча характеристика – залежність температури Т від часу роботи алгоритму t. Можна розглядати лінійну залежність: T = 1000 - k∙t, де k – змінний коефіцієнт.
* **DGR** – ступенева евристика.

Таблиця 2.1 – Варіанти алгоритмів

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№** | **Задача** | **АНП** | **АІП** | **АЛП** | **Func** |
| 29 | COLOR | - | - | ANNEAL | DGR |

# Виконання

## Псевдокод алгоритмів

### ANNEAL

#### ChooseColors(provinces)

T = 1000

k = 2

currentState = Copy(provinces);

**для кожного** prov **у** provinces **повторити**

prov.Color = 1

**все повторити**

**поки** T > 0 **повторити**

newState = getNewState(currentState)

newStateConflicts = conflictCount(newState)

currentStateConflict = conflictCount(currentState)

stateDifference = currentStateConflict - newStateConflicts

**якщо** newStateConflicts == 0 **то**

provinces = newState

**повернути** true

**все якщо**

**якщо** stateDifference > 0 || randomDouble>e^(stateDifference/T) **то**

currentState = newState

T -= k

**все якщо**

**все поки**

**повернути** false

#### conflictCount(state)

ctr = 0

**для кожного** prov **у** state **повторити**

ctr += prov.Conflicts(state)

**все повторити**

**повернути** ctr

#### getNewState(currentState)

newState = Copy(currentState)

withConflict = []

**для кожного** province **в** newState **повторити**

**якщо** province.Conflicts(newState) > 0 **то**

withConflict.Add(province)

**все якщо**

**все повторити**

randomProvince = випадкова провінція з withConflict

randomProvince.Color = випадковий колір з доступних

**повернути** newState

### DGR

#### findMinDGR(provinces)

minDegree = Int32.MaxValue  
i = -1  
**для кожного** p **в** provinces **виконати**  
 **якщо** колір області вже визначено

**то**

**продовжити**

**все якщо**  
 **якщо** для даної області не залишилося можливих кольорів,

**то**

**повернути** -1

**все якщо**  
 degree = 0

**для кожної** суміжної області **виконати**

**якщо** колір не визначено

**то**

degree = degree+1  
 **все якщо**

**все виконати**

**якщо** degree < minDegree

**то**

minDegree = degree  
 i = {індекс даної області у списку provinces}  
 **все якщо**

**все виконати**

**повернути** і

#### **ChooseColors(**provinces, count = 1 **)**

i = findMinDGR(provinces)  
**якщо** i == -1 **то**

**повернути** false

**все якщо**  
**для кожного** можливого кольору col і-тої області **виконати**  
 IterationCounter = IterationCounter+1  
 n\_prov = копія provinces  
 встановити col кольором і-тої області  
 StateCounter = StateCounter+1  
 **якщо** кількість зафарбованих областей count= кількості областей **то**

provinces = n\_prov  
 **повернути** true  
 **все якщо**  
 **якщо** ChooseColors(n\_prov, count+1) виконалась успішно **то**  
 provinces = n\_prov  
 **повернути** true  
 **все якщо**  
 **все виконати**  
**повернути** false

## Програмна реалізація

### Приклади роботи

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

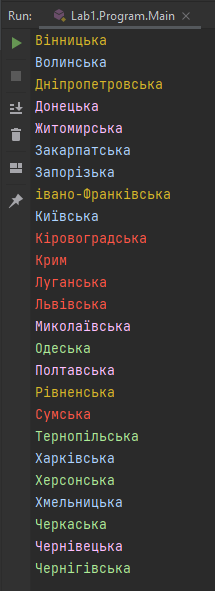


Рисунок 3.1 – Алгоритм ANNEAL

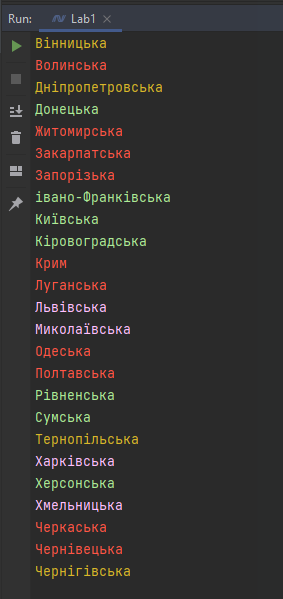


Рисунок 3.2 – Алгоритм Backtracking з DGR евристикою

## Дослідження алгоритмів

В таблиці 3.1 наведені характеристики оцінювання алгоритму ANNEAL задачі COLOR для 20 початкових станів.

Таблиця 3.1 – Характеристики оцінювання алгоритму

|  |  |  |  |
| --- | --- | --- | --- |
| Початкові стани | Ітерації | Кількість випадкових переходів | Всього станів згенеровано |
| Стан 1 | 45299 | 11 | 45300 |
| Стан 2 | 29639 | 6 | 29640 |
| Стан 3 | 21276 | 4 | 21277 |
| Стан 4 | 36259 | 3 | 36260 |
| Стан 5 | 8548 | 1 | 8549 |
| Стан 6 | 7488 | 1 | 7489 |
| Стан 7 | 41871 | 13 | 41872 |
| Стан 8 | 49621 | 7 | 49622 |
| Стан 9 | 49300 | 12 | 49301 |
| Стан 10 | 3834 | 0 | 3835 |
| Стан 11 | 73450 | 26 | 73451 |
| Стан 12 | 45221 | 5 | 45222 |
| Стан 13 | 15061 | 3 | 15062 |
| Стан 14 | 8395 | 1 | 8396 |
| Стан 15 | 20680 | 4 | 20681 |
| Стан 16 | 52040 | 14 | 52041 |
| Стан 17 | 12226 | 3 | 12227 |
| Стан 18 | 25825 | 6 | 25826 |
| Стан 19 | 14747 | 2 | 14748 |
| Стан 20 | 41964 | 8 | 41965 |
| Середнє: | 30137,2 | 6,5 | 30138,2 |

В таблиці 3.2 наведені характеристики оцінювання алгоритму Backtracking з DGR задачі COLOR для 20 початкових станів.

Таблиця 3.2 – Характеристики оцінювання алгоритму Backtracking з DGR евристикою

|  |  |  |  |
| --- | --- | --- | --- |
| Початкові стани | Ітерації | Кількість глухих кутів | Всього станів |
| Стан 1 | 158100 | 158076 | 158101 |
| Стан 2 | 33 | 9 | 34 |
| Стан 3 | 34 | 10 | 35 |
| Стан 4 | 31 | 7 | 32 |
| Стан 5 | 29 | 5 | 30 |
| Стан 6 | 31 | 7 | 32 |
| Стан 7 | 24 | 0 | 25 |
| Стан 8 | 32 | 8 | 33 |
| Стан 9 | 29 | 5 | 30 |
| Стан 10 | 28 | 4 | 29 |
| Стан 11 | 31 | 7 | 32 |
| Стан 12 | 31 | 7 | 32 |
| Стан 13 | 31 | 7 | 32 |
| Стан 14 | 38 | 14 | 39 |
| Стан 15 | 26 | 2 | 27 |
| Стан 16 | 27 | 3 | 28 |
| Стан 17 | 24 | 0 | 25 |
| Стан 18 | 27 | 13 | 38 |
| Стан 19 | 45 | 21 | 46 |
| Стан 20 | 63 | 39 | 64 |
| Середнє: | 7935 | 7912 | 7936 |

Висновок

При виконанні даної лабораторної роботи було розглянуто алгоритми ANNEAL та Backtracking з DGR евристикою і використано їх для вирішення задачі COLOR. Ми дослідили характеристики даних алгоритмів і перевірили для 20 початкових станів. На мові C# програмно реалізували дані алгоритми і перевірили їх ефективність. Обмеження по пам’яті та часу для алгоритму Backtracking з DGR евристикою введено не було, так як кількість станів, що зберігається в пам’яті досить обмежена і дорівнює кількості областей, а час не перевищує двох секунд (якщо розв’язок існує, інакше – близько 2х хвилин).