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

Лабораторна робота №3.1

з дисципліни
«Інтелектуальні вбудовані системи»

на тему «Реалізація розкладання числа на прості множники(факторизація числа)»

Виконав:
студент 3 курсу
ФІОТ, групи ІП-84
Кришталь Дмитро Вікторович
Номер залікової - 8414

Перевірив:
Регіда П. Г.

Київ 2021

Основні теоретичні відомості

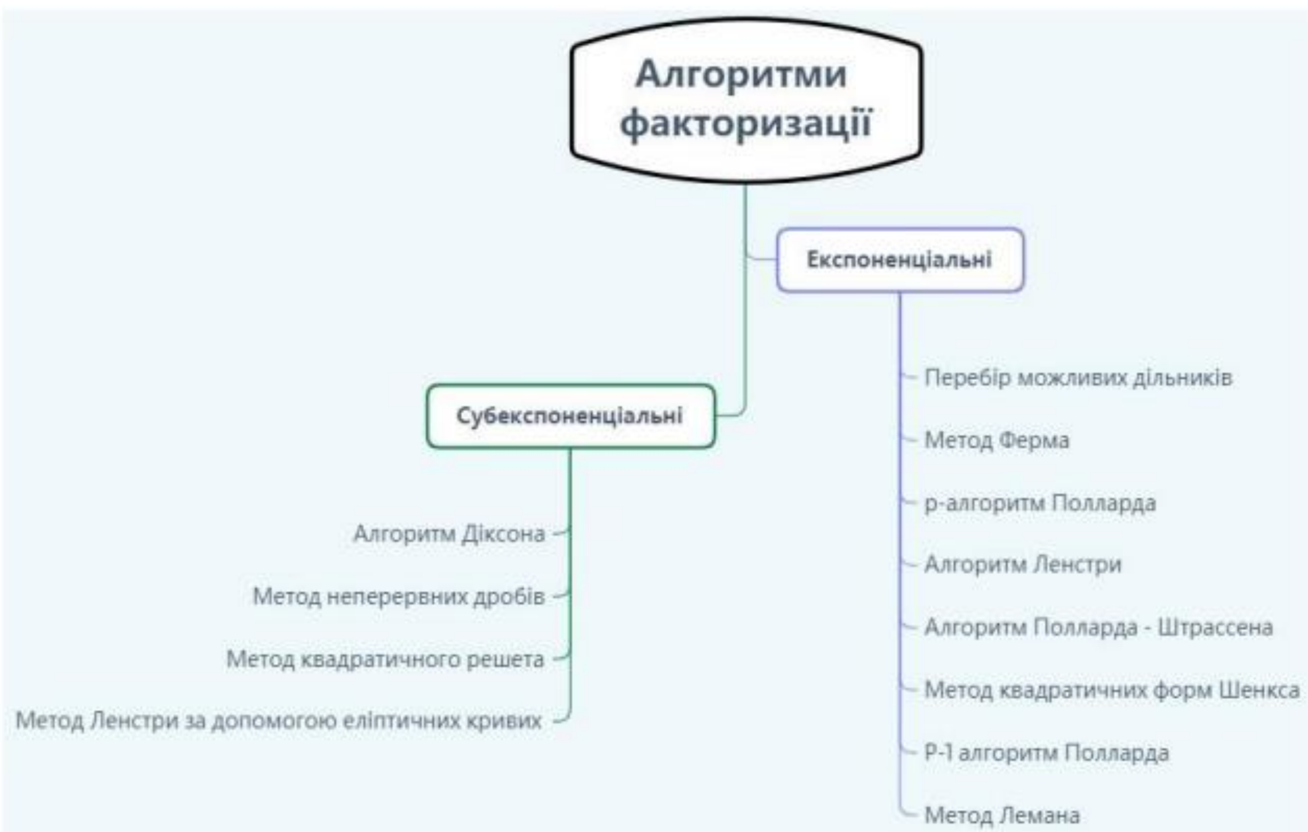
Факторизації лежить в основі стійкості деяких криптоалгоритмів, еліптичних кривих, алгебраїчній теорії чисел та кванових обчислень, саме тому дана задача дуже гостро досліджується, й шукаються шляхи її оптимізації.

На вхід задачі подається число n Є ℕ, яке необхідно факторизувати. Перед виконанням алгоритму слід переконатись в тому, що число не просте. Далі алгоритм шукає перший простий дільник, після чого можна запустити алгоритм заново, для повторної факторизації.

В залежності від складності алгоритми факторизації можна розбити на дві групи:

- Експоненціальні алгоритми (складність залежить експоненційно від довжини вхідного параметру);
- Субекспоненціальні алгоритми.

Існування алгоритму з поліноміальною складністю – одна з найважливіших проблем в сучасній теорії чисел. Проте, факторизація з даною складністю можлива на квантовому комп'ютері за допомогою алгоритма Шора.



Метод перебору можливих дільників.

Один з найпростіших і найочевидніших алгоритмів заключається в тому, щоб послідовно ділити задане число n на натуральні числа від 1 до $\lfloor\sqrt{n}\rfloor$. Формально, достатньо ділити лише на прості числа в цьому інтервалі, але для цього необхідно знати їх множину. На практиці складається таблиця простих чисел і на вхід подаються невеликі числа (до 2^{16}), оскільки даний алгоритм має низьку швидкість роботи.

Приклад алгоритму:

1. Початкова установка: t = 0, k = 0, n = N (t,k,n такі, що n = N / $p_1 \ldots p_n$ і n не мають простих множників, менших за d_k).
2. Якщо n = 1, закінчуємо алгоритм.
3. Присвоюємо q = [n / d_k], r = n mod d_k.
4. Якщо r ≠ 0, переходимо на крок 6.
5. Присвоюємо t++, $p_t = d_k$, n = q і повертаємось на крок 2.
6. Якщо q > d_k → k++ і повертаємось на крок 3.
7. Присвоїти t++, p_t = n і закінчити виконання алгоритму.

Модофікований метод факторизації Ферма.

Ідея алгоритму заключається в пошуку таких чисел A і B, щоб факторизоване число n мало вигляд: $n = A^2 – B^2$. Даний метод гарний тим, що реалізується без використання операцій ділення, а лише з операціями додавання й віднімання.

Приклад алгоритму:

1. Початкова установка: $x = 2[\sqrt{n}] + 1$, y = 1, $r = [\sqrt{n}]^2 – n$.
2. Якщо r = 0, то алгоритм закінчено: $n = \frac{x-y}{2} * \frac{x+y-2}{2}$
3. Присвоюємо r = r + x, x = x + 2.
4. Присвоюємо r = r - y, y = y + 2.
5. Якщо r > 0, повертаємось до кроку 4, інакше повертаємось до кроку 2.

Метод факторизації Ферма.

Ідея алгоритму заключається в пошуку таких чисел A і B, щоб факторизоване число n мало вигляд: $n = A^2 – B^2$. Даний метод гарний тим, що реалізується без використання операцій ділення, а лише з операціями додавання й віднімання.

Приклад алгоритму:

Початкова установка: $x = [\sqrt{n}]$ – найменше число, при якому різниця x^2-n невід'ємна. Для кожного значення k Є ℕ, починаючи з k = 1, обчислюємо $(\lceil\sqrt{n}\rceil + k)^2 - n$ і перевіряємо чи не є це число точним квадратом.

- Якщо не є, то k++ і переходимо на наступну ітерацію.
- Якщо є точним квадратом, тобто $x^2 - n = \left(\lceil\sqrt{n}\rceil + k\right)^2 - n = y^2$, то ми отримуємо розкладання: $n = x^2 - y^2 = (x+y)(x-y) = A * B$, в яких
$$x = (\lceil\sqrt{n}\rceil + k)$$

Якщо воно є тривіальним і єдиним, то n - просте

Умови завдання

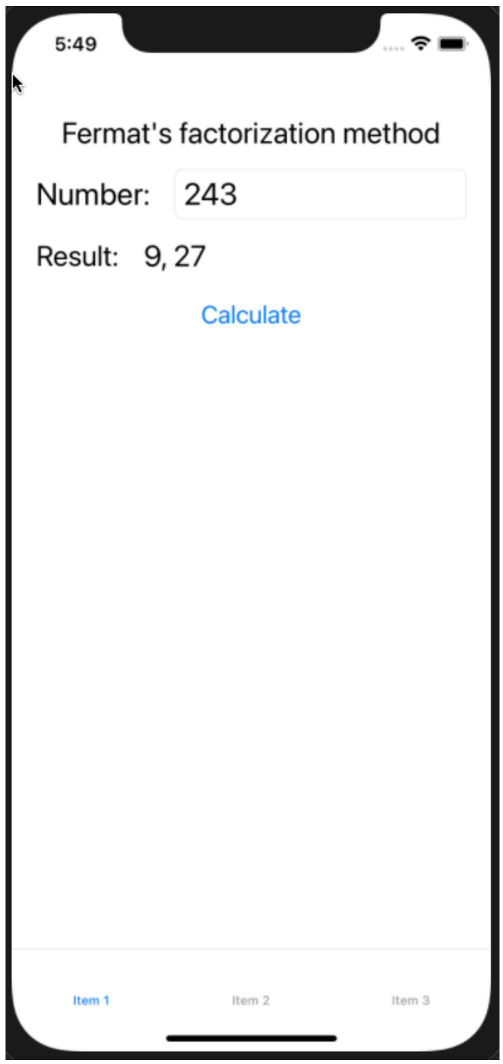
Розробити програму для факторизації заданого числа методом Ферма. Реалізувати користувацький інтерфейс з можливістю вводу даних.

Лістинг програми із заданими умовами завдання

```
import UIKit

class FirstViewController: UIViewController {
  @IBOutlet var text: UITextField!
  @IBOutlet var outputLabel: UILabel!
  @IBOutlet var button: UIButton!

  override func viewDidLoad() {
    super.viewDidLoad()

    text.delegate = self
    text.keyboardType = .numberPad
    outputLabel.text = ""
    button.addTarget(self, action: #selector(didTapCalculateButton), for: .touchUpInside)
  }
  @objc func didTapCalculateButton() {
    text.resignFirstResponder()
    guard let textEntered = text.text, let number = Int(textEntered) else {return}
    if (number % 2 == 0) {
      outputLabel.text = "Enter odd number"
      return
    }

    var num = ceil(sqrt(Double(number)))
    while(sqrt(pow(num, 2) - Double(number)) != floor(sqrt(pow(num, 2) - Double(number)))) {
      num += 1
    }
    outputLabel.text = "\(Int(num - sqrt(pow(num, 2) - Double(number)))), \(Int(num + sqrt(pow(num, 2) - Double(number))))"
```

```
    }
}

extension FirstViewController: UITextFieldDelegate {
    func textFieldShouldReturn(_ textField: UITextField) -> Bool {
        textField.resignFirstResponder()
        return true
    }
```

Результати виконання програми



Висновок

Під час виконання лабораторної роботи №3.1 ознайомився з основними принципами розкладання числа на множники з використання алгоритму ферма. Написав програму. Наведені скріншоти підтверджують правильність її роботи.