

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"

Інститут КНІТ
Кафедра ПЗ

ЗВІТ

До лабораторної роботи № 2
На тему: “Метод сортування вибором”
З дисципліни: “Алгоритми та структури даних”

Лектор:
доцент кафедри ПЗ
Коротєєва Т.О.

Виконав:
студент групи ПЗ-22
Коваленко Д.М.

Прийняв:
асистент кафедри ПЗ
Франко А.В.

«_____» _____ 2022 р.
 Σ = _____

Тема. Метод сортування вибором.

Мета. Вивчити алгоритм сортування вибором. Здійснити програмну реалізацію алгоритму сортування вибором. Дослідити швидкість алгоритму сортування вибором.

Лабораторне завдання

Створити віконний проект та написати програму, яка реалізує алгоритм сортування бульбашкою.

3. Задано одномірний масив дійсних чисел. До парних елементів масиву застосувати функцію $\sqrt{|x - 10|}$.
Отриманий масив посортувати в порядку зростання

Теоретичні відомості

Сортування вибором (англійською «Selection Sort») — простий алгоритм сортування лінійного масиву, на основі вставок. Має ефективність $O(n^2)$, що робить його неефективним при сортування великих масивів, і в цілому, менш ефективним за подібний алгоритм сортування включенням. Сортування вибором вирізняється більшою простотою, ніж сортування включенням, і в деяких випадках вищою продуктивністю.

Алгоритм працює наступним чином:

1. Знаходить у списку найменше значення.
2. Міняє його місцями із першим значенням у списку.
3. Повторює два попередніх кроки, доки список не завершиться (починаючи з другої позиції).

Фактично, таким чином ми поділили список на дві частини: перша (ліва) — повністю відсортована, а друга (права) — ні.

Покроковий опис роботи алгоритму сортування вибором

Алгоритм S

S1 - встановити $MIN = 0$

S2 - знайти найменший елемент в масиві

S3 - замінити найменший елемент на той, що розміщений на місці MIN

S4 - збільшити значення MIN на 1

S5 - повторити виконання S2 - S4

Хід роботи

Файл sort.rs

```
use crate::data::Data;

pub struct Sorted;

impl Sorted {
    pub fn sort(mut v: Vec<Data>) -> Vec<Vec<Data>> {
        let mut res = vec![v.clone()];
        for i in 0..(v.len() - 1) {
            let mut min = i;
            for j in (i + 1)..v.len() {
                if v[j] < v[min] {
                    min = j;
                }
            }
            if min != i {
                v.swap(i, min);
                res.push(v.clone());
            }
        }
        res
    }
}
```

Файл data.rs

```
use fake::{Dummy, Fake};

#[derive(Debug, Clone, PartialEq, Dummy)]
pub struct Data {
    #[dummy()]
    pub v: f32,
}

impl Data {
    pub fn new(len: usize) -> Vec<Self> {
        let mut vec = fake::vec![Data; len];
        vec.iter_mut()
            .step_by(2)
            .for_each(|x| x.v = f32::sqrt(f32::abs(x.v - 10.)));
        vec
    }

    pub fn diff(a: &Vec<Data>, b: &Vec<Data>) -> Vec<usize> {
        let mut res = Vec::new();

        for i in 0..a.len() {
            if a[i] != b[i] {
                res.push(i);
            }
        }

        res
    }
}

impl PartialOrd for Data {
    fn partial_cmp(&self, other: &Self) -> Option<std::cmp::Ordering> {
        self.v.partial_cmp(&other.v)
    }
}
```

Файл lib.rs

```
extern crate console_error_panic_hook;

mod data;
mod sort;
mod utils;

use wasmbindgen::{prelude::*, JsCast};
use web_sys::{
    HTMLElement, HtmlInputElement,
    HtmlTableCellElement, HtmlTableElement,
    HtmlTableRowElement,
    Text,
};

use data::Data;
use utils::document;
use sort::Sorted;

use std::{cell::RefCell, rc::Rc};
```

```

use std::panic;

#[wasm_bindgen]
extern "C" {
    #[wasm_bindgen(js_namespace = console)]
    fn log(s: &str);
}

#[wasm_bindgen(start)]
pub fn run() {
    panic::set_hook(Box::new(console_error_panic_hook::hook));

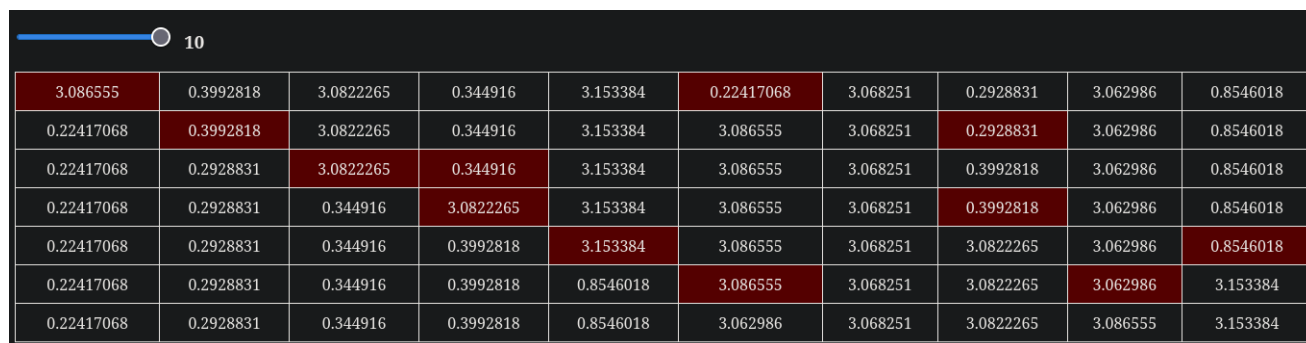
    let slider = Rc::new(RefCell::new(
        document()
        .get_element_by_id("slider")
        .unwrap()
        .dyn_into::<HtmlInputElement>()
        .unwrap(),
    ));
    let slider_onchange = slider.clone();
    let f = Closure::wrap(Box::new(move || {
        let len = slider_onchange.borrow().value_as_number();
        document()
        .get_element_by_id("slider_value")
        .unwrap()
        .dyn_into::<HtmlElement>()
        .unwrap()
        .set_inner_html(&len.to_string());
        let data = Sorted::sort(Data::new(len as usize));
        let table = document()
        .get_element_by_id("table")
        .unwrap()
        .dyn_into::<HtmlTableElement>()
        .unwrap();
        table.set_inner_html("");
        for i in 0..data.len() {
            let diff = if let Some(b) = data.get(i + 1) {
                Data::diff(&data[i], &b)
            } else {
                vec![100, 100]
            };
            let row = table
            .insert_row()
            .unwrap()
            .dyn_into::<HtmlTableRowElement>()
            .unwrap();
            for j in 0..data[i].len() {
                let cell = row
                .insert_cell()
                .unwrap()
                .dyn_into::<HtmlTableCellElement>()
                .unwrap();
                let text = document()
                .create_text_node(&data[i][j].v.to_string())
                .dyn_into::<Text>()
                .unwrap();
                cell.append_child(&text).unwrap();
                if diff.contains(&j) {
                    cell.set_bg_color("rgb(255,200,200)");
                }
            }
        }
    })
    .into());
}

```

```

    }
  }) as Box<dyn FnMut()>;
  slider
    .borrow()
    .set_oninput(Some(f.as_ref().unchecked_ref()));
  f.forget();
}

```



3.086555	0.3992818	3.0822265	0.344916	3.153384	0.22417068	3.068251	0.2928831	3.062986	0.8546018
0.22417068	0.3992818	3.0822265	0.344916	3.153384	3.086555	3.068251	0.2928831	3.062986	0.8546018
0.22417068	0.2928831	3.0822265	0.344916	3.153384	3.086555	3.068251	0.3992818	3.062986	0.8546018
0.22417068	0.2928831	0.344916	3.0822265	3.153384	3.086555	3.068251	0.3992818	3.062986	0.8546018
0.22417068	0.2928831	0.344916	0.3992818	3.153384	3.086555	3.068251	3.0822265	3.062986	0.8546018
0.22417068	0.2928831	0.344916	0.3992818	0.8546018	3.086555	3.068251	3.0822265	3.062986	3.153384
0.22417068	0.2928831	0.344916	0.3992818	0.8546018	3.062986	3.068251	3.0822265	3.086555	3.153384

Рис. 1: Виконання програми

Висновок

Під час виконання лабораторної роботи я вивчив алгоритм сортування вибором. Здійснив програмну реалізацію алгоритму сортування вибором. Дослідив швидкодію алгоритму сортування вибором.