**КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ**

**ІМЕНІ ТАРАСА ШЕВЧЕНКА**

Проект з дисципліни

**«Розробка програмного забезпечення під мобільні платформи»**

на тему:

**«Сортування масивів. Демонстрація і порівняння різних методів»**

**Виконав**:

студент 4 курсу ОС «Бакалавр»

спеціальності «Компʼютерні науки»,

освітньої програми «Інформатика»,

групи ТТП-42

*Сирота Сергій Вячеславович*

**Київ-2022**

ЗМІСТ

[ОПИС ЗАВДАННЯ](#_gjdgxs) 3

[КОРОТКИЙ ВИКЛАД ТЕОРІЇ](#_30j0zll) 4

[ОПИС АЛГОРИТМІВ](#_1fob9te) 5

[ОПИС РОБОТИ З ПРОГРАМОЮ ТА ЗНІМКИ ЕКРАНУ](#_3znysh7) 7

[СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ](#_2et92p0) 11

[ДОДАТКИ](#_tyjcwt) 12

[Додаток А](#_3dy6vkm) Фрагмент Коду Навігації 12

[Додаток Б](#_1t3h5sf) Фрагмент Коду Файлу, Що Відповідає За Ввід Масиву з файлу 14

[Додаток В](#_4d34og8) Фрагмент Коду Файлу, Що Відповідає За Відображення гістограми 15

[Додаток Г](#_2s8eyo1) Файл, Що Відповідає За Завантаження до файлу результатів 16

# ОПИС ЗАВДАННЯ

Завдання полягає в обчисленні сортування масиву. Основні етапи, передбачені даним варіантом:

1. Ввід:

* Невідсортований масив

1. Вивід:

* Відсортований масив
* Гістограма значень масиву, де вісь абсцис це індекс, вісь ординат, порівняльне значення з мінімальним елементом.
* Кількість порівнянь(ітерацій) при сортуванні для кожного методу

# КОРОТКИЙ ВИКЛАД ТЕОРІЇ

Для реалізації даної задачі був використаний сучасний набір бібліотек для побудови нитивного UI – Jetpack Compose[1]. Jetpack Compose — це тулкіт з реалізацією на мові програмування Kotlin з відкритим вихідним кодом [2], розроблений для створення додатків на кількох платформах, таких як iOS, Android, а також веб-додатків, використовуючи ту саму базу коду. Особливістю, заради якої був написаний Jetpack є синтаксис, який використовує “Composable” функції(як компонентами з бібліотеки React), замість традиційного XML синтаксису для розробки UI компонентів. Основним аргументом використання XML є те, що відокремлюється пласт UI від логіки. В той час, як бібліотеки по типу Jetpack Compose та React фокусуються на принципі “Separation of concerns”, де основним блоком для будівництва програми виступає компонент, який інкапсулює в собі інші компоненти, свою логіку та логіку представлення UI.

Сама бібліотека відносно нова, але вже встигла здобути популярність, в 2021 році 28 липня вийшов перший stable реліз бібліотеки[3]. А минулого тижня дизайн система Material3 була додана у список стабільних бібліотек для compose екосистеми.[4]

# ОПИС АЛГОРИТМІВ

Дана задача передбачала використання таких алгоритмів, як **pancake sort, selection sort, merge sort, insertion sort та quick sort**. Наведемо короткий опис кожного з них.

**Pancake sort** — це математична задача сортування невпорядкованої стопки млинців за розміром, коли лопатку можна вставити в будь-яку точку стопки та використати для перевертання всіх млинців над нею.

Складність - O(n2)

**Selection sort** — це алгоритм сортування порівняння на місці. Він має часову складність O(n2), що робить його неефективним для великих списків і, як правило, працює гірше, ніж подібне сортування вставкою. Сортування вибору відоме своєю простотою та має переваги в продуктивності порівняно зі складнішими алгоритмами в певних ситуаціях, особливо коли допоміжна пам’ять обмежена.

**Merge sort** (також зазвичай пишеться як mergesort) — це ефективний алгоритм сортування загального призначення на основі порівняння. Більшість реалізацій створюють стабільне сортування, що означає, що порядок рівних елементів однаковий у вхідних і вихідних даних. Складність O(n log n)

**Insertion sort** — це простий алгоритм сортування, який створює остаточний відсортований масив (або список) по одному елементу шляхом порівняння. Він набагато менш ефективний для великих списків, ніж більш просунуті алгоритми, такі як швидке сортування, сортування купою або сортування злиттям. Однак сортування вставкою надає кілька переваг. Складність O(n2)

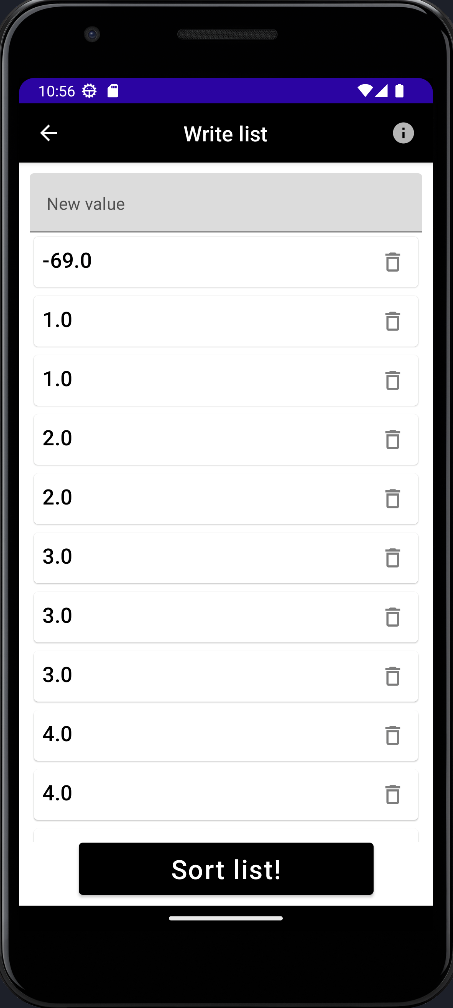
**Quick sort** - це алгоритм сортування на місці. Розроблений британським комп’ютерним науковцем Тоні Хоаром у 1959 році [5] і опублікований у 1961 році , він досі є широко використовуваним алгоритмом для сортування. При правильній реалізації він може бути дещо швидшим, ніж сортування злиттям, і приблизно в два-три рази швидшим, ніж сортування в купі.

# ОПИС РОБОТИ З ПРОГРАМОЮ ТА ЗНІМКИ ЕКРАНУ

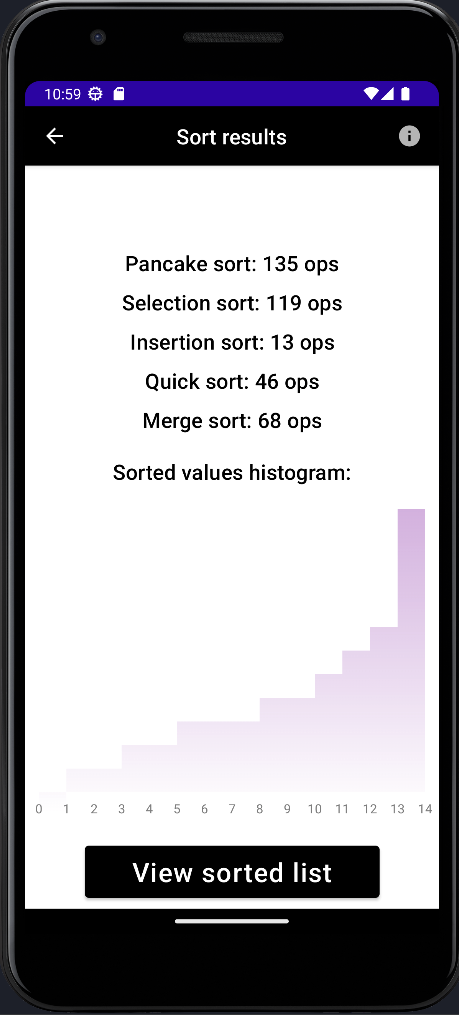
1. Початковий екран - передбачає вибір способу введення масиву. Кнопка “Load from file” відкриває список документів з MIME типом application/json.



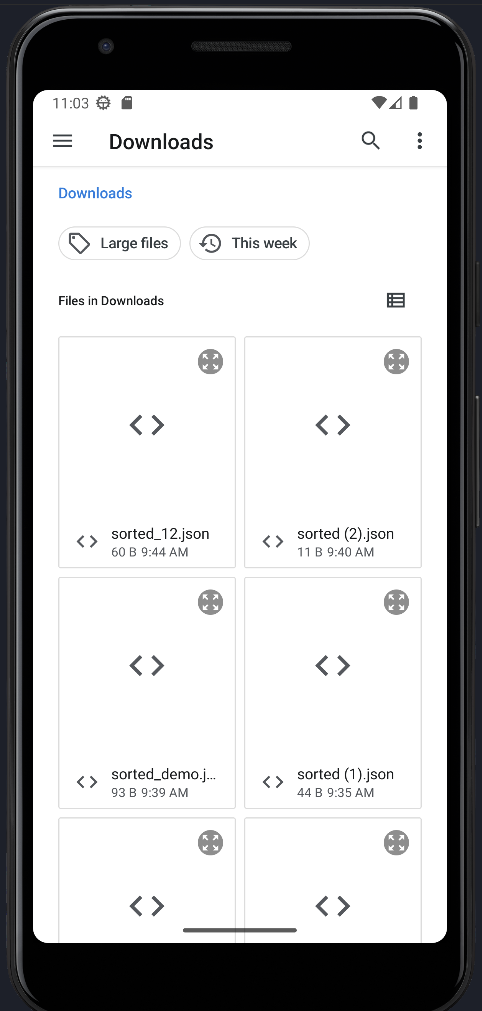
1. Екран введення масиву вручну передбачає введення елементів списку по одному, присутня можливість видалення елементів. Зліва вверху є кнопка «назад», яка переходить на головну сторінку, введений користувачем масив зберігається в state-і застосунку. Кнопка “sort list” переходить на сторінку з результатами сортування. Кнопка справа вгорі “I” переходить на сторінку з інформацією про розробника застосунку.



1. Екран з результатами сортування демонструє алгоритми сортування, кількість операцій порівняння та гістограму значень відсортованого масиву. Кнопка “View sorted list” переходить на сторінку з відсортованим масивом. Кнопка “Назад” та “I” переходять назад та на сторінку автора відповідно.



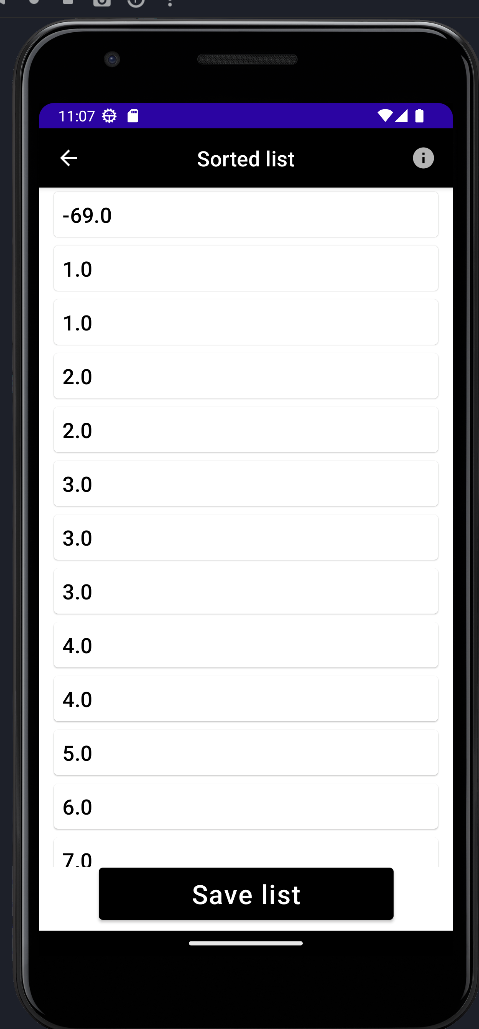
1. Екран введення масиву з файлу, який зʼявляєтся після натискання кнопки “input from file” в головному меню, може відрізнятися залежно від версії Android та наповнення папок користувачем. На цьому екрані пропонується вибрати **один** файл для завантаження його в застосунок.



1. Екран з даними про виконавця.



1. Екран з відсортованим масивом. Кнопка “save list” відкриває екран, який надає ОС Android для вибору, куди завантажити файл.



# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. <https://developer.android.com/jetpack/compose>
2. https://android.googlesource.com/platform/frameworks/support/+/refs/heads/androidx-main/compose/README.md
3. <https://android-developers.googleblog.com/2021/07/jetpack-compose-announcement.html>
4. <https://material.io/blog/material-3-compose-stable>
5. Computer History Museum. Archived from [the original](http://www.computerhistory.org/fellowawards/hall/bios/Antony,Hoare/) on 3 April 2015. Retrieved 22 April 2015.

# ДОДАТКИ

## ДОДАТОК А

**ФРАГМЕНТ КОДУ ГОЛОВНОГО ФАЙЛУ**

enum class Screen(val *title*: String) {  
 *Main*(title = "Rabbit sort"),  
 *WriteList*(title = "Write list"),  
 *SortResults*(title = "Sort results"),  
 *Author*(title = "Author"),  
 *SortedList*(title = "Sorted list")  
}  
  
  
@Composable  
fun RabbitSorterApp() {  
 val *navController* = *rememberNavController*()  
 val *viewModel*: ListViewModel = viewModel()  
 val *backStackEntry* by navController.currentBackStackEntryAsState()  
 val *currentScreen* = Screen.valueOf(  
 *backStackEntry*?.*destination*?.*route* ?: Screen.*Main*.*name* )  
  
 Scaffold(  
 topBar = **{** TopBar(  
 title = currentScreen.*title*,  
 canBack = currentScreen != Screen.*Main*,  
 backClick = **{** navController.navigateUp() **}**,  
 canAuthor = currentScreen != Screen.*Author*,  
 authorClick = **{** navController.navigate(Screen.*Author*.*name*) **}**,  
 )  
 **}** ) **{** innerPadding **->** val *uiState* by viewModel.*uiState*.collectAsState()  
 NavHost(  
 navController = navController,  
 startDestination = Screen.*Main*.*name*,  
 modifier = Modifier.padding(innerPadding)  
 ) **{** composable(Screen.*Main*.*name*) **{** MainPage(loadFromFileClick = **{** viewModel.saveInputList(**it**)  
 navController.navigate(Screen.*SortResults*.*name*)  
 **}**, inputByHandClick = **{** navController.navigate(Screen.*WriteList*.*name*) **}** )  
 **}** composable(Screen.*Author*.*name*) **{** AuthorPage()  
 **}** composable(Screen.*SortedList*.*name*) **{** *SortedListView*(list = SortedList(uiState.*list*).value())  
 **}** composable(Screen.*WriteList*.*name*) **{** ListInputView(  
 inputList = uiState.*list*,  
 onClickSort = **{** navController.navigate(Screen.*SortResults*.*name*) **}**,  
 saveTempList = **{** viewModel.saveInputList(**it**) **}**)  
 **}** composable(Screen.*SortResults*.*name*) **{** val *sorted* = SortedList(uiState.*list*)  
 SortResultsPage(  
 SortResultsPageProps(  
 sortedList = *sorted*.value(),  
 **{** navController.navigate(Screen.*SortedList*.*name*) **}**,  
 sortStats = *sorted*.sortStats()  
 )  
 )  
 **}  
 }  
 }**}

## ДОДАТОК Б

**ФРАГМЕНТ КОДУ ФАЙЛУ, ЩО ВІДПОВІДАЄ ЗА ВВІД МАСИВУ З ФАЙЛУ**

@Composable  
fun LoadFile(inputByFileClick: (l: List<Double>) -> Unit) {  
 val *context* = *LocalContext*.*current* val *contentResolver* = *context*.*contentResolver* val *launcher* =  
 *rememberLauncherForActivityResult*(contract = ActivityResultContracts.GetContent()) **{** uri **->** uri?.let **{** val *content* = BufferedReader(  
 InputStreamReader(  
 contentResolver.openInputStream(  
 **it** )  
 )  
 ).readText()  
 inputByFileClick(  
 *jsonToListDouble*(  
 *content* )  
 )  
 **}  
 }** MainButton("Load from file") **{** launcher.launch("application/json")  
 **}**}  
  
fun jsonToListDouble(t: String): List<Double> {  
 return Json.parseToJsonElement(t).*jsonArray*.map **{  
 it**.*jsonPrimitive*.*double* **}**}

## ДОДАТОК В

**ФРАГМЕНТ КОДУ ФАЙЛУ, ЩО ВІДПОВІДАЄ ЗА ВІДОБРАЖЕННЯ ГІСТОГРАМИ**

BarGraph(  
 header = **{** *Text*(  
 "Sorted values histogram:",  
 modifier = Modifier  
 .fillMaxWidth()  
 .padding(*PaddingValues*(bottom = 10.*dp*)),  
 style = MaterialTheme.*typography*.*h6*,  
 textAlign = TextAlign.*Center* )  
 **}**,  
 dataList = props.*sortedList*,  
)

## ДОДАТОК Г

**ФАЙЛ, ЩО ВІДПОВІДАЄ ЗА ЗАВАНТАЖЕННЯ ДО ФАЙЛУ РЕЗУЛЬТАТІВ**

@Composable  
fun ListWithSave(list: List<Double>) {  
 val *context* = *LocalContext*.*current* val *contentResolver* = *context*.*contentResolver* var *uploadLauncher* = rememberLauncherForActivityResult(  
 contract = ActivityResultContracts.CreateDocument("application/json"),  
 onResult = **{** it **->** it?.let **{** val *writer* = OutputStreamWriter(contentResolver.openOutputStream(**it**))  
 *writer*.write(  
 Json.encodeToString(  
 Json.*serializersModule*.serializer(),  
 list  
 )  
 )  
 *writer*.close()  
 **}  
 }**)  
 WithBottomButton(text = "Save list", callback = **{** uploadLauncher.launch("sorted.json")  
 **}**) **{** Column(  
 modifier = Modifier  
 .fillMaxSize()  
 .padding(horizontal = 10.*dp*)  
 .padding(*PaddingValues*(bottom = 60.*dp*)),  
 horizontalAlignment = Alignment.*CenterHorizontally* ) **{** LazyColumn **{** items(list) **{** item **->** PreviewNumber(number = item)  
 **}  
 }  
 }  
 }**}