# Објектно оријентисано програмирање 2

- Индивидуални пројекат

Logo

Description automatically generated

Немања Мајсторовић SV10/2020

# Рад УИ подсистема

Први аргумент командне линије је и обавезни а то је подразумевана путања до улазног фајла, уколико се он не унесе, долази до грешке. Други параметар је необавезан и везује се за критеријум сортирања, уколико корисник у графичкој спрези не одабере ниједан критеријум користиће се овај критеријум, а уколико га не унесе ни у аргументима командне линије сортираће се према параметру gateNo.

# Класе, изузеци и слободне функције

Text

Description automatically generated

Класа служи за обрађивање улазних аргумената, при чему јој се прослеђује број аргумената и сами аргументи који су и њени атрибути. Функција getPath служи за добављање подразумеване путање из аргумената док getFunctions формира ламбда функције за поређење и добављање вредности.

A screenshot of a computer

Description automatically generated with medium confidence

Класа која служи за чување промена, атрибут index\_changes служи за чување индекса летова у односу на претходну итерацију, док changes чува стање вектора у свакој итерацији, comparisons\_per\_iter и movements\_per\_iter чувају број поређења и померања у свакој итерацији, методе addIndexChanges I addChanges служе за додавање тренутних стања у свакој итерацији и raise методе служе за увећавање бројача у одређеној итерацији за дати индекс.

Text

Description automatically generated

Базна апстрактна класа за сортирање, садржи атрибуте за бројање промена, померања и итерација, Changes атрибут за праћење промена, саме податке који се мењају, ламбда функцију за поређење, као и назив самог алгоритма, сви атрибути имају идентификатор приступа protected како би им се могло приступити из наслеђене класе. Функција Sort је виртуелна метода коју свака класа наследница треба да override-ује, даље следе getter методе, и функција за постављање броја поређења на почетну вредност.

Text

Description automatically generated

Класа Flight описује ентитет лета, осим getter функција имамо преклапање оператора за излазни и улазни ток података, помоћну статичку функцију која служи за учитавање скупа летова из фајла на задатој путањи у вектор flights .

Text

Description automatically generatedText

Description automatically generated

Класа MyWindow служи за одабир параметара за визуелизацију односно путања до улазног фајла, путања до излазног фајла, алгоритам за сортирање, и критеријум за сортирање. Од осталих атрибута имамо Boolean атрибуте који нам говоре која су дугмад притиснута. Методе са префиксом cb\_ служе за увезивање дугмади са њиховим callback функцијама, док саме callback функције их само обоје и све остале Boolean вредности за притиснуту дугмад у истој групи поставља на нетачну вредност (симулира се radio button). Притиском на дугме Next прелази се на нов прозор за визуелизацију.

Text

Description automatically generated

DrawingWindow служи за визуелизацију сортирања, поседује next\_button односно дугме за исцртавање стања у следећој итерацији, exit\_button за искључење апликације, Boolean вредност за проверу да ли је дугме притиснуто, показивач на искоришћени Sort, сам вектор летова који су бивали мењани, тачке који служе за представљање позиције у итерацији, outs који представљају лабеле за приказ распореда летова на почетку и на крају сортирања, линије које спајају тачке и представљају визуелизацију, бројач итерација који бележи до које је итерације корисник стигао, callback функција next која учитава следећу итерацију, додаје нове линији, приказује их и затим повећава бројач итерација за 1, exit callback функција која искључује апликацију, connect\_dots метода која креира линију између две тачке и додаје је на прозор, getDotAt функцију која добавља тачку из “дводимензионалног” низа на основу реда и колоне у којој се налази(w представља ширину матрице односно број колона).

# Структура аргумената командне линије и пример коришћења

Graphical user interface, text, application

Description automatically generated

Невалидан унос аргумената, мора постојати подразумевана путања до улазног фајла.

A picture containing graphical user interface

Description automatically generated

Валидан унос аргумената, путања до улазног фајла је наведена, критеријум сортирања уколико корисник у графичкој спрези не одабере биће подразумевано gateNo.

A picture containing table

Description automatically generated

Валидан унос аргумената, путања је наведена, критеријум сортирања уколико корисник у графичкој спрези не одабере биће flightNo.

# Структура излазне датотеке

Text, table

Description automatically generated with medium confidence

Исписује се стање у свакој итерацији и одваја низом карактера “-“ и на крају се исписује број итерација, број померања као и број поређења.

# Опис и анализа одабраног алгоритма за сортирање

QuickSort је још један од “подели па владај” алгоритама, суштина је да бира пивот и да дели дати низ у зависности од пивота. У овом пројекту за пивот је увек биран последњи елемент. Кључни елемент алгоритма је партиционисање низа, последица овога је да су након партиционисања сви елементи мањи од пивота налазе са једне стране док се сви елементи већи од пивота налазе са супротне стране. Партиционисање се извршава у линеарној временској комплексности односно O(n). Временска комплексност QuickSort-а варира јер алгоритам умногоме зависи од избора пивота, па је у најбољем случају она једнака O(nlogn) док је у најгорем O(). До најгоре ситуације долази када се у сваком избору пивота бира најмањи или највећи елемент односно у нашем случају када је низ већ сортиран. У пракси до најгорег случаја се изузетно ретко долази.

# Напредни ОО концепти

- SelectionSort и QuickSort наслеђују класу Sort

- MyWindow I DrawingWindow наслеђују класу Window

- преклапање оператора << и >> у класи Flight

- наслеђивање и override-овање виртуелне sort методе из базне класе код класа QuickSort и SelectionSort

# Графичка спрега

Graphical user interface

Description automatically generated

MyWindow – поље за унос путање до улазног фајла, поље за унос путање до излазног фајла, 2 радио дугмета за одабир алгоритма, 4 радио дугмета за одабир критеријума сортирања, дугме Next за прелазак у следећи прозор за визуелизацију.

Table

Description automatically generated

DrawingWindow – прозор за визуелизацију, са леве стране су лабеле са почетним распоредом, док су са десне са крајњим распоредом, између се налазе тачке које репрезентују позиције у датој итерацији, док линије визуализују премештање елемената у датој итерацији, дугме next приказује следећу итерацију алгоритма, дугме Exit врши излаз из апликације.

# Тест случајеви

Diagram, engineering drawing

Description automatically generated

Diagram, engineering drawing

Description automatically generated

Diagram

Description automatically generated

Diagram, engineering drawing

Description automatically generated with medium confidence

# Излазни резултати

Пример 1. (LAS VEGAS, DALLAS, LONDON, MEXICO, LONDON, PARIS, PRAGUE, TORONTO, SYDNEY, WASHINGTON)

### Selection sort

|  |  |  |
| --- | --- | --- |
| редни број итерације | број поређења | број замена |
| 1 | 9 | 1 |
| 2 | 8 | 1 |
| 3 | 7 | 1 |
| 4 | 6 | 1 |
| 5 | 5 | 1 |
| 6 | 4 | 1 |
| 7 | 3 | 1 |
| 8 | 2 | 1 |
| 9 | 1 | 1 |
| Σ | 45 | 9 |

### QuickSort

|  |  |  |
| --- | --- | --- |
| редни број итерације | број поређења | број замена |
| 1 | 9 | 10 |
| 2 | 8 | 8 |
| 3 | 6 | 7 |
| 4 | 5 | 6 |
| 5 | 4 | 3 |
| 6 | 1 | 1 |
| 7 | 1 | 1 |
| Σ | 34 | 36 |

Пример 2. (LAS VEGAS, LONDON, LONDON, MEXICO, PARIS, PRAGUE, SYDNEY, TORONTO, WASHINGTON)

### Selection sort

|  |  |  |
| --- | --- | --- |
| редни број итерације | број поређења | број замена |
| 1 | 8 | 1 |
| 2 | 7 | 1 |
| 3 | 6 | 1 |
| 4 | 5 | 1 |
| 5 | 4 | 1 |
| 6 | 3 | 1 |
| 7 | 2 | 1 |
| 8 | 1 | 1 |
| Σ | 36 | 8 |

### QuickSort

|  |  |  |
| --- | --- | --- |
| редни број итерације | број поређења | број замена |
| 1 | 8 | 9 |
| 2 | 7 | 8 |
| 3 | 6 | 7 |
| 4 | 5 | 6 |
| 5 | 4 | 5 |
| 6 | 3 | 4 |
| 7 | 2 | 2 |
| Σ | 35 | 41 |

Пример 3. (LAS VEGAS , DALLAS, LONDON, MEXICO, LONDON, PARIS, PRAGUE, TORONTO, SYDNEY, WASHINGTON, MOSCOW, VIENNA, BELGRADE, BERN, NEW YORK, ROME, ATHENA, OSLO, HELSINKI)

### SelectionSort

|  |  |  |
| --- | --- | --- |
| редни број итерације | број поређења | број замена |
| 1 | 18 | 1 |
| 2 | 17 | 1 |
| 3 | 16 | 1 |
| 4 | 15 | 1 |
| 5 | 14 | 1 |
| 6 | 13 | 1 |
| 7 | 12 | 1 |
| 8 | 11 | 1 |
| 9 | 10 | 1 |
| 10 | 9 | 1 |
| 11 | 8 | 1 |
| 12 | 7 | 1 |
| 13 | 6 | 1 |
| 14 | 5 | 1 |
| 15 | 4 | 1 |
| 16 | 3 | 1 |
| 17 | 2 | 1 |
| 18 | 1 | 1 |
| Σ | 171 | 18 |

### QuickSort

|  |  |  |
| --- | --- | --- |
| редни број итерације | број поређења | број замена |
| 1 | 18 | 5 |
| 2 | 3 | 1 |
| 3 | 2 | 3 |
| 4 | 1 | 2 |
| 5 | 13 | 2 |
| 6 | 11 | 7 |
| 7 | 5 | 5 |
| 8 | 3 | 2 |
| 9 | 1 | 1 |
| 10 | 4 | 5 |
| 11 | 3 | 2 |
| 12 | 1 | 1 |
| Σ | 65 | 36 |

## Упоредни укупни резултати

### Поређења

|  |  |  |
| --- | --- | --- |
|  | SELECTION | QUICK |
| ПРИМЕР1 | 45 | 34 |
| ПРИМЕР2 | 36 | 35 |
| ПРИМЕР3 | 171 | 65 |

### Померања

|  |  |  |
| --- | --- | --- |
|  | SELECTION | QUICK |
| ПРИМЕР1 | 9 | 36 |
| ПРИМЕР2 | 8 | 41 |
| ПРИМЕР3 | 18 | 36 |

### Збир броја операција

|  |  |  |
| --- | --- | --- |
|  | SELECTION | QUICK |
| ПРИМЕР1 | 54 | 70 |
| ПРИМЕР2 | 44 | 76 |
| ПРИМЕР3 | 189 | 101 |

Из примера 1 увиђамо да QuickSort има више операција, међутим мање итерација, у примеру 2 који је уједно и најгори случај за QuickSort(јер су сви елементи већ сортирани) уочавамо да се QuickSort малтене своди на SelectionSort као и да има знатно више померања у односу на Selection што га и чини горим у овој ситуацији, међутим у примеру 3 где смо узели 19 градова увиђамо да QuickSort има много мање поређења у односу на SelectionSort док у табели видимо да је и збирни број операција много мањи у односу на SelectionSort. QuickSort не пролази кроз цео низ у свакој итерацији

# Уочени проблеми и ограничења

QuickSort се показао као лош када је низ већ сортиран, међутим у пракси на тај проблем ћемо наићи ретко, уколико се за пивот узастопце бирају вредности које су приближне средњој вредности елемената овај алгоритам има временску комплексност O(nlogn), као и MergeSort, међутим MergeSort захтева додатних O(n) меморије док је QuickSort in-place алгоритам, што значи да му није потребан додатан простор. SelectionSort у свакој итерацији нађе минимални елемент и постави га на почетак међутим да би знао да је тај елемент најмањи он мора проћи кроз све остале несортиране елементе чиме је његова временска комплексност O().