<데이터 구조 및 알고리즘 시각화>

Software Requirements Specification

For < 데이터 구조 및 알고리즘 시각화 시스템>

Version <1.0>

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| **<2024/03/30>** | **<1.0>** | **<문서생성>** | **<황의현>** |
| **<2024/05/06>** | **<1.0>** | **<문서 수정>** | **<황의현>** |
| **<2024/05/12>** | **<0.1>** | **<앱 작성 내역 추가>** | **<황의현>** |
| **<2024/05/26>** | **<0.2>** | **< 앱 작성 내역 수정 및 참고 추가>** | **<황의현>** |

목차

1. 소개 4

1.1 목적 4

1.2 범위 4

2. 요구사항 4

2.1 기능적 요구사항 4

2.2 비기능적 요구사항 4

3. 시스템 구조 4

3.1 사용자 인터페이스 4

3.2 데이터 구조 및 알고리즘 시각화 엔진 4

4. 인터페이스 4

5. 기타 요구사항 5

6. 참고 5

7. 결론 5

8. 앱 작성 내역 5

8.1 MainActivity.java 5

8.2 SortingView.java 9

8.3 activity\_main.xml 11

8.4 Sort.java 12

8.5 sort\_menu.xml 14

Software Requirements Specification

# 소개

## 목적

이 문서의 목적은 데이터 구조와 알고리즘을 시각적으로 보여주는 안드로이드 애플리케이션의 요구사항을 정의하는 것입니다.

## 범위

이 애플리케이션은 사용자가 다양한 데이터 구조와 알고리즘을 선택하고 시각화하여 학습할 수 있는 기능을 제공합니다

# 요구사항

## 기능적 요구사항

사용자는 앱을 실행하고 데이터 구조 및 알고리즘 목록을 볼 수 있어야 합니다.

사용자는 선택한 데이터 구조나 알고리즘을 시각화하여 볼 수 있어야 합니다.

앱은 배열, 연결 리스트, 스택, 큐, 트리, 그래프 등 다양한 데이터 구조를 지원해야 합니다.

앱은 정렬, 검색, 그래프 탐색 등 다양한 알고리즘을 시각화할 수 있어야 합니다.

사용자는 각 시각화된 데이터 구조나 알고리즘에 대한 설명을 볼 수 있어야 합니다.

## 비기능적 요구사항

앱은 직관적이고 사용하기 쉬운 사용자 인터페이스를 제공해야 합니다.

시각화는 성능에 영향을 주지 않아야 하며, 부드럽고 빠른 사용자 경험을 제공해야 합니다.

앱은 안정적이고 오류 없이 작동해야 하며, 예외 상황에 대한 적절한 처리가 이루어져야 합니다.

시각화된 데이터 구조나 알고리즘의 설명은 정확하고 명확해야 합니다.

# 시스템 구조

## 사용자 인터페이스

앱은 메인 화면에 데이터 구조와 알고리즘 목록을 표시합니다.

사용자는 목록에서 항목을 선택하여 시각화를 시작할 수 있습니다.

## 데이터 구조 및 알고리즘 시각화 엔진

앱은 선택한 데이터 구조나 알고리즘을 시각화하기 위한 엔진을 포함합니다.

이 엔진은 사용자의 입력을 받아 시각화된 결과를 생성합니다.

# 인터페이스

메인 화면: 데이터 구조와 알고리즘 목록을 표시하는 화면

시각화 화면: 선택한 데이터 구조나 알고리즘을 시각화하는 화면

# 기타 요구사항

앱은 오프라인에서도 사용할 수 있어야 합니다.

사용자는 시각화된 결과를 공유할 수 있는 기능을 제공받아야 합니다.

# 참고

이펙티브 자바(Effective Java) – 조슈아 블로치

안드로이드 개발자 가이드(Android Developer Guide) - 안드로이드 공식 문서

알고리즘 및 데이터 구조 관련 온라인 자료

# 결론

이 데이터 구조 및 알고리즘 시각화 안드로이드 애플리케이션은 사용자가 학습하고 이해할 수 있는 다양한 데이터 구조와 알고리즘을 제공합니다. 사용자는 직관적이고 사용하기 쉬운 인터페이스를 통해 앱을 쉽게 탐색하고, 선택한 데이터 구조 및 알고리즘을 시각화하여 확인할 수 있습니다. 이를 통해 사용자는 오프라인이거나 온라인에서 모바일 디바이스를 통해 언제 어디서든 데이터 구조와 알고리즘을 학습하고 공유할 수 있습니다.

참고로, 시각화된 결과와 각 데이터 구조 및 알고리즘에 대한 명확하고 이해하기 쉬운 설명은 사용자가 학습을 도와줄 것입니다. 이러한 요구사항을 충족시키는 안드로이드 애플리케이션을 개발함으로써 사용자들에게 탁월한 학습 경험을 제공할 것입니다.

# 일정

2024/05/06 – UI 구현

2024/05/26 – 알고리즘 구현

2024/06/01 – 메모리 누수 방지

2024/06/02 – 발표 자료 생성

# 앱 작성 내역

## MainActivity.java

package com.cookandroid.algcan;  
  
import android.os.Bundle;  
import android.os.Handler;  
import android.view.Menu;  
import android.view.MenuItem;  
import android.view.View;  
import android.widget.Button;  
import android.widget.EditText;  
import android.widget.TextView;  
import androidx.appcompat.app.AppCompatActivity;  
import java.lang.ref.WeakReference;  
import java.util.Arrays;  
  
public class MainActivity extends AppCompatActivity {  
 private EditText inputEditText;  
 private TextView resultTextView;  
 private SortingView sortingView;  
 private Handler handler;  
 private Runnable sortingRunnable;  
 private int[] numbers;  
 private int selectedSortAlgorithm;  
 private WeakReference<MainActivity> activityRef; // MainActivity의 약한 참조  
  
 // 정렬 알고리즘 상수  
 private static final int *SELECTION\_SORT* = 0;  
 private static final int *INSERTION\_SORT* = 1;  
 private static final int *BUBBLE\_SORT* = 2;  
 private static final int *QUICK\_SORT* = 3;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_main*);  
  
 inputEditText = findViewById(R.id.*inputEditText*);  
 resultTextView = findViewById(R.id.*resultTextView*);  
 sortingView = findViewById(R.id.*sortingView*);  
 handler = new Handler();  
 activityRef = new WeakReference<>(this); // MainActivity의 약한 참조  
  
 // 시작 버튼을 찾고 클릭 이벤트를 설정  
 Button startButton = findViewById(R.id.*startButton*);  
 startButton.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 startSorting();  
 }  
 });  
 }  
  
 @Override  
 public boolean onCreateOptionsMenu(Menu menu) {  
 getMenuInflater().inflate(R.menu.*sort\_menu*, menu);  
 return true;  
 }  
  
 @Override  
 public boolean onOptionsItemSelected(MenuItem item) {  
 int id = item.getItemId();  
 resultTextView.setText("정렬 결과: ");  
 if (id == R.id.*action\_selection\_sort*) {  
 selectedSortAlgorithm = *SELECTION\_SORT*;  
 } else if (id == R.id.*action\_insertion\_sort*) {  
 selectedSortAlgorithm = *INSERTION\_SORT*;  
 } else if (id == R.id.*action\_bubble\_sort*) {  
 selectedSortAlgorithm = *BUBBLE\_SORT*;  
 } else if (id == R.id.*action\_quick\_sort*) {  
 selectedSortAlgorithm = *QUICK\_SORT*;  
 }  
 return super.onOptionsItemSelected(item);  
 }  
  
 private void startSorting() {  
 // 입력된 숫자를 가져와서 배열로 변환  
 String inputString = inputEditText.getText().toString();  
 String[] inputArray = inputString.split(" ");  
 numbers = new int[inputArray.length];  
 for (int i = 0; i < inputArray.length; i++) {  
 numbers[i] = Integer.*parseInt*(inputArray[i]);  
 }  
  
 // 선택한 정렬 알고리즘에 따라 실행  
 switch (selectedSortAlgorithm) {  
 case *SELECTION\_SORT*:  
 startSelectionSortStep();  
 break;  
 case *INSERTION\_SORT*:  
 startInsertionSortStep();  
 break;  
 case *BUBBLE\_SORT*:  
 startBubbleSortStep();  
 break;  
 case *QUICK\_SORT*:  
 startQuickSortStep();  
 break;  
 }  
 }  
  
 // 선택 정렬 실행  
 private void startSelectionSortStep() {  
 int[] i = {0};  
 int[] minIndex = {0};  
 boolean[] swapped = {false};  
 sortingRunnable = new Runnable() {  
 @Override  
 public void run() {  
 Sort.*selectionSortStep*(numbers, i, minIndex, swapped).run();  
 SortingView sortingView = activityRef.get().sortingView; // SortingView의 약한 참조  
 if (sortingView != null) {  
 sortingView.setArrayToSort(numbers);  
 }  
 if (i[0] < numbers.length - 1) {  
 handler.postDelayed(this, 100);  
 } else {  
 resultTextView.setText("정렬 결과: " + Arrays.*toString*(numbers));  
 }  
 }  
 };  
  
 handler.post(sortingRunnable);  
 }  
  
 // 삽입 정렬 실행  
 private void startInsertionSortStep() {  
 int[] i = {0};  
 int[] j = {0};  
 sortingRunnable = new Runnable() {  
 @Override  
 public void run() {  
 Sort.*insertionSortStep*(numbers, i, j).run();  
 SortingView sortingView = activityRef.get().sortingView; // SortingView의 약한 참조  
 if (sortingView != null) {  
 sortingView.setArrayToSort(numbers);  
 }  
 if (!isSorted(numbers)) {  
 handler.postDelayed(this, 100);  
 } else {  
 resultTextView.setText("정렬 결과: " + Arrays.*toString*(numbers));  
 }  
 }  
 };  
  
 handler.post(sortingRunnable);  
 }  
  
 // 버블 정렬 실행  
 private void startBubbleSortStep() {  
 int[] i = {0};  
 int[] j = {0};  
 sortingRunnable = new Runnable() {  
 @Override  
 public void run() {  
 Sort.*bubbleSortStep*(numbers, i, j).run();  
 SortingView sortingView = activityRef.get().sortingView; // SortingView의 약한 참조  
 if (sortingView != null) {  
 sortingView.setArrayToSort(numbers);  
 }  
 if (!isSorted(numbers)) {  
 handler.postDelayed(this, 100);  
 } else {  
 resultTextView.setText("정렬 결과: " + Arrays.*toString*(numbers));  
 }  
 }  
 };  
  
 handler.post(sortingRunnable);  
 }  
  
 // 퀵 정렬 실행  
 private void startQuickSortStep() {  
 int low = 0;  
 int high = numbers.length - 1;  
 sortingRunnable = new Runnable() {  
 @Override  
 public void run() {  
 Sort.*quickSortStep*(numbers, low, high, sortingView).run();  
 SortingView sortingView = activityRef.get().sortingView; // SortingView의 약한 참조  
 if (sortingView != null) {  
 sortingView.setArrayToSort(numbers);  
 }  
 if (!isSorted(numbers)) {  
 handler.postDelayed(this, 100);  
 } else {  
 resultTextView.setText("정렬 결과: " + Arrays.*toString*(numbers));  
 }  
 }  
 };  
 handler.post(sortingRunnable);  
 }  
  
 // 배열이 정렬되었는지 확인하는 유틸리티 메서드  
 private boolean isSorted(int[] array) {  
 for (int i = 0; i < array.length - 1; i++) {  
 if (array[i] > array[i + 1]) {  
 return false;  
 }  
 }  
 return true;  
 }  
}

## SortingView.java

package com.cookandroid.algcan;  
  
import android.content.Context;  
import android.graphics.Canvas;  
import android.graphics.Paint;  
import android.util.AttributeSet;  
import android.view.View;  
import java.lang.ref.WeakReference;  
  
public class SortingView extends View {  
 private int[] arrayToSort;  
 private Paint paint;  
 private WeakReference<MainActivity> activityRef; // MainActivity의 약한 참조  
  
 int pivotIndex = -1;  
  
 public SortingView(Context context, AttributeSet attrs) {  
 super(context, attrs);  
 paint = new Paint();  
 arrayToSort = new int[]{};  
 if (context instanceof MainActivity) {  
 activityRef = new WeakReference<>((MainActivity) context);  
 }  
 }  
  
 public void setArrayToSort(int[] arrayToSort) {  
 this.arrayToSort = arrayToSort;  
 invalidateView(); // 변경된 경우에만 View 다시 그리기  
 }  
  
 public void setPivotIndex(int pivotIndex){  
 this.pivotIndex = pivotIndex;  
 invalidateView(); // 변경된 경우에만 View 다시 그리기  
 }  
  
 // View 다시 그리기 호출 최적화  
 private void invalidateView() {  
 MainActivity activity = activityRef.get();  
 if (activity != null) {  
 activity.runOnUiThread(() -> invalidate());  
 }  
 }  
  
 @Override  
 protected void onDraw(Canvas canvas) {  
 super.onDraw(canvas);  
  
 int width = getWidth();  
 int height = getHeight();  
  
 // 배열의 요소가 없을 때는 그리지 않음  
 if (arrayToSort.length == 0) {  
 return;  
 }  
  
 // 막대와 화면의 좌우 마진 설정  
 float margin = 20f;  
 float padding = 5f; // 막대 사이의 패딩  
 float availableWidth = width - 2 \* margin - (arrayToSort.length - 1) \* padding;  
 int barWidth = (int) (availableWidth / arrayToSort.length);  
  
 // 배열의 최대값 찾기  
 int maxValue = Integer.*MIN\_VALUE*;  
 for (int value : arrayToSort) {  
 maxValue = Math.*max*(maxValue, value);  
 }  
  
 // 배열의 각 요소를 막대로 그림  
 for (int i = 0; i < arrayToSort.length; i++) {  
 float left = margin + i \* (barWidth + padding);  
 float barHeight = (float) arrayToSort[i] / maxValue \* height; // 막대의 높이를 배열 값의 비율로 조정  
 float top = height - barHeight;  
 float right = left + barWidth;  
 float bottom = height;  
  
 // 각 막대의 색과 크기 설정  
 if(i == pivotIndex){  
 paint.setColor(0xFFFF0000);  
 }  
 else{  
 paint.setColor(0xFF000000); // 검정색  
 }  
 paint.setStyle(Paint.Style.*FILL*);  
 canvas.drawRect(left, top, right, bottom, paint);  
 }  
 }  
}

## activity\_main.xml

<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="vertical"  
 tools:context=".MainActivity">  
  
 <!-- 버튼: 정렬 시작 -->  
 <Button  
 android:id="@+id/startButton"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="시작"  
 android:layout\_gravity="center\_horizontal"  
 android:layout\_marginTop="16dp" />  
  
 <!-- 정렬할 숫자를 입력하는 에디트텍스트 -->  
 <EditText  
 android:id="@+id/inputEditText"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="16dp"  
 android:layout\_marginTop="8dp"  
 android:layout\_marginEnd="16dp"  
 android:hint="숫자를 입력하세요 (공백으로 구분)"  
 android:inputType="text" />  
  
 <!-- 정렬 결과를 보여주는 텍스트뷰 -->  
 <TextView  
 android:id="@+id/resultTextView"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="정렬 결과:"  
 android:layout\_gravity="center\_horizontal"  
 android:layout\_marginTop="16dp" />  
  
 <!-- 정렬 알고리즘 진행 상황을 보여주는 텍스트뷰 -->  
 <TextView  
 android:id="@+id/progressTextView"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text=""  
 android:layout\_gravity="center\_horizontal"  
 android:layout\_marginTop="16dp" />  
  
 <!-- 정렬 알고리즘을 시각화하는 뷰 -->  
 <com.cookandroid.algcan.SortingView  
 android:id="@+id/sortingView"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="16dp" />  
  
</LinearLayout>

## Sort.java

package com.cookandroid.algcan;  
  
public class Sort {  
  
 // 선택 정렬  
 public static Runnable selectionSortStep(int[] array, int[] i, int[] minIndex, boolean[] swapped) {  
 return () -> {  
 if (i[0] < array.length - 1) {  
 if (!swapped[0]) {  
 minIndex[0] = i[0];  
 for (int j = i[0] + 1; j < array.length; j++) {  
 if (array[j] < array[minIndex[0]]) {  
 minIndex[0] = j;  
 }  
 }  
 int temp = array[minIndex[0]];  
 array[minIndex[0]] = array[i[0]];  
 array[i[0]] = temp;  
 swapped[0] = true;  
 } else {  
 swapped[0] = false;  
 i[0]++;  
 }  
 } else {  
 // 정렬이 완료될 때까지 반복하도록 설정  
 i[0] = 0;  
 minIndex[0] = 0;  
 swapped[0] = false;  
 }  
 };  
 }  
  
 // 삽입 정렬  
 public static Runnable insertionSortStep(int[] array, int[] i, int[] j) {  
 return () -> {  
 if (i[0] < array.length) {  
 int key = array[i[0]];  
 j[0] = i[0] - 1;  
 while (j[0] >= 0 && array[j[0]] > key) {  
 array[j[0] + 1] = array[j[0]];  
 j[0]--;  
 }  
 array[j[0] + 1] = key;  
 i[0]++;  
 }  
 };  
 }  
  
 // 버블 정렬  
 public static Runnable bubbleSortStep(int[] array, int[] i, int[] j) {  
 return () -> {  
 if (i[0] < array.length - 1) {  
 if (j[0] < array.length - 1 - i[0]) {  
 if (array[j[0]] > array[j[0] + 1]) {  
 int temp = array[j[0]];  
 array[j[0]] = array[j[0] + 1];  
 array[j[0] + 1] = temp;  
 }  
 j[0]++;  
 } else {  
 j[0] = 0;  
 i[0]++;  
 }  
 }  
 };  
 }  
  
 // 퀵 정렬  
 public static Runnable quickSortStep(int[] array, int low, int high, SortingView sortingView) {  
 return () -> {  
 if (low < high) {  
 int pivotIndex = *hoarePartition*(array, low, high);  
 if (sortingView != null) {  
 sortingView.setPivotIndex(pivotIndex);  
 sortingView.setArrayToSort(array);  
 }  
 try {  
 Thread.*sleep*(100); // 100밀리초 지연  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 *quickSortStep*(array, low, pivotIndex, sortingView);  
 *quickSortStep*(array, pivotIndex + 1, high, sortingView);  
 }  
 };  
 }  
  
 public static int hoarePartition(int[] arr, int low, int high) {  
 int pivot = arr[low + (high - low) / 2];  
 int i = low - 1;  
 int j = high + 1;  
 while (true) {  
 do {  
 i++;  
 } while (arr[i] < pivot);  
 do {  
 j--;  
 } while (arr[j] > pivot);  
 if (i >= j)  
 return j;  
 int temp = arr[i];  
 arr[i] = arr[j];  
 arr[j] = temp;  
 }  
 }  
}

## sort\_menu.xml

<menu xmlns:android="http://schemas.android.com/apk/res/android">  
 <group android:id="@+id/sort\_group" android:checkableBehavior="single">  
 <item  
 android:id="@+id/action\_selection\_sort"  
 android:title="선택 정렬" />  
 <item  
 android:id="@+id/action\_insertion\_sort"  
 android:title="삽입 정렬" />  
 <item  
 android:id="@+id/action\_bubble\_sort"  
 android:title="버블 정렬" />  
 <item  
 android:id="@+id/action\_quick\_sort"  
 android:title="퀵 정렬" />  
 </group>  
</menu>