1. **소스코드 및 주석**

**1-1.\_1번문제**

#include <iostream>

#include <vector>

using namespace std;

void quickSort(int\* input, int start, int end, int\* p\_pivot\_count) {

if (start >= end) return;

int pivot = start;

int i = start + 1;

int j = end;

\*p\_pivot\_count = \*p\_pivot\_count + 1;

while (i <= j) {

while (input[i] <= input[pivot])

i++;

while (input[j] >= input[pivot] && j > start)

j--;

if (i > j)

{

int temp = input[j];

input[j] = input[pivot];

input[pivot] = temp;

}

else {

int temp = input[j];

input[j] = input[i];

input[i] = temp;

}

quickSort(input, start, j - 1, p\_pivot\_count);

quickSort(input, j + 1, end, p\_pivot\_count);

}

}

int main() {

int input[12] = { 6, 3, 11, 9, 12, 2, 8, 15, 18, 10, 7, 14 };

int inputlen = sizeof(input) / sizeof(int);

int pivot\_count = 0;

quickSort(input, 0, inputlen - 1, &pivot\_count);

for (int i = 0; i < inputlen; i++)

cout << input[i] << ' ';

cout << endl << " Ǻ Ƚ : " << pivot\_count << "ȸ" << endl;

return 0;

}

**1-2. 2번문제**

#include <iostream>

#include <fstream>

#include <random>

using namespace std;

typedef struct Weather {

string date = " ";

int temperature = 0;

} weather;

int RandomPivot(int start, int end) {

random\_device rd;

mt19937\_64 rng(rd());

uniform\_int\_distribution<\_\_int64> dist(start, end);

int pivot = dist(rng);

return pivot;

}

void SwapWeather(Weather\* weathers, int index1ToChange, int index2ToChange) {

if (index1ToChange != index2ToChange) {

Weather temp = weathers[index1ToChange];

weathers[index1ToChange] = weathers[index2ToChange];

weathers[index2ToChange] = temp;

}

}

void RandomQuickSortWeather(Weather\* weathers, int start, int end, int\* p\_pivotCount) {

if (start >= end)

return;

int pivot = RandomPivot(start, end);

SwapWeather(weathers, start, pivot);

pivot = start;

\*p\_pivotCount = \*p\_pivotCount + 1;

int left = start + 1;

int right = end;

while (left <= right) {

while (weathers[left].temperature <= weathers[pivot].temperature)

left++;

while (weathers[right].temperature >= weathers[pivot].temperature && right > start)

right--;

if (left > right)

SwapWeather(weathers, right, pivot);

else

SwapWeather(weathers, right, left);

}

RandomQuickSortWeather(weathers, start, right - 1, p\_pivotCount);

RandomQuickSortWeather(weathers, right + 1, end, p\_pivotCount);

}

void WeatherPrint(Weather\* weathers, int maxIndex) {

for (int i = 0;i <= maxIndex;i++) {

cout << "날짜 : " << weathers[i].date << " | 기온 : " << weathers[i].temperature << endl;

}

}

void MedianQuickSortWeather(Weather\* weathers, int start, int end, int\* p\_pivotCount) {

if (start >= end)

return;

int pivot = (start + end) / 2;

SwapWeather(weathers, start, pivot);

pivot = start;

\*p\_pivotCount = \*p\_pivotCount + 1;

int left = start + 1;

int right = end;

while (left <= right) {

while (weathers[left].temperature <= weathers[pivot].temperature)

left++;

while (weathers[right].temperature >= weathers[pivot].temperature && right > start)

right--;

if (left > right)

SwapWeather(weathers, right, pivot);

else

SwapWeather(weathers, right, left);

}

MedianQuickSortWeather(weathers, start, right - 1, p\_pivotCount);

MedianQuickSortWeather(weathers, right + 1, end, p\_pivotCount);

}

int main() {

int pivotCount\_for\_random = 0, pivotCount\_for\_median = 0, weatherCount = 0;

ifstream fin("daegu\_weather\_2024 (1).txt");

if (fin.is\_open()) {

string s;

int t;

while (!fin.eof())

{

fin >> s >> t;

weatherCount++;

}

fin.clear();

fin.seekg(0, ios::beg);

Weather\* weathers = new Weather[weatherCount];

Weather\* weathers\_for\_random = new Weather[weatherCount];

Weather\* weathers\_for\_median = new Weather[weatherCount];

for (int i = 0; i < weatherCount; i++) {

fin >> weathers[i].date >> weathers[i].temperature;

weathers\_for\_random[i] = weathers[i];

weathers\_for\_median[i] = weathers[i];

}

int maxIndex = weatherCount - 1;

RandomQuickSortWeather(weathers\_for\_random, 0, maxIndex, &pivotCount\_for\_random);

MedianQuickSortWeather(weathers\_for\_median, 0, maxIndex, &pivotCount\_for\_median);

cout << endl << "랜덤 방식/최종 피봇 선택 횟수 : " << pivotCount\_for\_random << "회" << endl;

cout << "중앙값 방식/최종 피봇 선택 횟수 : " << pivotCount\_for\_median << "회" << endl;

cout << endl << "----------------------------------" << endl << endl;

cout << "데이터 정렬 전 --------------------------" << endl;

WeatherPrint(weathers, maxIndex);

cout << "----------------------------------" << endl;

cout << "랜덤 방식 정렬 후 -----------------" << endl;

WeatherPrint(weathers\_for\_random, maxIndex);

cout << "----------------------------------" << endl;

cout << "중앙값 방식 정렬 후 --------------" << endl;

WeatherPrint(weathers\_for\_median, maxIndex);

cout << "----------------------------------" << endl;

}

return 0;

}

**1-3. 3번문제**

#include <iostream>

#include <fstream>

#include <random>

using namespace std;

typedef struct Weather {

string date = " ";

int temperature = 0;

} weather;

void SwapWeather(Weather\* weathers, int index1ToChange, int index2ToChange) {

if (index1ToChange != index2ToChange) {

Weather temp = weathers[index1ToChange];

weathers[index1ToChange] = weathers[index2ToChange];

weathers[index2ToChange] = temp;

}

}

void WeatherPrint(Weather\* weathers, int maxIndex) {

for (int i = 0;i <= maxIndex;i++) {

cout << "날짜 : " << weathers[i].date << " | 기온 : " << weathers[i].temperature << endl;

}

}

void ThreePartQuickSortWeather(Weather\* weathers, int start, int end, int\* p\_pivotCount) {

if (start >= end)

return;

int pivot = (start + end) / 2;

SwapWeather(weathers, start, pivot);

pivot = start;

\*p\_pivotCount = \*p\_pivotCount + 1;

int left = start + 1;

int mid = left;

int right = end;

while (mid <= right) {

if (weathers[mid].temperature < weathers[pivot].temperature) {

SwapWeather(weathers, left, mid);

left++;

mid++;

}

else if (weathers[mid].temperature > weathers[pivot].temperature) {

SwapWeather(weathers, mid, right);

right--;

}

else {

mid++;

}

}

SwapWeather(weathers, pivot, left - 1);

pivot = left - 1;

ThreePartQuickSortWeather(weathers, start, left - 1, p\_pivotCount);

ThreePartQuickSortWeather(weathers, right + 1, end, p\_pivotCount);

}

int main() {

int pivotCount\_for\_three\_part = 0, weatherCount = 0;

ifstream fin("daegu\_weather\_2024 (1).txt");

if (fin.is\_open()) {

string s;

int t;

while (fin >> s >> t) {

weatherCount++;

}

fin.clear();

fin.seekg(0, ios::beg);

Weather\* weathers = new Weather[weatherCount];

Weather\* weathers\_for\_three\_part = new Weather[weatherCount];

for (int i = 0; i < weatherCount; i++) {

fin >> weathers[i].date >> weathers[i].temperature;

weathers\_for\_three\_part[i] = weathers[i];

}

int maxIndex = weatherCount - 1;

ThreePartQuickSortWeather(weathers\_for\_three\_part, 0, maxIndex, &pivotCount\_for\_three\_part);

cout << "3 part positioning 방식/최종 피봇 선택 횟수 : " << pivotCount\_for\_three\_part << "회" << endl;

cout << endl << "----------------------------------" << endl << endl;

cout << "데이터 정렬 전 --------------------------" << endl;

WeatherPrint(weathers, maxIndex);

cout << "----------------------------------" << endl;

cout << "3 part positioning 방식 정렬 후 --------------" << endl;

WeatherPrint(weathers\_for\_three\_part, maxIndex);

cout << "----------------------------------" << endl;

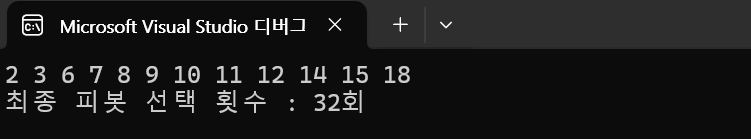
}

return 0;

}

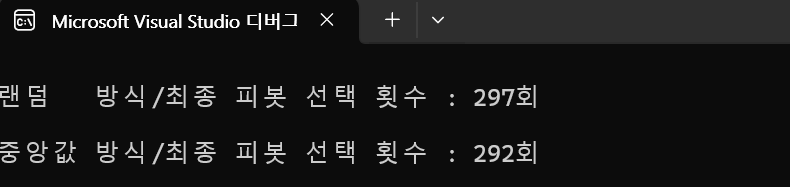
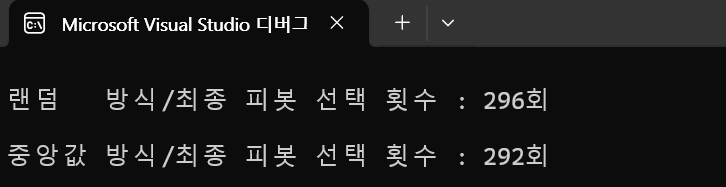
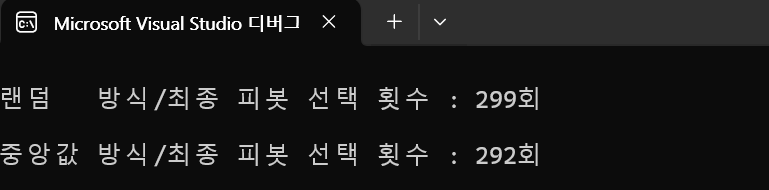
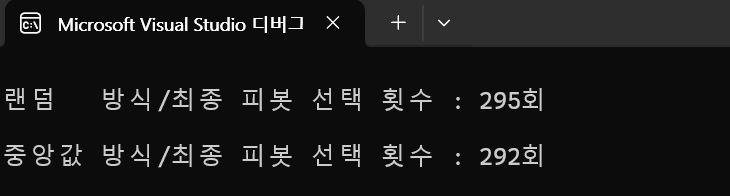
1. **실행화면 캡처**

**2-1.\_1번문제**



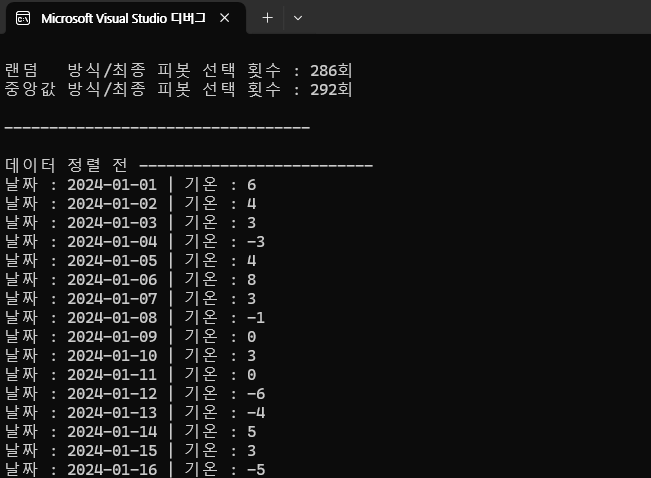
**2-2. 2번문제**

**피봇 선택 횟수 비교 : 랜덤 방식 VS 중앙값 방식**

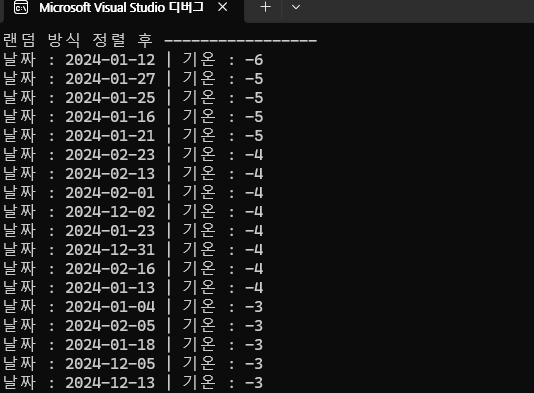


* 랜덤 방식 피봇 선택횟수 --> 약 300회 전후 랜덤한 값
* 중앙값 방식 피봇 선택횟수 --> 292회

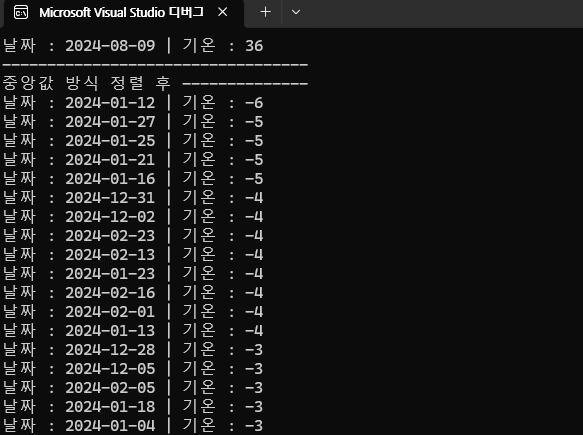
**정렬전 데이터**



**QUICK Sorting 후 (피봇 랜덤 방식)**



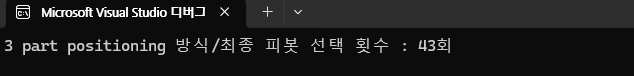
**QUICK Sorting 후 (피봇 중앙값 방식)**



* 제대로 잘 정렬되었음

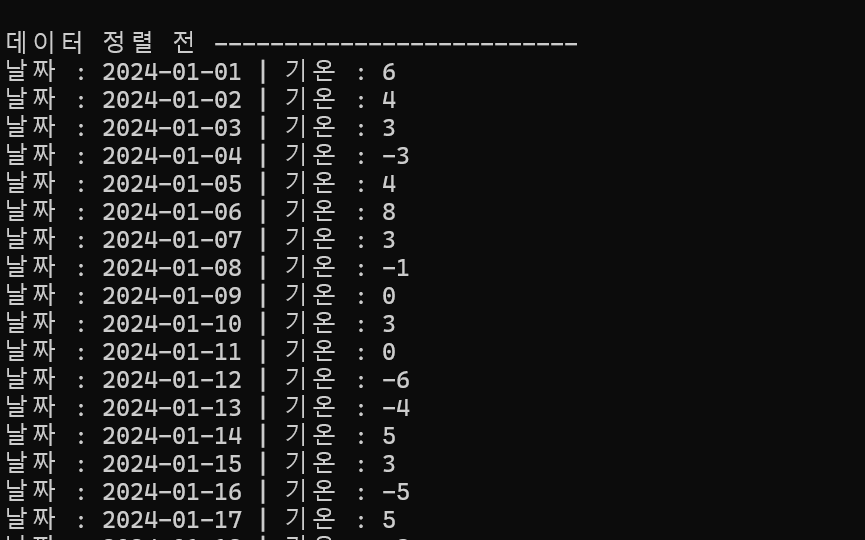
**2-3. 3번문제**

**최종 피벗 선택 횟수**

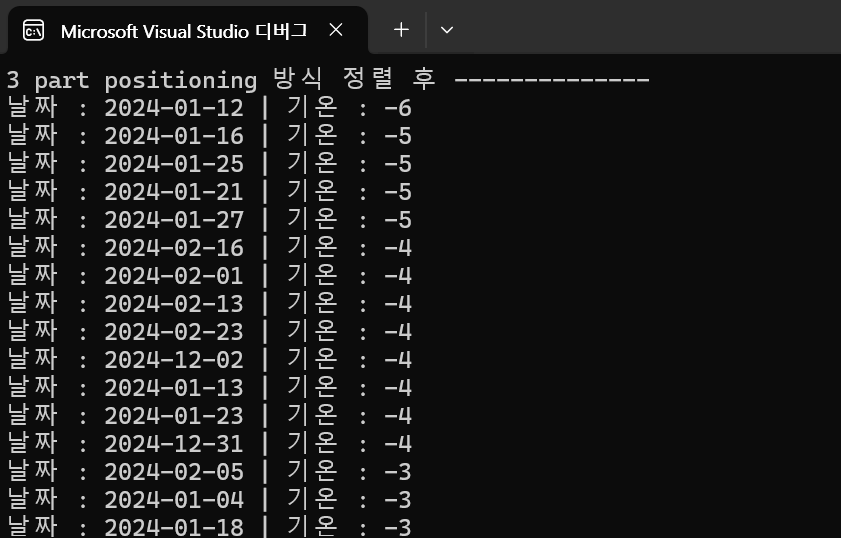
****

* 매우 효율적으로 변했음을 알 수 있다

**정렬 전 데이터**

****

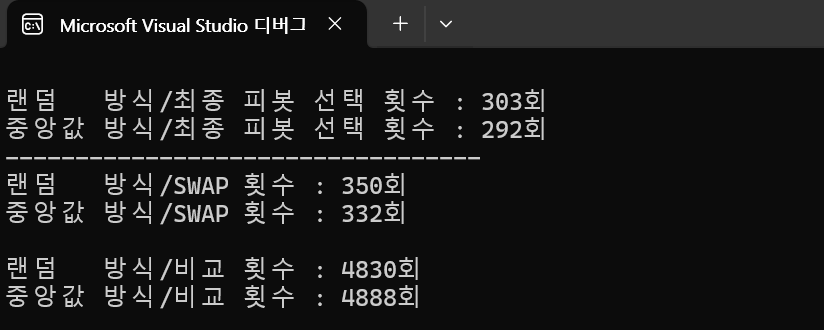
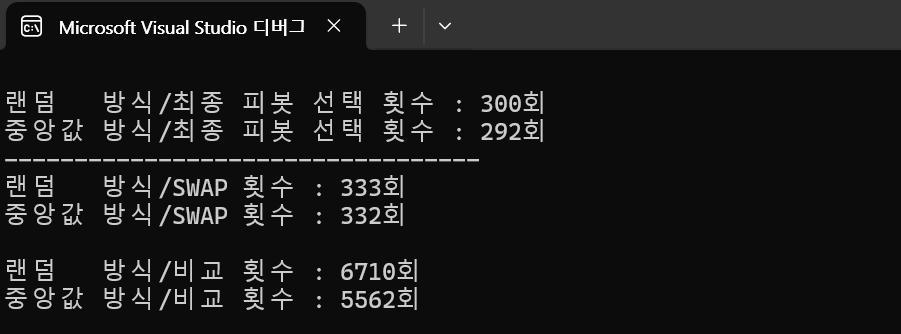
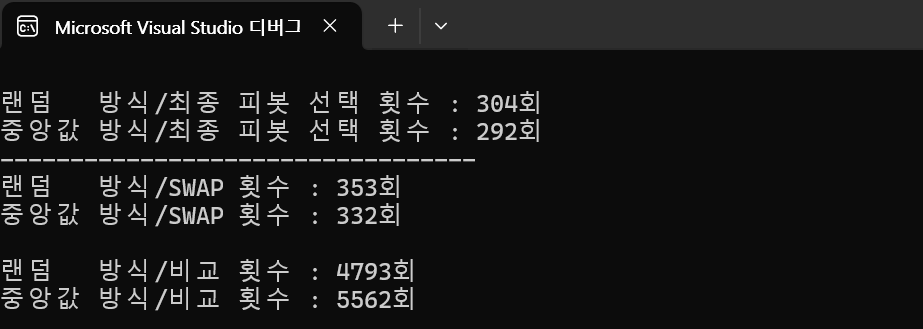
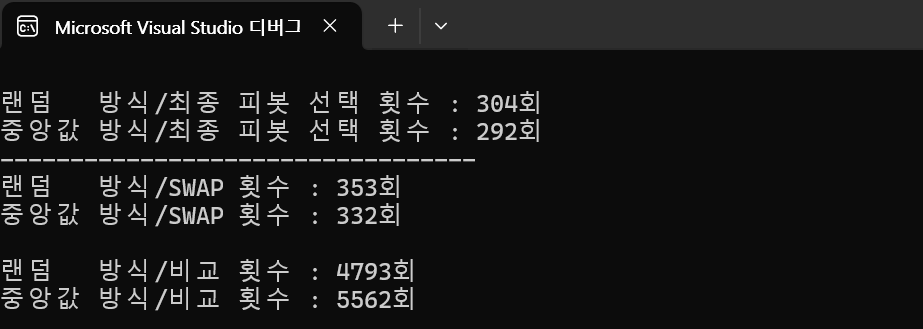
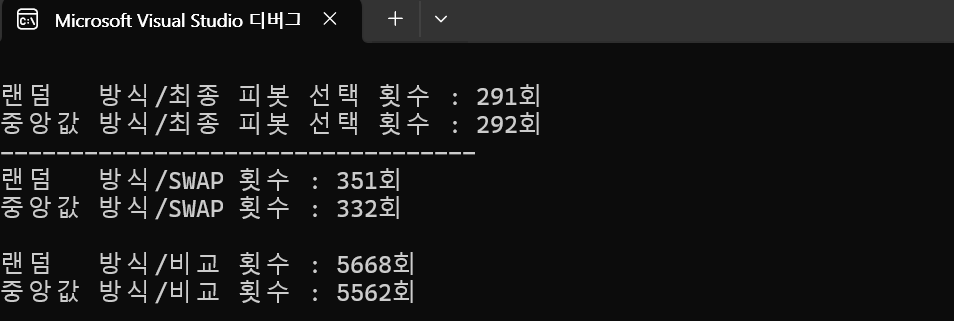
**정렬 후 데이터 - 3 part positioning**



* 제대로 잘 정렬되었음

1. **고찰**

**3-1. 2번문제 추가 분석 : SWQP 횟수, 비교 횟수 출력**



* 대체로 중앙값이 효율적이지만 큰 차이는 나지않는다
* 출력된 값을 보고 대략적으로 분석하였음
* 피봇 선정 랜덤
  + 랜덤 방식 SWAP 횟수 : 340회 전후 랜덤한 값
  + 랜덤 방식 비교 횟수 : 약5500회 전후 랜덤한 값
* 피봇 선정 중앙값
  + 중앙값방식 SWAP 횟수 : 332회
  + 중앙값 방식 비교 횟수 : 4888회 || 5562회
    - 기온 데이터에 중복된 값이 있어서 중앙값 비교횟수는 4888회 또는 5562회 둘 중 하나로 출력되는 것같다

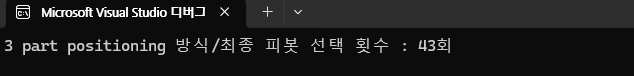
===> 랜덤 방식은 가끔씩 효율이 완전히 나빠질때가 있음

===> 중앙값 방식은 랜덤 방식보다 효율이 약간 좋음

===> 피봇 선택 횟수가 적어도 비교횟수가 많아질 수 있음 (∵bad 분할)

**3-2. 3번 문제 : 3 part positioning는 매우 효율적**

* 3 part positioning의 경우 최종 피벗 선택 횟수가 매우 줄어들었다
* Quick sort안에 중복되는 데이터 처리 문제를 해결!
  + 기온 데이터는 중복되는 값이 많은데 mid변수와 조건문을 통하여 넘어가는 방식을 통해 효율성을 높인 것 같음

****