Algorithms
Spring 2024

HW2
Report Document

Student:
TSOI VADIM
UID: 2022203502
Department of Software Engineering
Kwangwoon University

May 12, 2024

1 Method 1

In the divide_array function of the first method, I search for the greatest difference between adjacent numbers in the list k-1 times, then I slice the list up to that difference and repeat the process with the remaining part of the list. At the end of the process, I add the remaining elements. In the find_max_diff function, I find the difference between the minimum and maximum values in the intervals k and k+1.

The algorithm's speed is acceptable: the time complexity is O(n log n), necessitated by sorting. Without sorting, the complexity would be O(n), and the space complexity is O(n).

However, the problems with this algorithm are not limited to the uneven distribution of students. The main issue is that, in real life, the most significant differences are often observed between students who did not attend classes or study at all and received very low grades (0-10), and average students with grades around 50-60. In such cases, the algorithm will divide these students into two groups in the first pass and then continue to divide within the group of low-scoring students.

```python
# UID: 2022203502 최바딤
debug: bool = False


def find_max_diff(arr: list[list], k: int) -> int:
    # Time complexity: O(n) min, max functions
    # Space complexity: O(1)
    if not arr:
        return

    max_sum: int = 0

    for i in range(0, k - 1):
        min_score = min(arr[i], key=lambda x: x[1])
        max_score = max(arr[i + 1], key=lambda x: x[1])

        if debug:
            print(f"{min_score=}, {max_score=}")

        sum_diff = min_score[1] - max_score[1]

        if debug:
            print(f"{sum_diff=}\n")

        max_sum += sum_diff

    return max_sum


def divide_array(arr: list, k: int) -> list[list]:
    # Time complexity: O(nlogn)
    # Space complexity: O(n)
    arr.sort(reverse=True, key=lambda x: x[1])

    if debug:
        print(list)

    final_arr: list = []
    temp_arr: list = []
    max_diff: int = 0
    max_diff_index: int = -1

    for i in range(k - 1):
        temp_arr = arr[max_diff_index + 1 :]
        arr_len = len(temp_arr)

        for i in range(0, arr_len - 1):
            diff = temp_arr[i][1] - temp_arr[i + 1][1]

            if diff > max_diff:
                max_diff = diff
                max_diff_index = i

        max_diff = 0
        final_arr.append(temp_arr[: max_diff_index + 1])

    final_arr.append(temp_arr[max_diff_index + 1 :])

    if debug:
        print("\n")
        for i in range(k):
            print(f"{final_arr[i]}")
        print("\n")

    return final_arr


def main():
    k = int(input("Enter k: "))
    arr = input("Enter array: ").split()
    arr = [(i, int(j)) for i, j in enumerate(arr, 1)]

    if debug:
        print(arr)

    final_array = divide_array(arr, k)
    max_sum = str(find_max_diff(final_array, k))

    print(f"Maximum sum of differences: {max_sum}")

    if debug:
        print(final_array)

    with open("Partition1.txt", "w") as f:
        for arr in final_array:
            f.write(" ".join(f"{i}({j})" for i, j in arr) + "\n")


if __name__ == "__main__":
    main()
```

method1.py

2 Method 2

In the second method, I start with an initially sorted list, which I divide into k deques. For each group, I measure the variance. Then, in a loop, I move an element from one group to another, starting from the left to the right, if it reduces the variance; if the transfer does not reduce the variance, I end the loop. After that, I repeat the same process but from right to left. However, I continue the loop if the variance in one of the groups equals zero, as it was in test 1, even if the variance increases, it may improve if the variance in one of the lists is zero. I also end the loop if the length of one of the deques becomes less than or equal to one, or if the variance in both groups becomes zero.

Regarding additional restrictions on the number of students, according to which the total number of students in groups 1 and 2 cannot exceed 30% of n, and in groups 1, 2, 3, and 4, no more than 70% of n, I will need to add the following conditions to exit the loops:

- For the first condition:

```
if (i == 1 or i == 2) and (len(group[i]) * 100 / n > 30):
break
```

- For the second condition:

```
if ((i == 1 or i == 2 or i == 3 or i == 4) and (len(group[1])
+ len(group[2]) + len(group[3]) + len(group[4]) * 100 / n > 70)):
break
```

I believe this method is significantly better as it divides the students into more or less equal groups and doesn't have the issues that were present in the first method.

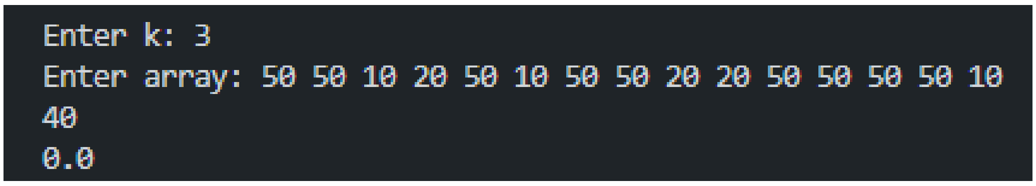
```python
# UID: 2022203502 최바딤
from collections import deque

debug = False


def count_variance(arr: list[tuple]) -> float:
    # Time complexity: O(n)
    # Space complexity: O(n)
    if not arr:
        return

    data = [i[1] for i in arr]

    mean = sum(data) / len(data)
    squared_diff = [(x - mean) ** 2 for x in data]
    sum_squared_diff = sum(squared_diff)
    variance = sum_squared_diff / len(data)

    return variance


def sort_and_divide_list_by_k_groups(arr: list, k: int) -> list:
    # Time complexity: O(nlogn)
    # Space complexity: O(n)
    sorted_scores = sorted(arr, reverse=True, key=lambda x: x[1])

    group_size = len(sorted_scores) // k
    groups = [
        deque(sorted_scores[i * group_size : (i + 1) * group_size])
        for i in range(k - 1)
    ]
    groups.append(deque(sorted_scores[(k - 1) * group_size :]))

    return groups


def divide_students(scores, k):
    # Time complexity: O(n^2)
    # Space complexity: O(n)
    groups = sort_and_divide_list_by_k_groups(scores, k)
    before_variance = [count_variance(i) for i in groups]
    new_variance = [i for i in before_variance]

    for i in range(k - 1):
        # From right to left
        while True:
            before_var_sum = before_variance[i] + before_variance[i + 1]

            # If in both groups variance is 0, break
            if new_variance[i] == 0 and new_variance[i + 1] == 0:
                break

            # Pop the first element from second group and insert to end of first group
            # If length of second group is greater than 1
            if len(groups[i + 1]) > 1:
                val = groups[i + 1].popleft()
                groups[i].append(val)
            else:
                break

            # Calculate new variance
            new_variance[i] = count_variance(groups[i])
            new_variance[i + 1] = count_variance(groups[i + 1])
            new_var_sum = new_variance[i] + new_variance[i + 1]

            if new_var_sum < before_var_sum or new_variance[i] == 0:
                before_variance[i] = count_variance(groups[i])
                before_variance[i + 1] = count_variance(groups[i + 1])

            else:
                val = groups[i].pop()
                groups[i + 1].appendleft(val)
                break

        # From left to right
        while True:
            before_var_sum = before_variance[i] + before_variance[i + 1]

            if new_variance[i] == 0 and new_variance[i + 1] == 0:
                break

            # Pop the last element from first group and insert to front of second group
            # If length of second group is greater than 1
            if len(groups[i]) > 1:
                val = groups[i].pop()
                groups[i + 1].appendleft(val)
            else:
                break

            # Calculate new variance
            new_variance[i] = count_variance(groups[i])
            new_variance[i + 1] = count_variance(groups[i + 1])
            new_var_sum = new_variance[i] + new_variance[i + 1]

            if new_var_sum < before_var_sum or new_variance[i + 1] == 0:
                before_variance[i] = count_variance(groups[i])
                before_variance[i + 1] = count_variance(groups[i + 1])

            else:
                val = groups[i + 1].popleft()
                groups[i].append(val)
                break

    return groups, round(sum(before_variance), 3)


def main():
    k = int(input("Enter k: "))
    arr = input("Enter array: ").split()
    arr = [(i, int(j)) for i, j in enumerate(arr, 1)]

    if debug:
        print(arr)

    res = divide_students(arr, k)

    print(f"Maximum sum of differences: {res[1]}")

    if debug:
        print(res[0])

    with open("Partition2.txt", "w") as f:
        for group in res[0]:
            f.write(" ".join(f"{i}({j})" for i, j in group) + "\n")


if __name__ == "__main__":
    main()
```

method2.py

3 Main fail and Outputs

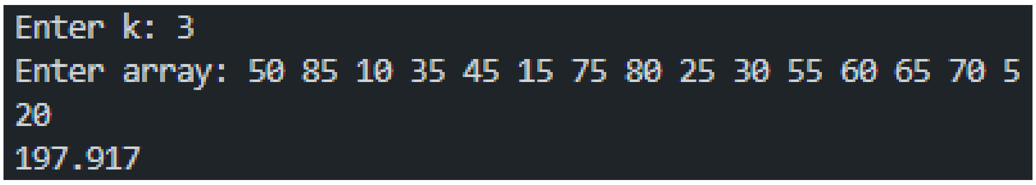
```python
# UID: 2022203502 최바딤
from method1 import divide_array, find_max_diff
from method2 import divide_students


k = int(input("Enter k: "))
arr = input("Enter array: ").split()
arr = [(i, int(j)) for i, j in enumerate(arr, 1)]

final_array = divide_array(arr, k)
max_sum = str(find_max_diff(final_array, k))
print(max_sum)

with open("Partition1.txt", "w") as f:
    for index in final_array:
        f.write(" ".join(f"{i}({j})" for i, j in index) + "\n")

res = divide_students(arr, k)
print(res[1])

with open("Partition2.txt", "w") as f:
    for index in res[0]:
        f.write(" ".join(f"{i}({j})" for i, j in index) + "\n")
```

main.py



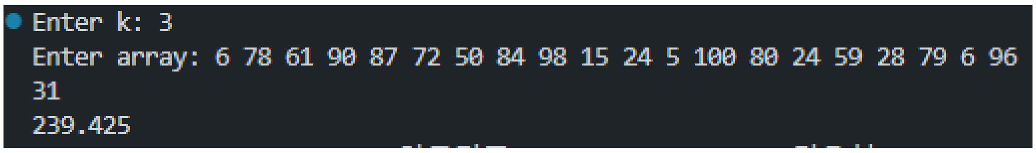
```
Enter k: 3
Enter array: 50 50 10 20 50 10 50 50 20 20 50 50 50 50 10
40
0.0
```

Figure 1: Test 1



```
Enter k: 3
Enter array: 50 85 10 35 45 15 75 80 25 30 55 60 65 70 5
20
197.917
```

Figure 2: Test 2



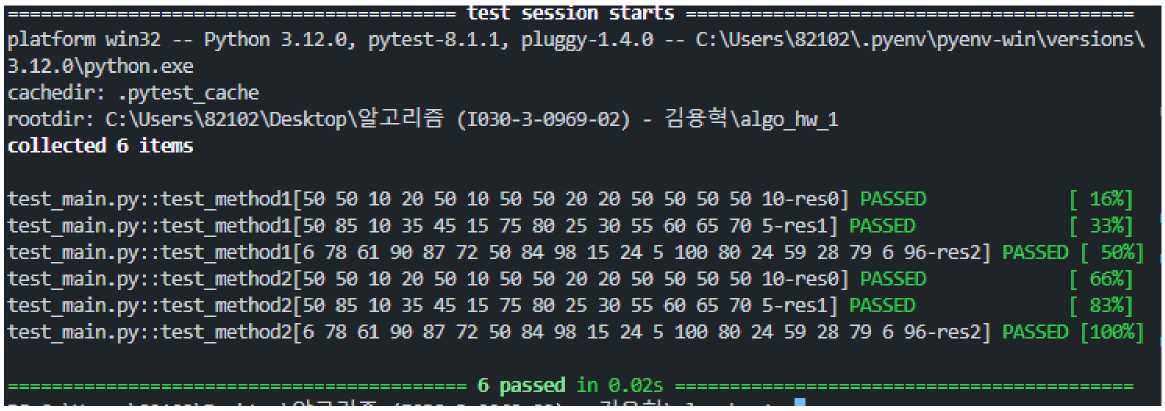
```
Enter k: 3
Enter array: 6 78 61 90 87 72 50 84 98 15 24 5 100 80 24 59 28 79 6 96
31
239.425
```

Figure 3: Test 3

4 Tests

```python
# UID: 2022203502 최바딤
import pytest
from collections import deque

from method1 import divide_array
from method2 import divide_students


@pytest.mark.parametrize(
    "arr, res",
    [
        (
            "50 50 10 20 50 10 50 50 20 20 50 50 50 50 10",
            [
                [
                    (1, 50),
                    (2, 50),
                    (5, 50),
                    (7, 50),
                    (8, 50),
                    (11, 50),
                    (12, 50),
                    (13, 50),
                    (14, 50),
                ],
                [(4, 20), (9, 20), (10, 20)],
                [(3, 10), (6, 10), (15, 10)],
            ],
        ),
        (
            "50 85 10 35 45 15 75 80 25 30 55 60 65 70 5",
            [
                [
                    (2, 85),
                    (8, 80),
                    (7, 75),
                    (14, 70),
                    (13, 65),
                    (12, 60),
                    (11, 55),
                    (1, 50),
                    (5, 45),
                ],
                [(4, 35), (10, 30), (9, 25)],
                [(6, 15), (3, 10), (15, 5)],
            ],
        ),
        (
            "6 78 61 90 87 72 50 84 98 15 24 5 100 80 24 59 28 79 6 96",
            [
                [
                    (13, 100),
                    (9, 98),
                    (20, 96),
                    (4, 90),
                    (5, 87),
                    (8, 84),
                    (14, 80),
                    (18, 79),
                    (2, 78),
                    (6, 72),
                    (3, 61),
                    (16, 59),
                    (7, 50),
                ],
                [(17, 28), (11, 24), (15, 24)],
                [(10, 15), (1, 6), (19, 6), (12, 5)],
            ],
        ),
    ],
)
def test_method1(arr, res):
    arr = arr.split()
    arr = [(i, int(j)) for i, j in enumerate(arr, 1)]

    assert divide_array(arr, 3) == res


@pytest.mark.parametrize(
    "arr, res",
    [
        (
            "50 50 10 20 50 10 50 50 20 20 50 50 50 50 10",
            [
                [
                    (1, 50),
                    (2, 50),
                    (5, 50),
                    (7, 50),
                    (8, 50),
                    (11, 50),
                    (12, 50),
                    (13, 50),
                    (14, 50),
                ],
                [(4, 20), (9, 20), (10, 20)],
                [(3, 10), (6, 10), (15, 10)],
            ],
        ),
        (
            "50 85 10 35 45 15 75 80 25 30 55 60 65 70 5",
            [
                [(2, 85), (8, 80), (7, 75), (14, 70), (13, 65)],
                [(12, 60), (11, 55), (1, 50), (5, 45)],
                [(4, 35), (10, 30), (9, 25), (6, 15), (3, 10), (15, 5)],
            ],
        ),
        (
            "6 78 61 90 87 72 50 84 98 15 24 5 100 80 24 59 28 79 6 96",
            [
                [(13, 100), (9, 98), (20, 96), (4, 90), (5, 87)],
                [
                    (8, 84),
                    (14, 80),
                    (18, 79),
                    (2, 78),
                    (6, 72),
                    (3, 61),
                    (16, 59),
                    (7, 50),
                ],
                [(17, 28), (11, 24), (15, 24), (10, 15), (1, 6), (19, 6), (12, 5)],
            ],
        ),
    ],
)
def test_method2(arr, res):
    arr = arr.split()
    arr = [(i, int(j)) for i, j in enumerate(arr, 1)]
    deque = [list(d) for d in divide_students(arr, 3)[0]]
    assert deque == res
```

test_main.py



```
=========================================== test session starts ===========================================
platform win32 -- Python 3.12.0, pytest-8.1.1, pluggy-1.4.0 -- C:\Users\82102\.pyenv\pyenv-win\versions\
3.12.0\python.exe
cachedir: .pytest_cache
rootdir: C:\Users\82102\Desktop\알고리즘 (I030-3-0969-02) - 김용혁\algo_hw_1
collected 6 items

test_main.py::test_method1[50 50 10 20 50 10 50 50 20 20 50 50 50 50 10-res0] PASSED             [ 16%]
test_main.py::test_method1[50 85 10 35 45 15 75 80 25 30 55 60 65 70 5-res1] PASSED              [ 33%]
test_main.py::test_method1[6 78 61 90 87 72 50 84 98 15 24 5 100 80 24 59 28 79 6 96-res2] PASSED [ 50%]
test_main.py::test_method2[50 50 10 20 50 10 50 50 20 20 50 50 50 50 10-res0] PASSED             [ 66%]
test_main.py::test_method2[50 85 10 35 45 15 75 80 25 30 55 60 65 70 5-res1] PASSED              [ 83%]
test_main.py::test_method2[6 78 61 90 87 72 50 84 98 15 24 5 100 80 24 59 28 79 6 96-res2] PASSED [100%]

============================================ 6 passed in 0.02s ============================================
```

Figure 4: Test 3