컴퓨터알고리즘과실습 실습4

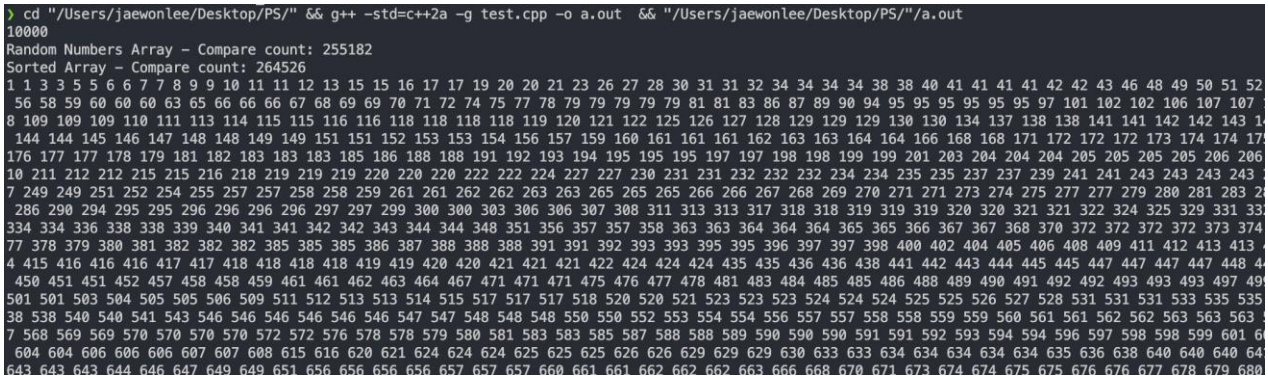
2020112046 이재원

1번

```
#include <iostream>
using namespace std;


class PQ {
   private:
    int N;
    int i;

   public:
    int *a;
    int heapCompare;

    PQ(int max) {
        a = new int[max];
        N = 0;
        i = 0;
        heapCompare = 0;
    }
    ~PQ() {
        delete a;
    }

    void insertData(int v) {
        a[i++] = v;
    }

    int get(int i) {
        return a[i];
    }

    // insert a new element into the priority queue - O(logN)
    // !---------------
    void MakeHeap(int a[], int root, int LastNode) {
        int child;
        int temp;
        temp = a[root];
        child = 2 * root;
        while (child <= LastNode) {
            heapCompare++;
            if ((child < LastNode) && (a[child] < a[child + 1])) {
                child++;
            }
            heapCompare++;
            if (temp >= a[child]) {
                break;
```

```
            } else {
                a[child / 2] = a[child];
                child *= 2;
            }
        }
        a[child / 2] = temp;
    }
};

void swap(int &a, int &b) {
    int temp = a;
    a = b;
    b = temp;
}

void heapSort(int a[], int N, PQ &pq) {
    for (int i = N / 2 - 1; i >= 0; i--)
        pq.MakeHeap(a, i, N - 1);
    for (int i = N - 1; i >= 1; i--) {
        swap(a[0], a[i]);
        pq.MakeHeap(a, 0, i - 1);
    }
}

int main() {
    int N;
    cin >> N;
    PQ pq(N);
    for (int i = 0; i < N; i++) {
        pq.insertData(rand() % 10000 + 1);
    }
    heapSort(pq.a, N, pq);
    cout << "Random Numbers Array - Compare count: " << pq.heapCompare << endl;

    //!----------------
    //최악의 경우 : 이미 정렬된 배열을 입력받을 경우 -> 최대 힙을 사용하기 때문에 비교 횟수가 많아짐.
    int K = 10000;
    PQ pq2(K);
    for (int i = 0; i < K; i++) {
        pq2.insertData(i + 1);
    }
    heapSort(pq2.a, K, pq2);
    cout << "Sorted Array - Compare count: " << pq2.heapCompare << endl;
    //!----------------
    for (int i = 0; i < N; i++) {
        cout << pq.get(i) << " ";
    }

    return 0;
}
```

실행 결과



heapSort가 최악의 시간복잡도를 겪을 때는, 이미 정렬된 값을 넣었을 때 이므로, 정렬하여 넣어 주었다.

그러나 heapSort는 최선, 최고, 최악 모두 time complexity가 O(nlogn)이다.

비교 횟수가 그리 크지 않은 차이를 보이는 것을 알 수 있다.

2번

```
#include <iostream>
using namespace std;

int compareCount = 0;

void swap(int a[], int i, int j) {
    int temp = a[i];
    a[i] = a[j];
    a[j] = temp;
}

int partition(int a[], int l, int r) {
    int i, j;
    int v;

    if (r > l) {
        v = a[l];
        i = l;
        j = r + 1;
        while (1) {
            while (a[++i] < v)
                if (i == r) break;
            while (a[--j] > v)
                if (j == l) break;
            compareCount++;
```

```
            if (i >= j) break;
            swap(a, i, j);
        }
        swap(a, l, j);
    }
    return j;
}

void select(int a[], int l, int r, int k) {
    int j;
    if (r > l) {
        j = partition(a, l, r);
        if (j > l + k - 1) select(a, l, j - 1, k);
        if (j < l + k - 1) select(a, j + 1, r, k - (j - l + 1));
    }
}

int main() {
    int N;
    cin >> N;
    int a[N], b[N], c[N];
    for (int i = 0; i < N; i++) {
        a[i] = rand() % 10000 + 1;
        b[i] = a[i];
        c[i] = a[i];
    }

    select(a, 0, N - 1, 1);
    cout << "Minimum: " << a[0] << ", Compare: " << compareCount << endl;
    compareCount = 0;
    select(b, 0, N - 1, N);
    cout << "Maximum: " << b[N - 1] << ", Compare: " << compareCount << endl;
    compareCount = 0;
    select(c, 0, N - 1, N / 2);
    cout << "Middle: " << c[N / 2] << ", Compare: " << compareCount << endl;

    // print a array
    for (int i = 0; i < N; i++) {
        cout << a[i] << " ";
    }
}
```

실행 결과

```
❯ cd "/Users/jaewonlee/Desktop/PS/" && g++ -std=c++2a -g test.cpp -o
10000
Minimum: 1, Compare: 3145
Maximum: 10000, Compare: 3184
Middle: 4927, Compare: 6541
1 1 7 16 20 13 7 3 6 8 17 19 15 15 5 5 11 10 9 17 9 11 3 12 6 20 130
 118 95 122 78 118 66 95 126 109 58 86 97 109 146 134 147 110 60 79
9 130 60 90 52 116 102 107 42 51 138 27 21 107 138 129 50 30 142 144
48 102 70 145 38 141 148 243 154 257 206 183 210 168 249 157 243 160
178 237 176 247 205 246 201 232 174 162 172 216 183 171 188 159 156
66 173 182 198 179 172 204 203 210 220 186 257 175 234 194 185 258 2
```

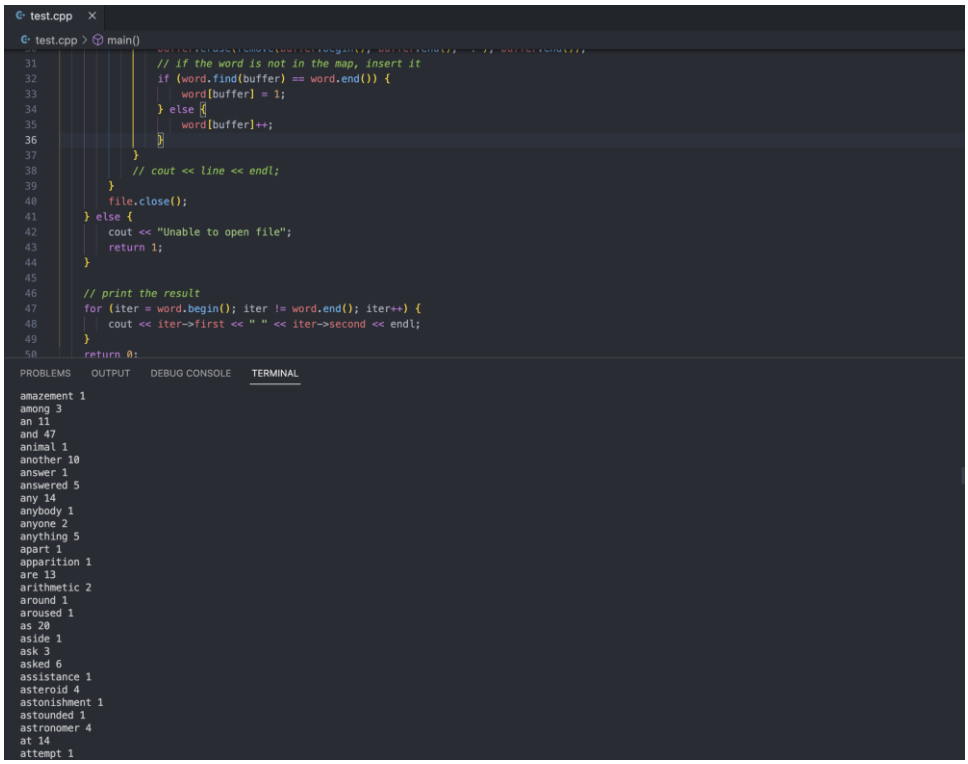
Minimum, Maximum값을 찾을 때는 partition한 후 한 쪽만 보면 되기 때문에, Middle값을 찾을 때보다 2배정도 비교 횟수가 작아지는 것을 볼 수 있으며, Minimum, Maximum값을 찾는 비교 횟수는 비슷한 것을 볼 수 있다.

3번

```
#include <algorithm>
#include <fstream>
#include <iostream>
#include <map>
#include <sstream>
#include <string>
#include <vector>

using namespace std;
map<string, int> word;
map<string, int>::iterator iter;

int main() {
    string line;
    string buffer;

    ifstream file("datafile.txt");
    if (file.is_open()) {
        while (getline(file, line)) {
            // split the line into words with blank
            istringstream iss(line);
            while (getline(iss, buffer, ' ')) {
                // remove , and " and . and : and ; and ? and ! from the buffer
                buffer.erase(remove(buffer.begin(), buffer.end(), ','), buffer.end());
                buffer.erase(remove(buffer.begin(), buffer.end(), '"'), buffer.end());
                buffer.erase(remove(buffer.begin(), buffer.end(), '.'), buffer.end());
                buffer.erase(remove(buffer.begin(), buffer.end(), ':'), buffer.end());
                buffer.erase(remove(buffer.begin(), buffer.end(), ';'), buffer.end());
                buffer.erase(remove(buffer.begin(), buffer.end(), '?'), buffer.end());
                buffer.erase(remove(buffer.begin(), buffer.end(), '!'), buffer.end());
                // if the word is not in the map, insert it
                if (word.find(buffer) == word.end()) {
                    word[buffer] = 1;
                } else {
                    word[buffer]++;
                }
            }
            // cout << line << endl;
        }
        file.close();
    } else {
        cout << "Unable to open file";
        return 1;
    }
```

```
    // print the result
    for (iter = word.begin(); iter != word.end(); iter++) {
        cout << iter->first << " " << iter->second << endl;
    }
    return 0;
}
```

실행 결과



어린왕자(Little Prince) 의 일부분을 발췌하여 실행해봤음.

소감

슬슬 재밌는 것 같습니다.. ^ㅁ^