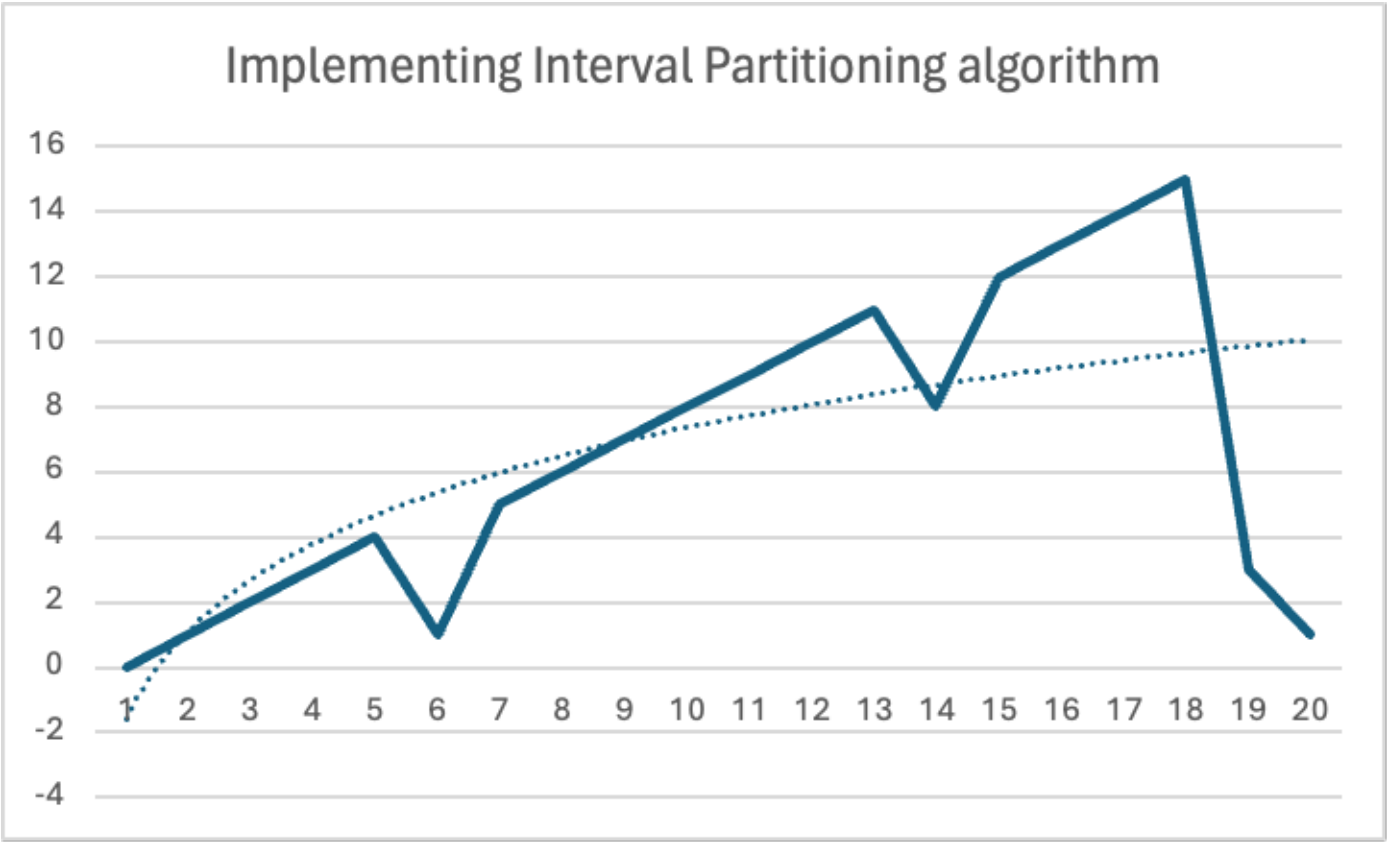
演算法-(Homework 1) Implementing Interval Partitioning algorithm

解釋工作原理：

1. 定義了一個 `Lecture` 結構來表示每個講座的開始和結束時間。
2. 按照講座的開始時間對講座進行排序。
3. 維護一個向量 `endTimes` 來跟踪每個教室中當前安排的講座的結束時間。
4. 遍歷每個講座，對於每個講座，嘗試找到一個可以安排它而不與其他講座重疊的教室。
5. 如果找不到這樣的教室，就分配一個新的教室。
6. 需要的最少教室數量是 `endTimes` 向量的長度。

時間複雜度

對講座進行排序的時間複雜度為 O(nlogn)。原因為，遍歷每個講座一次，對於每個講座，可能需要遍歷先前安排的講座的結束時間，最壞情況下為 O(n) 時間。因此，算法的整體時間複雜度為 O(nlogn + n)，主要由排序步驟支配。因此，整體時間複雜度為 O(nlogn)。



Source code

```
#include <iostream>
#include <vector>
```

```cpp
#include <algorithm>
#include <map>

using namespace std;

struct Lecture {
    int start;
    int end;
};

bool compareLectures(const Lecture& a, const Lecture& b) {
    return a.start < b.start;
}

map<int, vector<Lecture>> minClassrooms(vector<Lecture>& lectures) {
    sort(lectures.begin(), lectures.end(), compareLectures);
    map<int, vector<Lecture>> classrooms; // Map to store lectures assigned to each classroom
    vector<int> endTimes;

    for (const auto& lecture : lectures) {
        bool scheduled = false;
        for (int i = 0; i < endTimes.size(); ++i) {
            if (endTimes[i] <= lecture.start) {
                endTimes[i] = lecture.end;
                classrooms[i].push_back(lecture); // Assign the lecture to this classroom
                scheduled = true;
                break;
            }
        }
        if (!scheduled) {
            endTimes.push_back(lecture.end);
            classrooms[endTimes.size() - 1].push_back(lecture); // Assign the lecture to a new classroom
        }
    }

    return classrooms;
}

int main() {
    vector<Lecture> lectures = {
        {900, 1000}, {1100, 1600}, {1230, 1330}, {1030, 1600}, {930, 1400},
        {1030, 1300}, {1130, 1530}, {1230, 1630}, {1100, 1200}, {930, 1400},
        {1130, 1700}, {930, 1530}, {1200, 1730}, {1130, 1400}, {1230, 1500},
        {1100, 1600}, {1200, 1700}, {1530, 1630}, {1630, 1730}, {900, 1600}
    };

    map<int, vector<Lecture>> classrooms = minClassrooms(lectures);
    cout << "所需的最少教室數量為：" << classrooms.size() << endl;

    // Output the lecture schedule for each classroom
```

```
    for (const auto& [classroom, assignedLectures] : classrooms) {
        cout << "教室 " << classroom + 1 << "：";
        for (const auto& lecture : assignedLectures) {
            cout << " Lecture(" << lecture.start << "-" << lecture.end << ")";
        }
        cout << endl;
    }

    return 0;
}
```