챕터 7. Greedy Aproach (탐욕 알고리즘) - (4) 회의실 배정 문제와 탐욕법

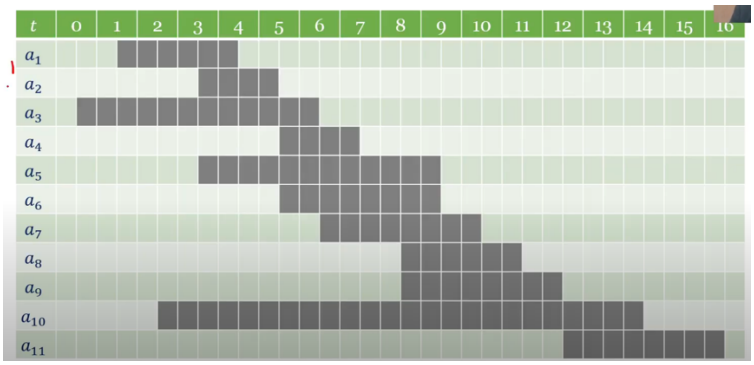
BOJ 1931. 회의실 배정 (교재 4.3 스케줄 짜기와 비슷함)

- 한 개의 회의실이 있는데
    - n개의 회의에 대한 회의실 사용표를 만들려고 한다.
        - Si: 회의 시작 시간, fi: 회의 종료 시간
    - 최대한 많은 회의를 할 수 있도록 회의 스케줄을 짜보자.

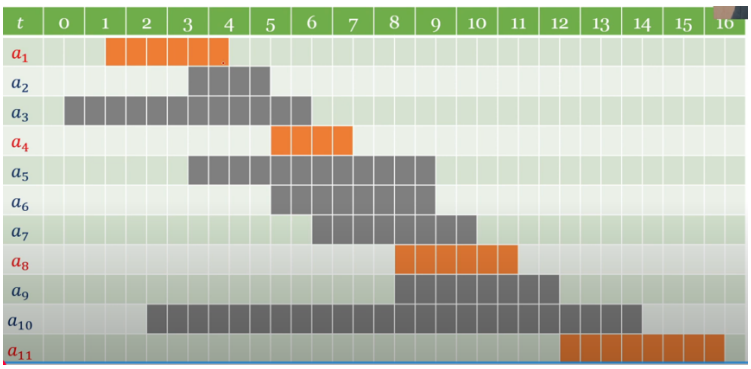
S = {a1, a2, ---, an}

i	1	2	3	4	5	6	7	8	9	10	11
s_i	1	3	0	5	3	5	6	8	8	2	12
f_i	4	5	6	7	9	9	10	11	12	14	16

S = {a1, a2, ---, a11}



S' = {a1, a4, a8, a12} -> max = 4



단순 무식한 방법: Brute-Force

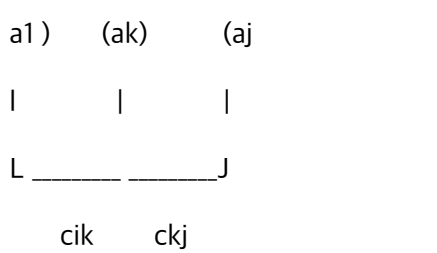
- 회의 시간이 겹치면 회의를 개최할 수 없다.
    - S의 모든 부분 집합에서 서로 겹치지 않는 부분 집합의 최대값 찾기

- 조합 최적화 문제:
    - 최적값은 집합 크기의 최대값
    - 최적해는 해당하는 부분 집합

- S의 부분 집합의 개수: 2^n
    - 지수 시간 복잡도를 가짐: exponential

재귀적 관계 분석하기:

- Sij: 회의 i가 종료한 후 시작하고, 회의 j가 시작하기 전 종료하는 회의의 값
- c(i, j): Sij의 최적값 (회의 i에서 j까지 개최할 수 있는 최대 회의 크기의 값)
- 재귀적 관계: 임의의 k에 대해서 다음 관계가 성립함
    - c(i, j) = c(i, k) + c(k, j) + 1



- 재귀식:
    - $S_{ij} = \emptyset$이면 $c(i,j) = 0$
    - $S_{ij} \neq \emptyset$이면 $c(i,j) = \max_{a_k \in S_{ij}} \{c(i,k) + c(k,j) + 1\}$

분할정복법과 동적계획법

- 분할해서 정복하기
    - 중복 부분 문제가 발생: overlapping subproblems
    - 시간 복잡도: 지수 시간
- 동적 계획법
    - 최적 부분 구조가 성립: optimal substructure

- 시간 복잡도: O(n^3)

- 더 효율적인 방법은 없을까?

  - Greedy Approach: O(n)

탐욕적 부분 최적구조: greedy optimal substructure

- 모든 부분 문제에 대한 최적해를 찾지 않고,

  - 최적해에 넣을 수 있는 한 개의 회의를 선택한다면?

- 전처리: 회의 종료시간의 오름차순으로 회의를 정렬

  - f1 <= f2 <= --- <= fn

- 회의가 서로 겹치지 않는다는 것:

  - 두 회의 ai, aj에 대해서는 fi <= fj일 때 f1 <= sj이면 서로 겹치지 않음.

탐욕법: Greedy Approach

- 전처리: 회의 종료 시간의 오름차순으로 회의를 정렬

  - $S= \{a_1, a_2, ---, a_n\}$, $f_1 <= f_2 <= --- <= f_n$

- 초기화: $F = \Phi$, $f_F = 0$

- 탐욕적 선택: greedy selection

  - S에서 가장 먼저 끝나는 회의 a_k를 선택(k = 1, 2, ---, n)

- 실행 가능성: feasible?

  - a_k를 F에 추가할 수 있으면 추가: $f_F <= s[k]$

- 최적해 판단: solved? : S의 모든 원소를 탐색했으면 종료

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

int main() {
    int n;
    cin >> n;

    // 회의 시작 시간과 종료 시간 쌍을 저장
    vector<pair<int, int> > meetings(n);

    for (int i = 0; i < n; i++) {
        cin >> meetings[i].first >> meetings[i].second; // 시작, 종료 시간 입력
    }

    vector<pair<int, int> > selected;
    int last_end = 0;

    for (int i = 0; i < n; i++) {
        int start = meetings[i].first;
        int end = meetings[i].second;
        if (start >= last_end) {
            selected.push_back({start, end});
            last_end = end;
        }
    }

    // 출력
    cout << selected.size() << endl;
    for (int i = 0; i < selected.size(); i++) {
        cout << selected[i].first << " " << selected[i].second << endl;
    }

    return 0;
}

```

탐욕법으로 풀 수 없는 경우:

- 회의를 개최할 때 회의실 사용 비용을 받는다고 가정하면

  - 회의를 개최할 때 비용이 최대화 되는 회의 스케줄은?

i	1	2	3	4	5	6	7	8	9	10	11
s_i	1	3	0	5	3	5	6	8	8	2	12
f_i	4	5	6	7	9	9	10	11	12	14	16
c_i	1	2	8	1	2	7	3	2	9	4	3