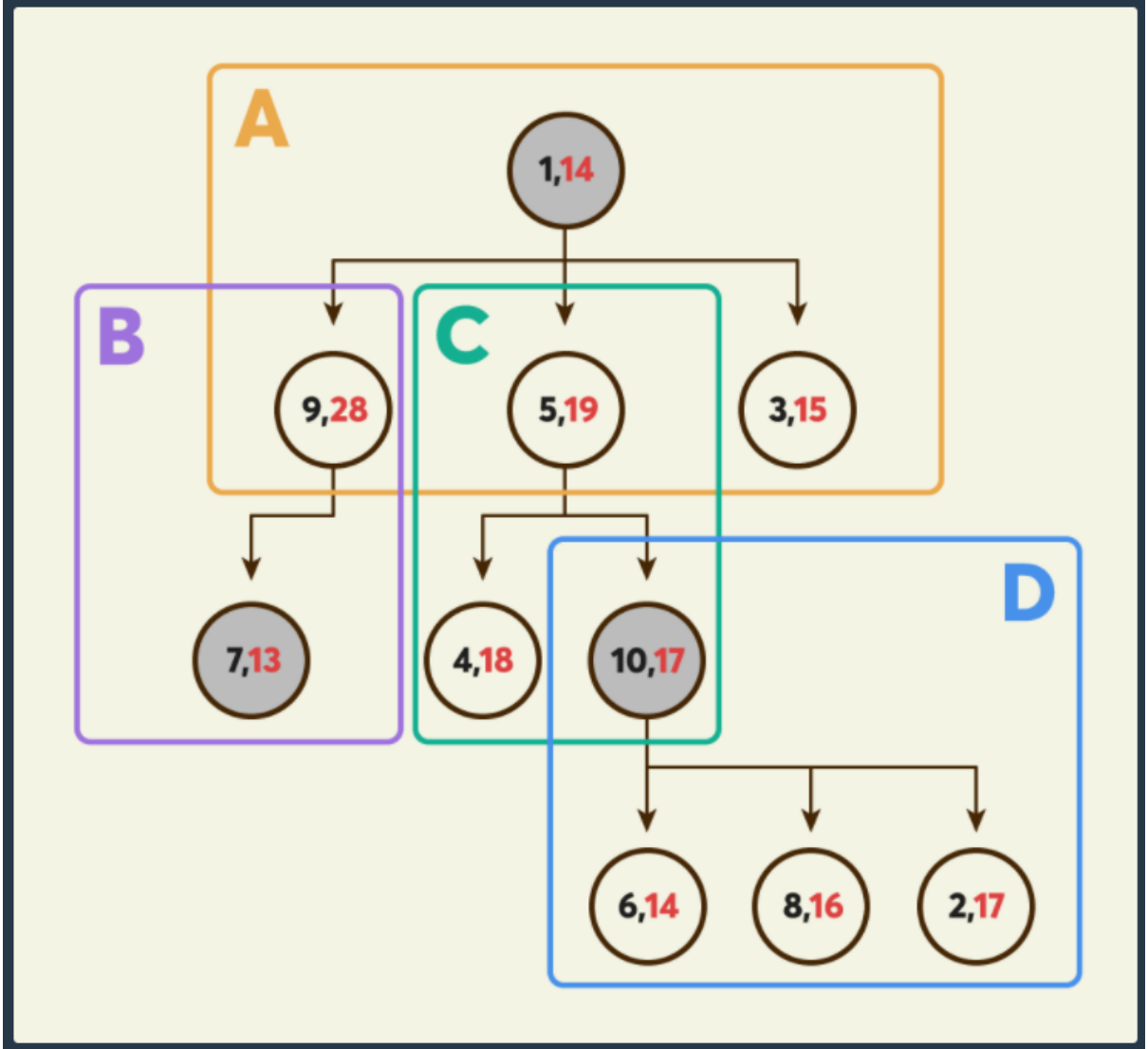
매출 하락 최소화

유통전문회사 카카오상사 의 오너인 제이지 는 새로운 사업 아이템을 구상하기 위해 전문경영인(CEO)인 프로도 에게 회사의 경영을 부탁하였습니다.
"카카오상사"는 직원들을 여러 개의 팀 단위로 조직을 구성하고 있으며 아래 그림은 CEO를 포함하여 10명의 직원과 4개의 팀으로 구성되어 있는 회사 조직도를 보여주고 있습니다.



그림의 조직도는 다음과 같이 설명할 수 있습니다.

1. 그림의 각 원들은 각각의 직원 1명을 표시하고 있으며, CEO를 포함하여 총 10명의 직원을 표시하고 있습니다.
2. 원 안에 적힌 두 개의 숫자는 직원의 정보를 담고 있습니다. 왼쪽 숫자는 `직원번호` 이며 직원을 식별할 수 있도록 1번부터 순서대로 발급되는 일련번호이며, 오른쪽 숫자는 `해당 직원의 하루평균 매출액` 을 나타냅니다. 위 그림에서 `1번` 직원은 14원을, `9번` 직원은 28원의 하루평균 매출액을 기록하고 있습니다.
3. CEO를 포함하여 모든 직원은 팀장 또는 팀원이라는 직위를 가지고 있으며 그림에서는 팀장과 팀원의 관계를 화살표로 표시하고 있습니다. 화살표가 시작되는 쪽의 직원은 팀장, 화살표를 받는 쪽의 직원은 팀원을 의미합니다.
   3-1. 직원번호 `1번` 은 회사의 CEO로 고정되어 있으며, CEO는 항상 팀장이고 팀원일 수 없어 화살표를 받는 쪽이 될 수 없습니다.
   3-2. 반면에 CEO를 제외한 나머지 모든 직원들은 다른 누군가로부터 정확히 1개의 화살표를 받게 됩니다.
   3-3. 한 직원은 최대 2개의 팀에 소속될 수 있습니다. 만약 어떤 직원이 두 개의 팀에 소속되어 있다면, 반드시 하나의 팀에서는 팀장, 나머지 팀에서는 팀원이어야 합니다. 팀장을 겸임하거나, 두 개의 팀에서 팀원이 될 수는 없습니다. 예를들어 `10번` 직원은 `D팀` 의 팀장이면서 동시에 `5번` 직원이 팀장으로 있는 `C팀` 에 속한 팀원입니다.
   3-4. `5번, 9번, 10번` 직원은 받는 쪽의 화살표와 시작하는 화살표가 모두 있으므로 팀장인 동시에 팀원입니다.
   3-5. `2번, 3번, 4번, 6번, 7번, 8번` 직원은 시작하는 화살표가 없고 받는 쪽의 화살표만 있으므로 팀장이 아니며 오직 팀원입니다.
   3-6. `1번` 직원인 CEO는 받는 쪽의 화살표가 없고 시작하는 화살표만 있으며 항상 팀원이 아닌 팀장입니다.
   3-7. 그림의 조직도에는 `A, B, C, D` 총 4개의 팀이 존재하며, 각각 `1번, 9번, 5번, 10번` 직원이 팀장 직위를 담당하게 됩니다.

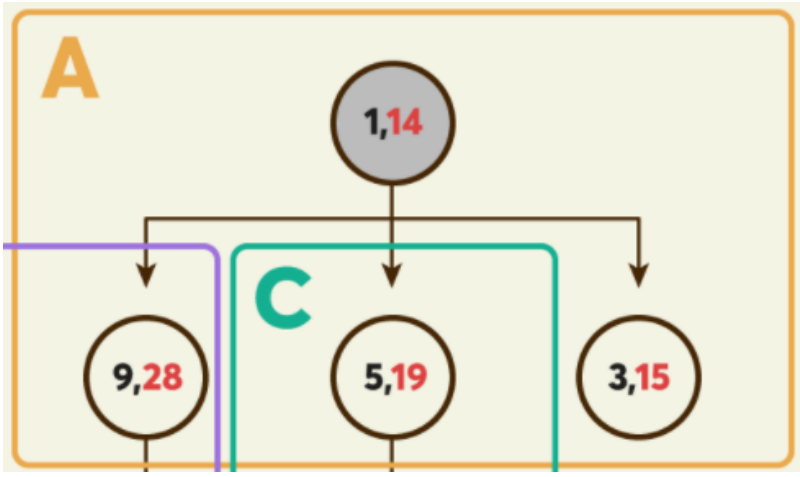
"제이지"는 자신이 구상한 새로운 사업 아이템에 대해 직원들에게 설명하고자 하루 일정으로 워크숍을 계획하고 있습니다. 단, 모든 직원을 참석시킬 수 없어 아래와 같은 기준으로 워크숍에 참석할 직원들을 선발하려고 합니다.

1. 워크숍에서 교육받은 내용은 전 직원들에게 공유되어야 하므로 `모든 팀은 최소 1명 이상` 의 직원을 워크숍에 참석시켜야 합니다.
2. 워크숍 기간 동안, 회사의 매출 손실을 최소화하는 것이 중요하므로 워크숍에 참석하는 직원들의 하루평균 매출액의 합이 최소가 되어야 합니다.

위 그림의 조직도에서 회색으로 색칠된 `1번, 7번, 10번` 직원을 워크숍에 참석시키면 모든 팀에서 최소 한 명 이상의 직원을 참석시킨 것이 되며, 해당 직원들의 하루평균 매출액의 합은 `44(14+13+17)원` 입니다. `10번 직원은 C팀과 D팀 모두에 속해 있으므로, 두 팀에서 모두 참석한 것으로 인정됩니다.`

[문제]

직원들의 하루평균 매출액 값을 담은 배열 sales, 직원들의 `팀장-팀원` 의 관계를 나타내는 2차원 배열 links가 매개변수로 주어집니다. 이때, 모든 팀에서 최소 한 명 이상 워크숍에 참석하면서, 참석하는 직원들의 하루평균 매출액의 합을 최소로 하려고 합니다. 그렇게 최소화된 매출액의 합을 구해서 return 하도록 solution 함수를 완성해 주세요.



팀당 한명 이상의 인원이 참여해야 해야함.

일반적으로 생각했을 때는, 팀원들을 포괄하고 있는 팀장이 참석하는 것이

최적해 일 수도 있지만, 반드시 최적해가 아닐 수도 있음

why?

여러팀이 얽혀있어서

어느팀에서는 팀장이지만 어느팀 안에서는 팀원 일 수 있기 때문

Tree dp를 다음과 같이 정의합니다.

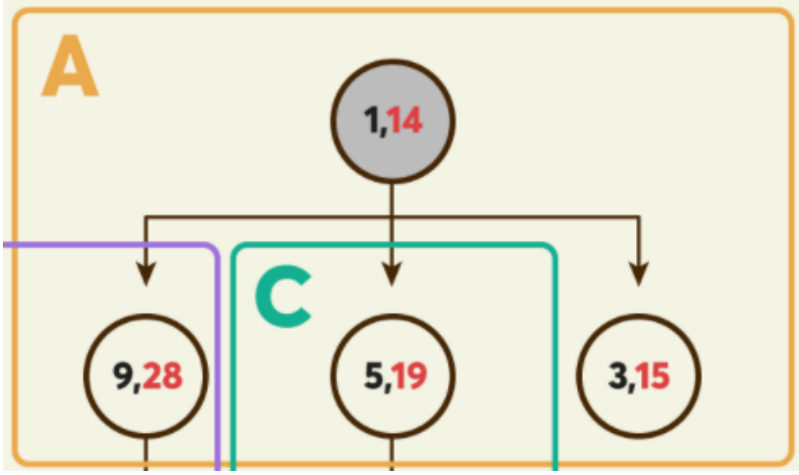
dp[i][0] i번 노드가 루트인 서브트리에서

i가 불참하는 경우의 최적해

dp[i][1] i번 노드가 루트인 서브트리에서

i가 참여하는 경우의 최적해

두 결과 중 작은 값이 이 문제의 답으로 도출됨.

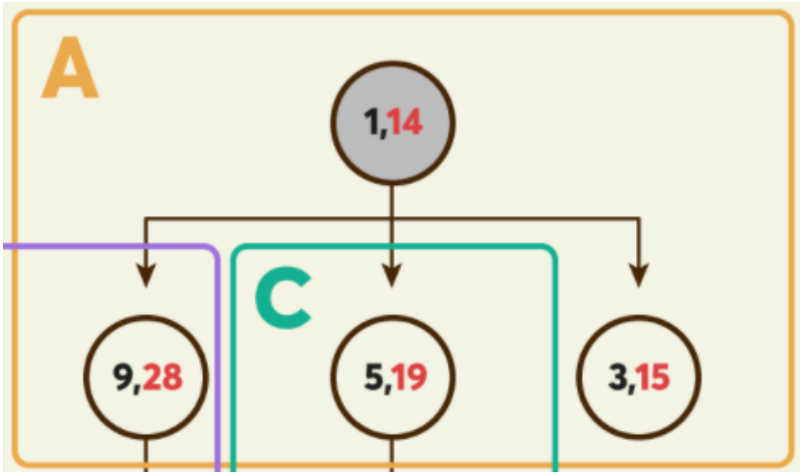


dp[i][1] 의 계산 방법.

팀장이 참여하므로 아래 팀원들은 참여하던 안하던 상관이 없음

```
dp[i][1] = min( dp[child(0)][1], dp[child(0)][0])
          +min( dp[child(1)][1], dp[child(1)][0])
          +min( dp[child(2)][1], dp[child(2)][0])
          + sales[i];
```

자식들의 최적해 + 참석하는 i의 sales 값을 더해주면 됨



dp[i][0] 의 계산 방법

min(dp[child(0)][1], dp[child(0)][0])

위 과정에서 자식들의 합을 구할 때에 만약

dp[i][1] 이 한번이라도 선택되었다는 것은

즉 (dp[child(0)][0] > dp[child(0)][1]) 의 값이 만족 한다는 것이고

과정에서 한명 이상의 자식이 참여한다는 것이므로

자식 노드의 합을 최적해로 갖게됩니다.

But

한번도 dp[i][1] 이 선택되지 않았다면,

i를 기준으로 자식노드들은 모두 불참하는 것이 이득인 노드들

i가 불참하기 때문에 한명은 강제적으로 참여시켜야 하기 때문에

참여 했을 때와 참여 하지 않았을 때의 값의 차가 최소인 자식을 선택하여

최적해를 구하면 요구하는 답을 도출 할 수 있음

코드

```
package Algo_Study_Programmers;

import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.util.*;

class Employee {

  int id;
  int cost;

  public Employee(int id, int cost) {
    this.id = id;
    this.cost = cost;
  }
}
```

```java
public class Solution_매출하락최소화 {

  static List<Employee>[] treeMap;
  static boolean[] visit;
  static int[][] dp;
  static int N;

  private static void dfs(int cur) {

    visit[cur] = true;

    if(treeMap[cur].size() == 0) {
      return;
    }

    int sum = 0, cnt = 0;

    for(Employee child : treeMap[cur]) {
      if(visit[child.id]) {
        continue;
      }

      dfs(child.id);

      if(dp[child.id][0] > dp[child.id][1]) {
        sum += dp[child.id][1];
        cnt++;
        min = Math.min(min, dp[child.id][1] - dp[child.id][0]);
      }
      else {
        sum += dp[child.id][0];
      }
    }

    dp[cur][1] += sum;

    if(cnt > 0) {
      dp[cur][0] = sum;
    }
    else {
      int min = Integer.MAX_VALUE;

      for(Employee child : treeMap[cur]) {
        min = Math.min(min, dp[child.id][1] - dp[child.id][0]);
      }
      dp[cur][0] = sum + min;
    }
  }
    public static int solution(int[] sales, int[][] links) {

        N = sales.length;
        treeMap = new ArrayList[N + 1];
        dp = new int[N + 1][2];
        visit = new boolean[N + 1];
        Employee[] employees = new Employee[N + 1];
        for(int i = 1; i <= N; i++) {

          treeMap[i] = new ArrayList<Employee>();
          dp[i][1] = sales[i - 1];
          employees[i] = new Employee(i, sales[i- 1]);
        }

        for(int[] link : links) {
          treeMap[link[0]].add(employees[link[1]]);
        }

        dfs(1);

      int answer = Math.min(dp[1][0], dp[1][1]);
        return answer;
    }
```

```
  public static void main(String[] args) {

    int[] sales = new int[] { 14, 17, 15, 18, 19, 14, 13, 16, 28, 17 };
    int[][] links = new int[][] {{10, 8}, {1, 9}, {9, 7}, {5, 4}, {1, 5}, {5, 10}, {10, 6}, {1, 3}, {10, 2}};
    System.out.println(solution(sales, links));
  }
}
```