과제8b

60211579
서호준

1번문제

```java
public static int[][] graph1(int[][] w, int n){
    // 1. 각 정점이 청색인지를 나타내는 isblue 배열을 선언하고, 가장 가까운 적색 정점을 나타내는 near,
    // 선택된 간선을 나타내는 배열 T, 최솟값을 나타낼 minval, 새로운 적색 점을 나타낼 newred 선언
    boolean isblue[] = new boolean[n];
    int near[] = new int[n];
    int T[][] = new int[n][n];
    int minval = 1000;
    int newred = 0;

    // 2. 입력받은 배열을 0과 무한(여기서는 임시로 10000으로 설정)을 제외하고 모두 부호를 바꾼다.
    for(int i = 0; i < n; i++){
        for(int j = 0; j < n; j++){
            if(w[i][j] != 10000 && w[i][j] != 0) {
                w[i][j] = -1 * w[i][j];
            }
        }
    }

    // 최소 신장 알고리즘 부분(프림 알고리즘 사용함)
    isblue[0] = false; // 3. 정점 0을 적색으로 바꾼다.
    for(int i = 1; i < n; i++){ // 4. 0번을 제외한 나머지 정점을 청색으로 설정하고, 가장 가까운 적색 정점을 0으로 설정한다.
        isblue[i] = true;
        near[i] = 0;
    }
    for(int i = 1; i < n; i++){ // 5. i가 1에서 n-1까지 증가하는 동안,
        minval = 1000; // 5-1. minval을 무한으로 설정하고,
        for(int j = 0; j < n; j++){ // 적색 정점들에 가장 가까운 청색 정점을 찾는다.
            if(isblue[j] && w[j][near[j]] < minval){
                minval = w[j][near[j]];
                newred = j;
            }
            isblue[newred] = false; // 5-2. 정점 newred를 적색으로 바꾼다.
            T[newred][near[newred]] = 1; // 5-3. 가장 가중치가 작은 간선을 T에 추가한다.
        }
        for(int j = 0; j < n; j++){ // 5-4. j가 1에서 n-1까지 증가하는 동안,
            if(isblue[j] && w[j][newred] < w[j][near[j]]){ // 5-5. isblue[j] && w[j][newred] < w[j][near[j]]이면 near[j] = newred
                near[j] = newred;
            }
        }
    }

    T[0][0] = 0; // 처음 추가되면서 1로 만들었던 0,0번을 0으로 만들어준다.

    return T; // T를 반환한다. (1로 표시된 간선이 바로 최대 신장 트리를 나타냄)
}
```

2번문제

```java
public static String[] process(int[] time, int person) { // 각 프로세스의 실행 시간이 담긴 time이라는 배열을 선언하고, 작업자들의 수를 나타내는 person을 입력받는다.
    // (이때, 작업자의 번호는 0 ~ person -1이다.)
    int length = time.length; // 1-1. 프로세스의 수를 저장하는 length 변수를 선언한다.
    int minTime = 0; // 1-2. 작업자마다 대기 시간이 존재할텐데, 작업자들을 모두 통틀어서 가장 적은 대기시간을 저장
    int minTimeIndex = 0; // 1-3. 가장 적은 대기시간을 가진 작업자의 번호를 저장한다.
    int timeIndex = 0; // 1-4. 가장 적은 대기시간을 가진 작업자에게 줄 프로세스 번호인 timeIndex 변수를 선언한다.

    boolean [] isDone = new boolean[length]; // 2-1. 해당 인덱스의 프로세스가 끝났는지를 나타내는 isDone 배열을 선언한다.
    int[] waitTime = new int[person]; // 2-2. 각 작업자들의 프로세스에 따른 대기 시간을 저장하는 waitTime 배열을 선언한다.
    int[] count = new int[person]; // 2-3. 작업자들이 실행할 프로세스의 순서 저장하는 count 배열을 선언한다.
    String[] num = new String[length]; // 2-4. 프로세스의 순서 결과를 저장할 num 배열을 선언한다.

    for(int i = 0; i < length; i++){ // 3-1. i가 0 ~ length-1까지 증가하는 동안, 인덱스를 돌면서 isDone 배열과 num 배열을 각각 false와 빈 문자열로 초기화합니다.
        isDone[i] = false;
        num[i] = "";
    }
    for(int i = 0; i < person; i++){ // 3-2. i가 0 ~ person-1까지 증가하는 동안, 인덱스를 돌면서 waitTime 배열과 count 배열을 0으로 초기화한다.
        waitTime[i] = 0;
        count[i] = 0;
    }

    for(int i = 0; i < length; i++) { // 4-1. i가 0 에서 length-1까지 증가하는 동안
        minTime = 10000; // 4-2. 최소 대기 시간을 무한으로 초기화(여기서는 임시로 10000으로 대체)
        for (int j = 0; j < person; j++) { // 4-3. j가 0에서 person-1까지 증가하는 동안
            if (waitTime[j] < minTime) { // 4-4. waitTime[j]가 minTime보다 작을 때, minTime = waitTime[j], minTimeIndex = j
                minTime = waitTime[j];
                minTimeIndex = j;
            }
        }
        count[minTimeIndex]++; // 4-5. 반복문을 모두 돈 다음에는, count[minTimeIndex]++
        minTime = waitTime[minTimeIndex] + 10000; // 4-6. 최소 대기 시간을 다시 무한으로 초기화
        for (int k = 0; k < length; k++) { // 4-7. k가 0 에서 length-1까지 증가하는 동안
            if((waitTime[minTimeIndex] + time[k] < minTime) && (!isDone[k])){ // 4-8. waitTime[minTimeIndex] + time[k] < minTime이고 k번 프로세스가 끝나지 않았다면
                minTime = waitTime[minTimeIndex] + time[k]; // 4-9. minTime = waitTime[minTimeIndex] + time[k], timeIndex = k
                timeIndex = k;
            }
        }
        num[timeIndex] += minTimeIndex;
        num[timeIndex] += "-" + count[minTimeIndex]; // 5. 몇 번째 사람이 몇 번째로 실행할지 결과 저장
        isDone[timeIndex] = true; // 6. 실행된 프로세스라고 바꿔줌
        waitTime[minTimeIndex] = minTime; // 7. waitTime[minTimeIndex] = minTime
    }
    return num; //8. 결과를 반환한다. 0-1일 경우 0번째 사람이 첫 번째로 실행할 프로세스라는 뜻.
```