탐욕 기법 문제 1

이충기

명지대학교
컴퓨터공학과

문제 1: 작업 실행 순서 정하기

한 개의 프로세서에서 실행할 $n(> 1)$개의 작업들이 있다. i, $1 \leq i \leq n$, 번째 작업의 실행 시간은 t_i이다. 이 작업들은 어떤 순서로도 실행될 수 있다. 그러나 프로세서는 한 번에 한 개의 작업만을 수행할 수 있다. 모든 작업들이 컴퓨터 시스템에서 보내는 총 시간이 최소가 되는 일정을 찾아야 한다. 한 작업이 시스템에서 보내는 시간은 시스템에서 기다리는 시간과 실행 시간의 합이다.

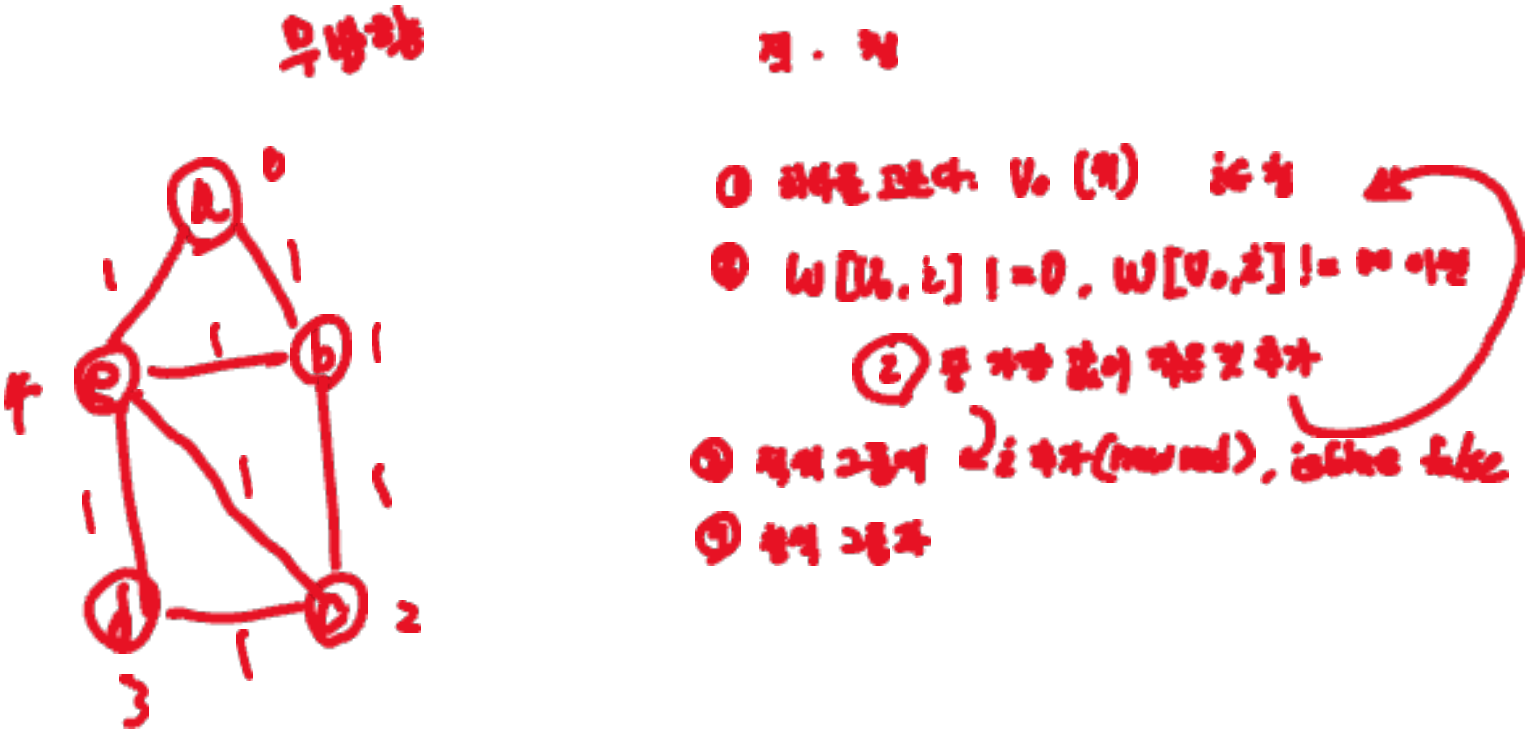
(1)(**10 점**) $n = 3$인 경우를 고려하라. 예를 들면, 첫 번째 작업은 2초가 걸리고 두 번째 작업은 4 초가 걸리고 세 번째 작업은 3초가 걸린다. 작업들을 어떤 순서로 수행하는 것이 세 개의 작업들이 시스템에서 보내는 총 시간을 최소화하는가?

(2)(**40 점**) 이 문제를 위한 탐욕적인 알고리즘을 작성하라.

(3)(**10 점**) (2)에서 작성한 탐욕적인 알고리즘은 항상 최적의 해답을 만들어 내는가?

문제 2: 신장 트리 찾기

(40 점) 프림(Prim)의 최소 비용 신장 트리 알고리즘을 이용하여 간선들에 가중치들이 없는 한 연결된 무방향 그래프의 한 신장 트리를 찾는 알고리즘을 의사코드로 작성하라.



문제①

(1) n=3 이고 각각 2초 4초 3초

(1번) (2번) (3번)

대기시간은 앞의 프로세스들의 실행시간 합이될것

1,2,3 2+(2+4)+(2+4+3) = 2+6+9 = 17

1,3,2 2+(2+3)+(2+4+3) = 2+5+9 = 16 → 답: 1,3,2 번 순서로 실행

2,1,3 4+(4+2)+(2+4+3) = 4+6+9 = 19

2,3,1 4+(4+3)+(2+4+3) = 4+7+9 = 20

3,1,2 3+(3+2)+(2+4+3) = 3+5+9 = 17

3,2,1 3+(3+4)+(2+4+3) = 3+7+9 = 19

(2)

```java
public static int[] process(int[] A, int n){ // 1. 번호 순서대로 실행 시간이 담긴 A라는 배열과 프로세스의 개수 n을 입력받는다.
    //2. 실행시간에 대한 최솟값의 인덱스 min과 순서를 넣을 배열 num, 프로세스가 이미 끝났는지의 여부를 나타내는 배열 done을 선언한다.
    int min;
    int[] num = new int[n];
    int[] done = new int[n];

    //3. done과 num 배열을 모두 0으로 채워 초기화한다.
    Arrays.fill(done, val: 0);
    Arrays.fill(num, val: 0);

    for(int i = 0; i < n; i++){ //4. 지역변수 i가 0에서 n-1까지 증가하는 동안
        int j  = 0; // 4-1. 반복문 속에 지역변수 j를 선언해 0으로 초기화한다.

        for(; j < n; j++) { // 4-2. j가 0에서 n-1까지 증가하는 동안
            if (done[j] == 0) { // 4-3. done[j] 가 0이라면 반복문을 탈출한다.
                break;
            }
        }
        min = j; // 4-4. min 변수 속에 j를 대입한다.

        for(; j < n; j++){ // 4-5. 그 뒤에 j가 0에서 n-1까지 증가하는 동안
            if(done[j] == 0 && A[j] <= A[min]){ // 4-6. done[j]가 0이고 A[j]가 A[min] 이하라면
                min = j; // 4-7. min에 j를 대입하고 그렇지 않다면 아무것도 하지 않는다.
            }
            num[i] = min + 1;// 4-8. num[i]에 min + 1(실제 프로세스 번호는 1부터이므로)을 대입하여 저장한다.
        }
        done[min] = 1; // 4-9. done[min]에 1을 저장하여 해당 min 번 프로세스가 끝났다고 표시한다.
    }
    return num; // 5. num 배열을 반환한다.
}
```

(3) 탐욕적인 알고리즘은 보통 최적의 결과를 내지 않는다.

2. 신장트리 알고리즘

```java
public static int[] process(int[] A, int n){ // 1. 번호 순서대로 실행 시간이 담긴 A라는 배열과 프로세스의 개수 n을 입력받는다.
    //2. 실행시간에 대한 최솟값의 인덱스 min과 순서를 넣을 배열 num, 프로세스가 이미 끝났는지의 여부를 나타내는 배열 done을 선언한다.
    int min;
    int[] num = new int[n];
    int[] done = new int[n];

    //3. done과 num 배열을 모두 0으로 채워 초기화한다.
    Arrays.fill(done, val: 0);
    Arrays.fill(num, val: 0);

    for(int i = 0; i < n; i++){ //4. 지역변수 i가 0에서 n-1까지 증가하는 동안
        int j  = 0; // 4-1. 반복문 속에 지역변수 j를 선언해 0으로 초기화한다.

        for(; j < n; j++) { // 4-2. j가 0에서 n-1까지 증가하는 동안
            if (done[j] == 0) { // 4-3. done[j] 가 0이라면 반복문을 탈출한다.
                break;
            }
        }
        min = j; // 4-4. min 변수 속에 j를 대입한다.

        for(; j < n; j++){ // 4-5. 그 뒤에 j가 0에서 n-1까지 증가하는 동안
            if(done[j] == 0 && A[j] <= A[min]){ // 4-6. done[j]가 0이고 A[j]가 A[min] 이하라면
                min = j; // 4-7. min에 j를 대입하고 그렇지 않다면 아무것도 하지 않는다.
            }
            num[i] = min + 1;// 4-8. num[i]에 min + 1(실제 프로세스 번호는 1부터이므로)을 대입하여 저장한다.
        }
        done[min] = 1; // 4-9. done[min]에 1을 저장하여 해당 min 번 프로세스가 끝났다고 표시한다.
    }
    return num; // 5. num 배열을 반환한다.
}
```