

Report

프로그래머스 프로세스 문제(42587)



과목 : 코딩테스트지도

담당교수 : 이승진 교수님

학부 : IT융합자율학부

학번 : 202114136

이름 : 장준희

제출일 : 2025-03-30

작업 git url: <https://github.com/jjune960/coding-assignments/>

문제 설명

운영체제의 역할 중 하나는 컴퓨터 시스템의 자원을 효율적으로 관리하는 것입니다. 이 문제에서는 운영체제가 다음 규칙에 따라 프로세스를 관리할 경우 특정 프로세스가 몇 번째로 실행되는지 알아내면 됩니다.

1. 실행 대기 큐(Queue)에서 대기중인 프로세스 하나를 꺼냅니다.
2. 큐에 대기중인 프로세스 중 우선순위가 더 높은 프로세스가 있다면 방금 꺼낸 프로세스를 다시 큐에 넣습니다.
3. 만약 그런 프로세스가 없다면 방금 꺼낸 프로세스를 실행합니다.
 - 3.1 한 번 실행한 프로세스는 다시 큐에 넣지 않고 그대로 종료됩니다.

예를 들어 프로세스 4개 [A, B, C, D]가 순서대로 실행 대기 큐에 들어있고, 우선순위가 [2, 1, 3, 2]라면 [C, D, A, B] 순으로 실행하게 됩니다.

현재 실행 대기 큐(Queue)에 있는 프로세스의 중요도가 순서대로 담긴 배열 priorities와, 몇 번째로 실행되는지 알고싶은 프로세스의 위치를 알려주는 location이 매개변수로 주어질 때, 해당 프로세스가 몇 번째로 실행되는지 return 하도록 solution 함수를 작성해주세요.

제한사항

- priorities의 길이는 1 이상 100 이하입니다.
 - priorities의 원소는 1 이상 9 이하의 정수입니다.
 - priorities의 원소는 우선순위를 나타내며 숫자가 클 수록 우선순위가 높습니다.
- location은 0 이상 (대기 큐에 있는 프로세스 수 - 1) 이하의 값을 가집니다.
 - priorities의 가장 앞에 있으면 0, 두 번째에 있으면 1 ... 과 같이 표현합니다.

요구분석

실행 대기 큐에서 프로세스를 순회하면서 우선순위가 더 높은 프로세스일 때만 실행하라.

1차 시도

1. Queue 만들기

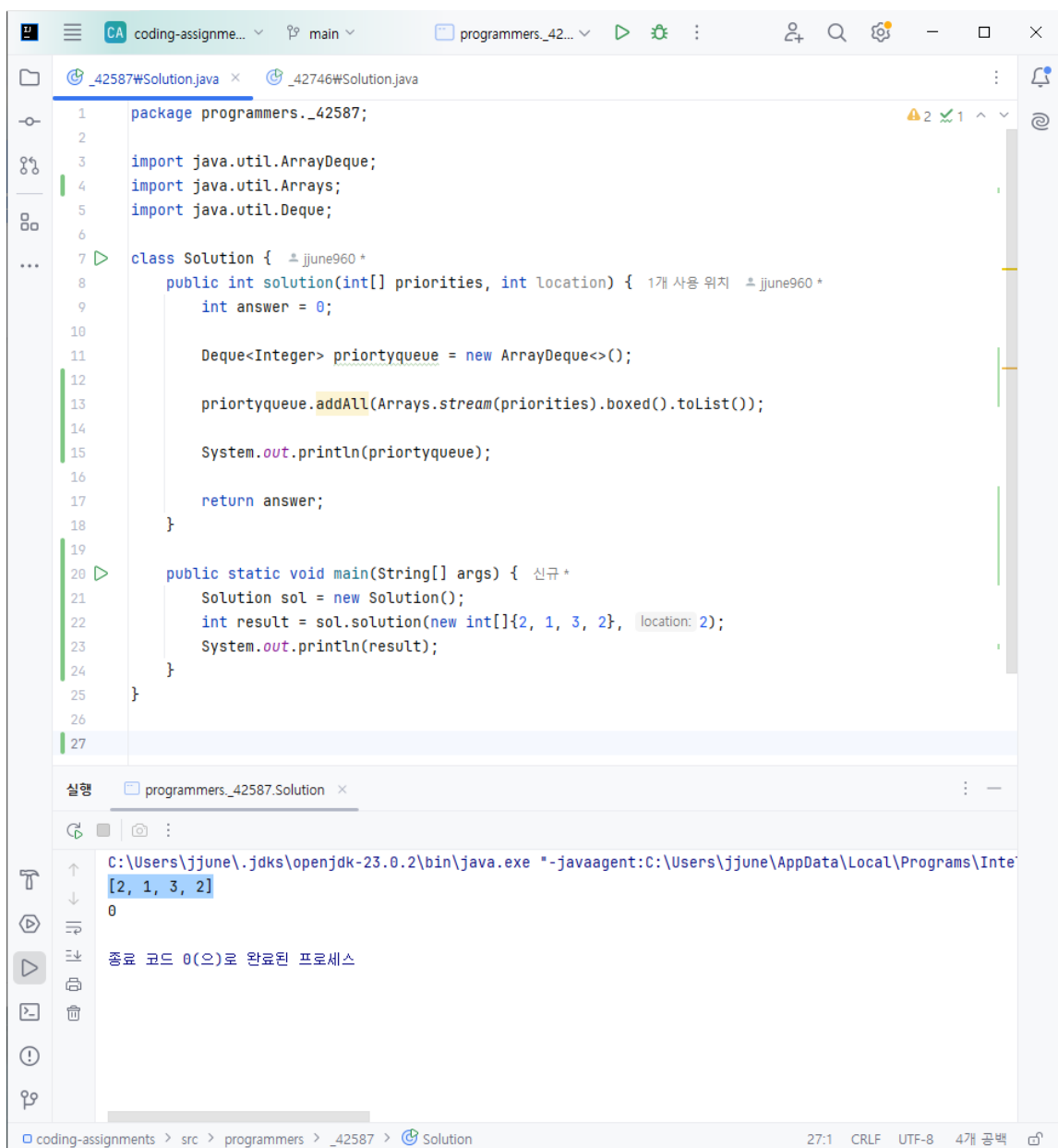
프로세스를 실행하기 위한 Queue를 만들어(priorities) 보겠다.

효율적인 알고리즘인 ArrayDeque를 사용한다.

```
Deque<Integer> priorityqueue = new ArrayDeque<>();
```

2. Queue에 값 삽입

addAll을 이용해 Queue(ArrayDeque)에 값을 삽입한다.



The screenshot shows an IDE with a Java file named `_42587#Solution.java`. The code defines a `Solution` class with a `solution` method and a `main` method. The `solution` method initializes a `Deque<Integer> priorityqueue` and adds elements from an array to it using `addAll`. The `main` method creates a `Solution` object and calls `solution` with an array `{2, 1, 3, 2}` and `location: 2`. The execution output shows the array `[2, 1, 3, 2]` and the result `0`.

```
package programmers._42587;

import java.util.ArrayDeque;
import java.util.Arrays;
import java.util.Deque;

class Solution {
    public int solution(int[] priorities, int location) {
        int answer = 0;

        Deque<Integer> priorityqueue = new ArrayDeque<>();

        priorityqueue.addAll(Arrays.stream(priorities).boxed().toList());

        System.out.println(priorityqueue);

        return answer;
    }

    public static void main(String[] args) {
        Solution sol = new Solution();
        int result = sol.solution(new int[]{2, 1, 3, 2}, location: 2);
        System.out.println(result);
    }
}
```

실행 결과:

```
C:\Users\jjune\jdk\openjdk-23.0.2\bin\java.exe "-javaagent:C:\Users\jjune\AppData\Local\Programs\IntelliJ IDEA\bin\idea-agent-23.0.2\idea-agent-23.0.2.jar" -Dfile.encoding=UTF-8 [2, 1, 3, 2]
0
종료 코드 0(으)로 완료된 프로세스
```

3. 반복문 사용 준비

반복문으로 queue를 만들 준비를 해보겠다.

```
int index = 0; // 인덱스 기록용
int max = priorities.length; // 배열 크기(인덱스 파악용)
int prepriority = 0; // 이전 큐의 우선순위

while (true) {

}
```

4. 인덱스 크기 제한

Return할 index를 제대로 추적하기 위해 사용한다.

```
while (true) {
    if(index >= max)
        max = 0;
}
```

5. 조건문 넣기 (실패)

```
while (true) {
    if(index >= max)
        max = 0;

    int priority = priorityqueue.removeFirst();

    if(priority > prepriority) {
        prepriority = priority;
        //answer = index;
        priorityqueue.addLast(priority);
    }
    else {
        priorityqueue.addFirst(priority);
    }

    index++;
}
```

제대로 작동하지 않는다.

실패 원인 (GPT 사용)

1. 무한 루프

현재 코드에서는 while(true) 루프 내에서 종료 조건이 제대로 구현되지 않아 무한 루프에 빠지게 됩니다.

- **종료 조건 부재:**

출력되어야 하는 문서(예: 찾고자 하는 문서)가 출력되었을 때 루프를 종료해야 하는데, 이 조건이 없으면 무한히 큐를 순회하게 됩니다.

- **인덱스 변수의 잘못된 사용:**

index와 max 변수의 활용이 적절하지 않아서, 언제 루프를 종료할지 결정할 수 없습니다.

2. priorityqueue의 배치 이동으로는 원하는 답을 얻을 수 없음

현재 코드에서는 단순히 priorityqueue의 요소를 이동시킴으로써 우선순위 재배치를 시도하고 있는데, 이 방식으로는 실제 인쇄 순서를 정확하게 구할 수 없습니다.

- **우선순위 비교의 문제:**

단순히 현재 값과 이전 우선순위를 비교하는 것만으로는 "현재 큐에 남아 있는 다른 문서들 중 더 높은 우선순위가 존재하는지"를 판단할 수 없습니다.

- **배치 이동 문제:**

요소를 뒤로 보내거나 앞으로 넣는 것만으로는 어느 시점에 문서가 실제로 인쇄되어야 하는지 결정할 수 없으며, 올바른 출력 순서를 계산할 수 없습니다.

2차 시도 (GPT 활용)

1. process용 queue 만들기

priorities queue대신 process queue를 만들어야 한다.

0부터 priorities 길이까지의 숫자가 들어있는 queue를 만든다.

```
Deque<Integer> processDeque = new ArrayDeque<>();
for (int i = 0; i < priorities.length; i++) {
    processDeque.offer(i);
}
```

2. 현재 문서보다 우선순위가 높은 것이 있는지 확인

큐에서 인덱스를 하나씩 꺼내고, 남은 문서들과 우선순위를 비교합니다.

현재 문서보다 우선순위가 높은 문서가 있으면, 해당 문서를 큐의 뒤쪽으로 이동시킵니다.

그렇지 않으면, 문서를 출력 처리하고 출력 순서를 기록합니다.

```
while (!processDeque.isEmpty()) {
    int current = processDeque.removeFirst();
    boolean isHighest = true;

    // 남은 문서들 중 현재 문서보다 우선순위가 높은 것이 있는지 확인
    for (int index : processDeque) {
        if (priorities[index] > priorities[current]) {
            isHighest = false;
            break;
        }
    }
}
```

3. 문서 재배치 또는 출력 처리

우선순위가 가장 높은 것부터 출력한다.

```
if (!isHighest) {
    processDeque.offer(current);
} else {
    // 문서를 출력 처리(카운트 증가)
    answer++;
    if (current == location) {
        return answer;
    }
}
```

실행결과

```
C:\Users\jjune\.jdk\openjdk-23.0.2\bin\java.exe
1
```

종료 코드 0(으)로 완료된 프로세스

정상적으로 실행된다.

배운 점

Queue를 만들 배열을 신중하게 선택하고 무한 루프를 방지하는 조건을 명확히 설정해야 한다는 점을 배웠다.

github commit hash

Commits on Mar 30, 2025

docs: 프로세스 배운 점 jjune960 committed now	4387dbd		
docs: 프로세스 2차 시도 (실행결과) jjune960 committed 3 minutes ago	c01ec6c		
docs: 프로세스 2차 시도 (3. 문서 재배치 또는 출력 처리) jjune960 committed 19 minutes ago	8d0e081		
docs: 프로세스 2차 시도 (2. 현재 문서보다 우선순위가 높은 것이 있는지 확인) jjune960 committed 28 minutes ago	e3dffaa		
docs: 프로세스 2차 시도 (1. process용 queue 만들기) jjune960 committed 38 minutes ago	dca49f3		
feat: 프로세스 1차 시도 (4. 인덱스 크기 제한) jjune960 committed 1 hour ago	13f2235		
feat: 프로세스 1차 시도 (4. 인덱스 크기 제한) jjune960 committed 2 hours ago	ea52d89		
feat: 프로세스 1차 시도 (4. 인덱스 크기 제한) jjune960 committed 2 hours ago	ea471bf		
feat: 프로세스 1차 시도 (3. 반복문 사용 준비) jjune960 committed 2 hours ago	865140b		
feat: 프로세스 1차 시도 (2. Queue에 값 삽입) jjune960 committed 3 hours ago	80c8667		
feat: 프로세스 1차 시도 (1. Queue 만들기) jjune960 committed 3 hours ago	1829b02		
Create _42587(Process) library and docs folder and file jjune960 committed 10 hours ago	b82c51d		