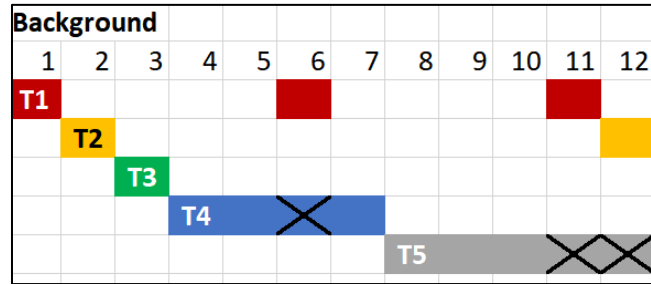


1. background

- periodic task 의 주기가 짧을수록 task 의 우선순위가 높습니다.



- 위의 그림과 같이 우선순위가 낮은 task 를 수행할 차례여도 높은 우선순위의 task 의 주기가 돌아오면 높은 우선순위의 task 를 먼저 수행합니다.
- aperiodic task 는 periodic task 를 수행하지 않는 slack 에서 수행합니다.
- 구현시 stack 의 자료구조의 성질을 가진 vector 의 push_back(), pop_back()과 find 함수를 적절히 이용하여 구현했습니다. 주기가 짧은 task 가 우선순위가 높기 때문에 짧은 주기의 task 가 candidate vector(수행할 후보 task 를 저장하는 vector)에서 짧은 주기의 task 를 먼저 꺼내서 수행합니다. 이때, 우선순위를 주기 위해서 aperiodic task 는 100, slack 은 101 로 정의했습니다.
- 또한, 각 task 의 deadline 을 만족하지 못할 경우 deadline miss 를 출력하도록 구현했습니다.
- 하지만 다 구현하고 난 후 다시 생각을 해보니, linked list 로 우선순위 큐를 구현하여 활용했으면 가장 상위 노드가 우선순위가 가장 높은 노드이기 때문에, find 함수를 쓰지 않을 수 있었다는 생각이 들었습니다.

- 수행결과

```

C:\WINDOWS\system32\cmd.exe
periodic task의 개수 입력 : 5
aperiodic task의 개수 입력 : 3
periodic task의 computation time과 주기 입력 : 1 10 1 15 1 5 5 60 3 30
aperiodic task의 computation time과 주기 입력 : 1 10 1 29 2 7
0~1 : 1
1~2 : 2
2~3 : 3
3~4 : 4
4~5 : 4
5~6 : 1
6~7 : 4
7~8 : 5
8~9 : 5
9~10 : 5
10~11 : 1
11~12 : 2
12~13 : 5
13~14 : 5
14~15 : 100
15~16 : 1
16~17 : 3
17~18 : 100
18~19 : 100
19~20 : 101
20~21 : 1
21~22 : 2
22~23 : 101
23~24 : 101

```

```

C:\WINDOWS\system32\cmd.exe
23~24 : 101
24~25 : 101
25~26 : 1
26~27 : 101
27~28 : 101
28~29 : 101
29~30 : 100
30~31 : 1
31~32 : 2
32~33 : 3
33~34 : 4
34~35 : 4
35~36 : 1
36~37 : 4
37~38 : 101
38~39 : 101
39~40 : 101
40~41 : 1
41~42 : 2
42~43 : 101
43~44 : 101
44~45 : 101
45~46 : 1
46~47 : 3
47~48 : 101
48~49 : 101
49~50 : 101
50~51 : 1
51~52 : 2
52~53 : 101
53~54 : 101
54~55 : 101
55~56 : 1
56~57 : 101
57~58 : 101
58~59 : 101
59~60 : 101
background average waiting time : 8.33333
계속하려면 아무 키나 누르십시오 . . .

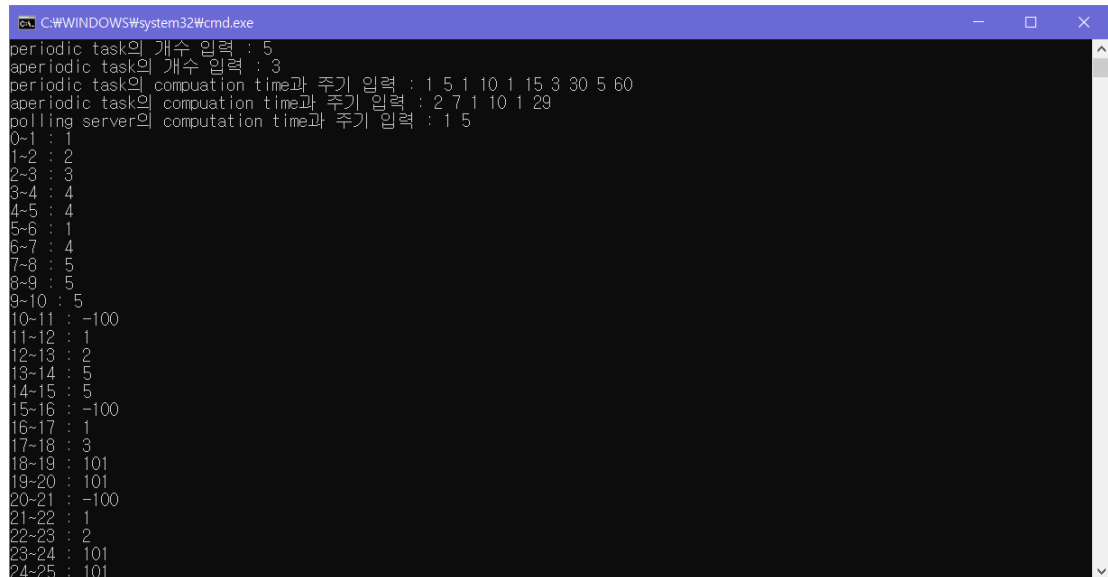
```

주어진 test case 를 넣고 수행했을 경우 background 의 average waiting time 은 $\{(14-7) + (17-7) + (18-10) + (29-29)\} / 3 = 8.333$ 이 나옵니다.

2. polling

- polling 도 background 와 마찬가지로 periodic task 의 주기가 짧을수록 task 의 우선순위가 높고, 우선순위가 낮은 task 를 수행할 차례여도 높은 우선순위의 task 의 주기가 돌아오면 높은 우선순위의 task 를 먼저 수행합니다.
- 하지만 background 와 달리 polling 은 polling server 의 주기마다 aperiodic task 의 arrival time 을 check 하고, polling server 의 computation time 만큼 aperiodic task 를 수행합니다. 따라서 polling server 의 주기일 때, 가장 우선순위가 높은 task 는 aperiodic task 입니다.
- polling 도 background 와 비슷하게 vector 을 이용하여 구현하였고, polling server 의 주기일 때 aperiodic task 의 우선순위를 가장 높게 주기 위해서 구현시 aperiodic task 는 -100, slack 은 101 로 정의했습니다.

- 수행결과



```
C:\WINDOWS\system32\cmd.exe
periodic task의 개수 입력 : 5
aperiodic task의 개수 입력 : 3
periodic task의 computation time과 주기 입력 : 1 5 1 10 1 15 3 30 5 60
aperiodic task의 computation time과 주기 입력 : 2 7 1 10 1 29
polling server의 computation time과 주기 입력 : 1 5
0-1 : 1
1-2 : 2
2-3 : 3
3-4 : 4
4-5 : 4
5-6 : 1
6-7 : 4
7-8 : 5
8-9 : 5
9-10 : 5
10-11 : -100
11-12 : 1
12-13 : 2
13-14 : 5
14-15 : 5
15-16 : -100
16-17 : 1
17-18 : 3
18-19 : 101
19-20 : 101
20-21 : -100
21-22 : 1
22-23 : 2
23-24 : 101
24-25 : 101
```

```
C:\WINDOWS\system32\cmd.exe
22-23 : 2
23-24 : 101
24-25 : 101
25-26 : 1
26-27 : 101
27-28 : 101
28-29 : 101
29-30 : 101
30-31 : -100
31-32 : 1
32-33 : 2
33-34 : 3
34-35 : 4
35-36 : 1
36-37 : 4
37-38 : 4
38-39 : 101
39-40 : 101
40-41 : 1
41-42 : 2
42-43 : 101
43-44 : 101
44-45 : 101
45-46 : 1
46-47 : 3
47-48 : 101
48-49 : 101
49-50 : 101
50-51 : 1
51-52 : 2
52-53 : 101
53-54 : 101
54-55 : 101
55-56 : 1
56-57 : 101
57-58 : 101
58-59 : 101
59-60 : 101
background average waiting time : 7.33333
```

주어진 test case 를 넣고 수행했을 경우 polling 의 average waiting time 은 $\{(10-7) + (15-7) + (20-10) + (30-29)\} / 3 = 7.333$ 이 나옵니다.

따라서 polling server 는 background 보다 average waiting time 이 더 짧은 것을 알 수 있습니다.