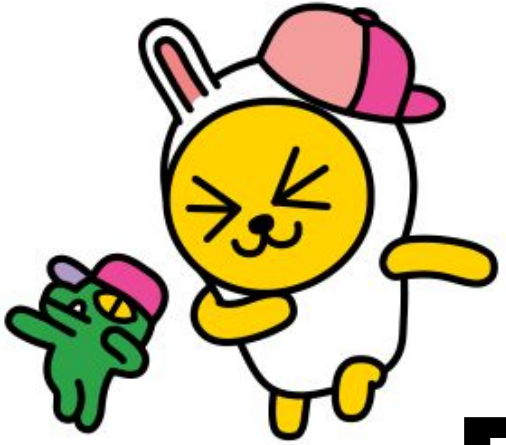
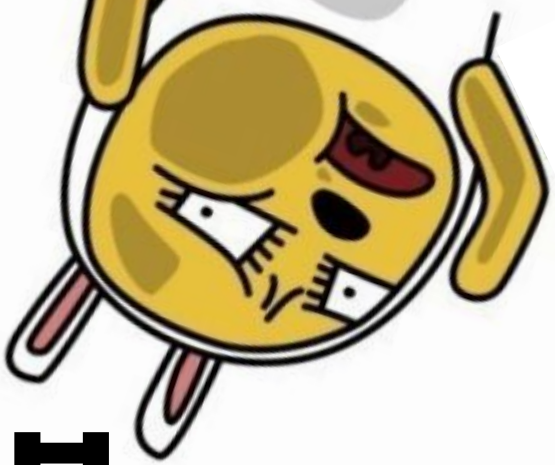
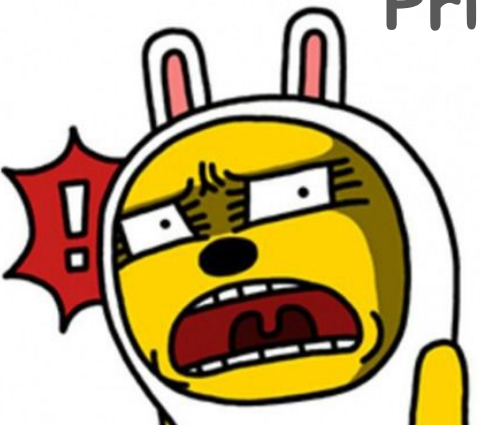
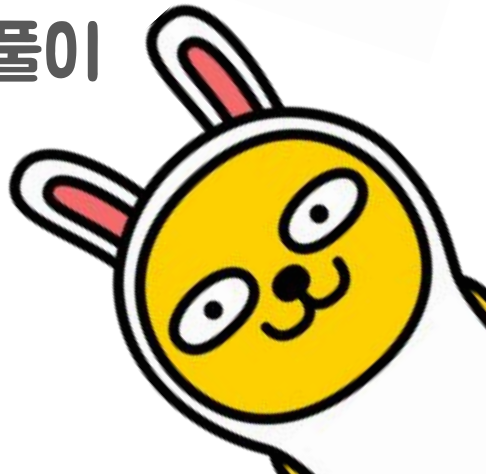




무지의 먹방 라이브

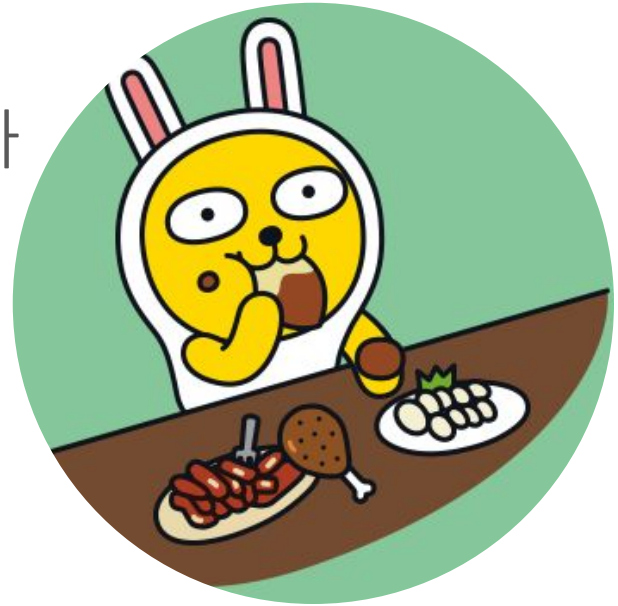
Priority Queue를 활용한 알고리즘 풀이





문제 선정 이유

- 무지의 먹방 라이브 (링크)
  - 높은 레벨(Lv4)의 문제를 풀고 싶었다
  - 풀이에 필요한 알고리즘이 쉬워 보였다
  - 무지 그림이 *MUZIMUZI* 귀여웠다



문제 소개

- 무지는 1번 음식부터 먹기 시작하며, 회전판은 번호가 증가하는 순서대로 음식을 무지 앞으로 가져다 놓는다. 마지막 번호의 음식을 섭취한 후에는 회전판에 의해 다시 1번 음식이 무지 앞으로 온다. 무지는 음식 하나를 1초 동안 섭취한 후 남은 음식은 그대로 두고, 다음 음식을 섭취한다.
    - 다음 음식이란, 아직 남은 음식 중 다음으로 섭취해야 할 가장 가까운 번호의 음식을 말한다.
    - 회전판이 다음 음식을 무지 앞으로 가져오는데 걸리는 시간은 없다고 가정한다.
- 무지가 먹방을 시작한 지 K 초 후에 네트워크 장애로 인해 방송이 잠시 중단되었다. 무지는 네트워크 정상화 후 다시 방송을 이어갈 때, 몇 번 음식부터 섭취해야 하는지를 알고자 한다.
- **각 음식을 모두 먹는데 필요한 시간이 담겨있는 배열 food_times, 네트워크 장애가 발생한 시간 K 초가 매개변수로 주어질 때 몇 번 음식부터 다시 섭취하면 되는지 return 하도록 solution 함수를 완성하라.**

Priority Queue

- **우선순위 큐**는 일반적인 큐의 순서대로 원소를 제거하는 선입선출(FIFO) 방식이 아니라, **순서에 상관없이 추가가 되어도 값을 뽑을 때는 가장 작은값(오름차순)을 제거**하는 특성을 지닌 자료 구조
- 우선순위 큐(Priority Queue)를 사용하여 값을 오름차순 정렬로 제일 작은 값을 빼내는 get()을 사용
- **파이썬** : **from queue import PriorityQueue**
- PriorityQueue에 데이터를 넣는 것은 **put()**, 뽑는 것은 **get()**을 사용
  데이터를 넣을 때는 Tuple 형식으로 (정렬되는 기준 key 값, value값)

```python
from queue import PriorityQueue # 우선순위 queue를 통한 풀이
# http://www.daleseo.com/python-priority-queue/ (사용법 참조)
def solution(food_times, k):
    if sum(food_times) <= k: return -1
    # 음식을 먹는 시간을 다 더해도 k를 못 넘길 경우 : 어차피 -1 리턴

    answer = 0 # 답 초기화
    pq = PriorityQueue() # 우선순위 queue (put -> get ~ 가장 작은 값을 제거)
    length = len(food_times) # 남은 음식의 가짓수
    for i in range(length):
        pq.put((food_times[i], i+1)) # for문을 돌려서 소요시간과 음식 idx 맵핑
#    print(pq.queue)

    sum_value = 0 # 먹방한 초
    prev = 0 # 마지막으로 다 먹은 음식 초값 (얼만큼 round 돌았나)

    while ((pq.queue[0][0] - prev) * length) <= k - sum_value:
    # 정전 시간 전에 다 먹을 수 있나? 계속 검증 (먹을 수 없다면...)
        now = pq.get()[0] # 한 바퀴를 돌았기 때문에 가장 작은 값을 줄인다
        sum_value += (now - prev) * length # round 돌은만큼
        length -= 1 # 길이가 줄어든다
        prev = now # 가장 작은 값이 기준이 되어 모두에게 뺀 효과
#        print(pq.queue)

    result = sorted(pq.queue, key=lambda x:x[1])
    # 남은 요소들을 우선순위 기준으로 재정렬 = 남은 음식들 번호

    return result[(k - sum_value) % len(pq.queue)][1]
    #  정전되기 직전 회차에서의 음식 먹는 순서
```

Q&A