SWEA 5215 햄버거 다이어트

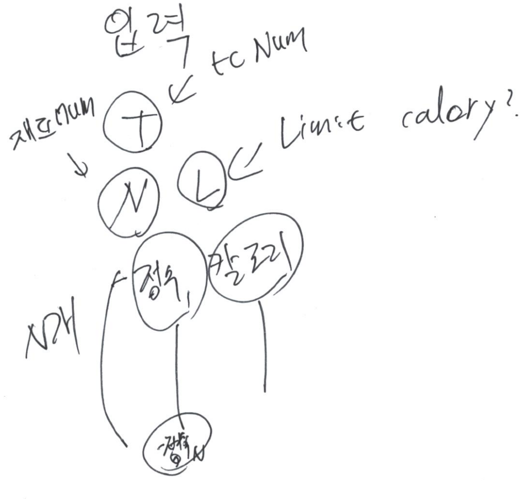
날짜	@2021년 8월 9일
링크	
열	
태그	D3 SWEA

SWEA 5215 D3

준비한 재료를 조합해서 콘다 (새로 재료로 만들지 않는다)

다이어트 칼로리 이하의 조합중에서

민기가 가장 선호하는 햄버거를 만들어주자



=> 조합 문제

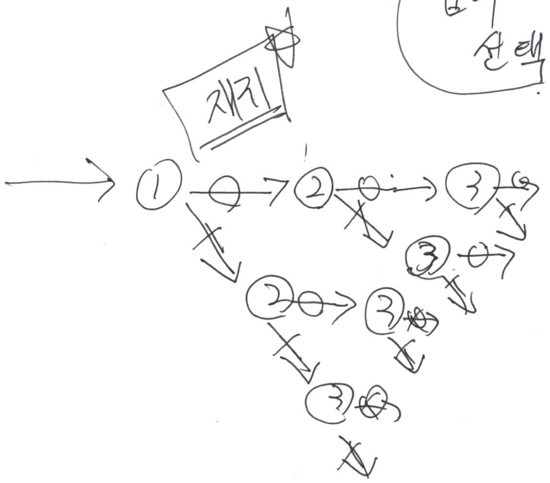
~~nCk~~ 때

Subset 부분집합 개념이다.

n개중에 ~~k개~~

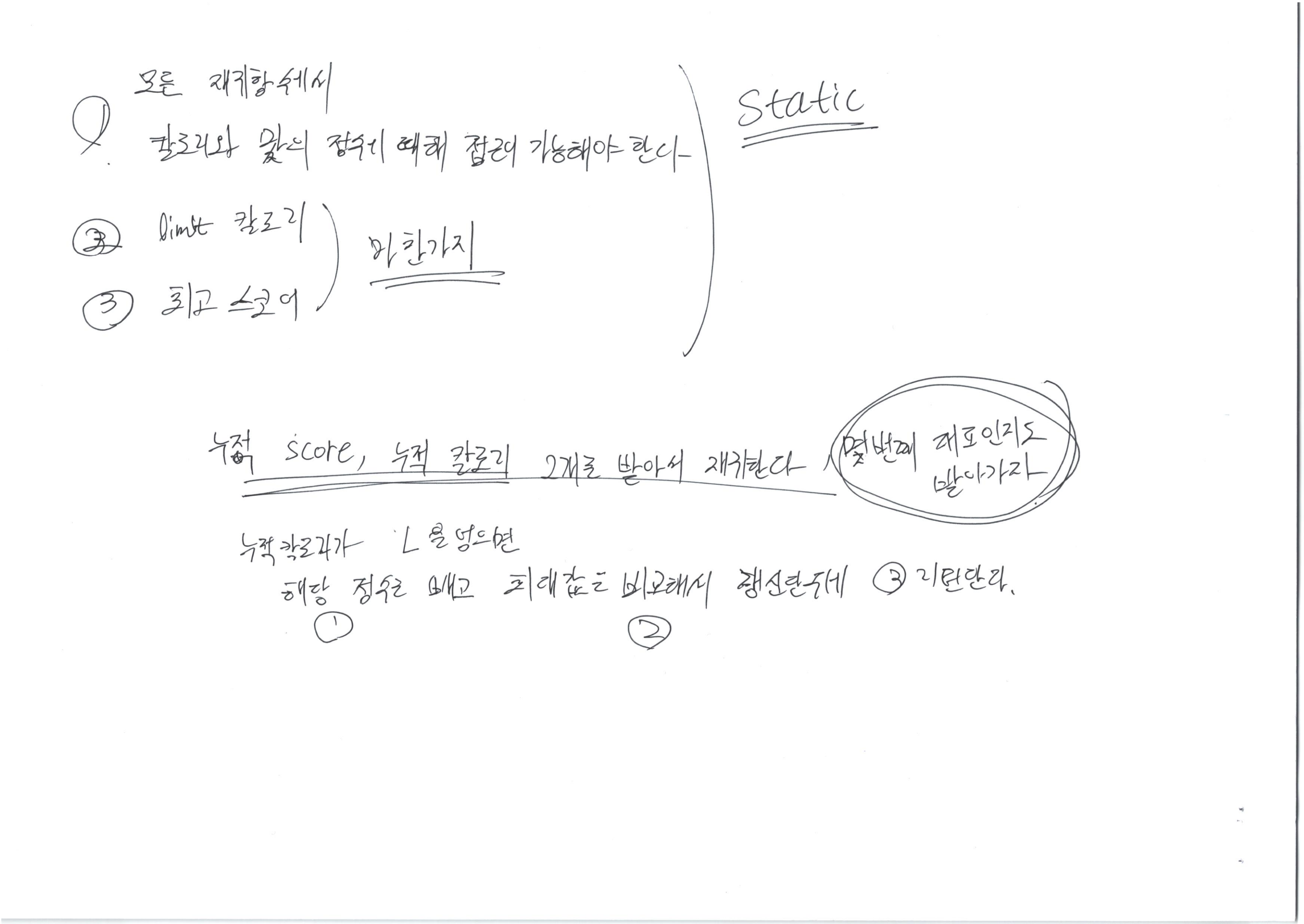
k개는 아니지만

합이 ≤ ~~[illegible]~~인 만큼은 자유롭게 선택



if ~~[illegible]~~ sum ≥ return

sum+≥L



① 모든 재귀함수에서 칼로리와 맛의 점수에 대해 접근 가능해야 한다 — Static

② limit 칼로리

③ 최고 스코어

) 다 한가지

누적 score, 누적 칼로리 2개를 받아서 재귀한다 / 몇번째 재료인지도 받아가자

누적칼로리가 L을 넘으면

해당 점수를 빼고 ① 최대값을 비교해서 ② 갱신한후에 ③ 리턴한다.

```
import java.awt.BufferCapabilities;
import java.io.BufferedReader;
```

```java
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.StringTokenizer;

//public class Solution {
public class SWEA5215_D3_1 {
  static IngreInfo[] info = null;
  static int num;
  static int limit;
  static int maxScore;

  public static void main(String[] args) throws NumberFormatException, IOException {
    System.setIn(new FileInputStream("C:/CodingStudy/SWEA/D3/5215_input.txt"));
    BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
    int T = Integer.parseInt(br.readLine());
    StringTokenizer st;
    for (int tc = 1; tc <= T; tc++) {
      maxScore = 0;
      st = new StringTokenizer(br.readLine());
      num = Integer.parseInt(st.nextToken());
      limit = Integer.parseInt(st.nextToken());
      info = new IngreInfo[num];
      for (int i = 0; i < num; i++) {
        info[i] = new IngreInfo();
      }

      for (int i = 0; i < num; i++) {
        st = new StringTokenizer(br.readLine());
        info[i].score = Integer.parseInt(st.nextToken());
        info[i].calorie = Integer.parseInt(st.nextToken());
      } // end input

      findMax(0, 0, 0); // start find MaxScore
      System.out.println("#" + tc + " " + maxScore);

    } // end tc
  }// end main

  public static void findMax(int idx, int sumScore, int sumCalorie) {
    if (idx >= num) {
      if (sumCalorie <= limit) {
        if (maxScore < sumScore)
          maxScore = sumScore;
      }
      return;
    }

    if (sumCalorie + info[idx].calorie <= limit)
      findMax(idx + 1, sumScore + info[idx].score, sumCalorie + info[idx].calorie);
    findMax(idx + 1, sumScore, sumCalorie);

  }

  public static class IngreInfo {
    int score = 0;
    int calorie = 0;
  }
}
```