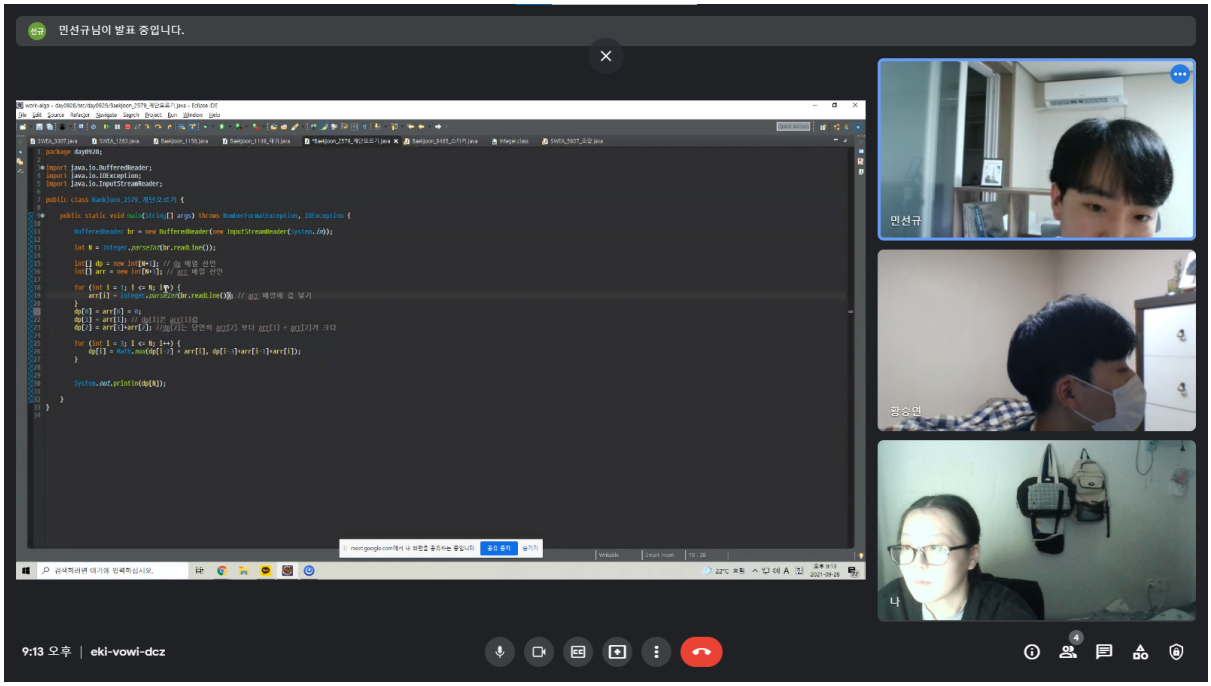
9월 3주차 스터디

주제 : DynamicPrograming

멤버 : 민선규, 황승연, 우정연

발표일시 : 9월 28일 21시



문1. 백준 9465 스티커 실버2 - https://www.acmicpc.net/problem/9465

```
package day0928;

import java.util.Scanner;

public class Baekjoon_9465_스티커 {

  public static void main(String[] args) {

    Scanner sc = new Scanner(System.in);

    int tc = sc.nextInt();

    for(int TC =1 ; TC <=tc ; TC++) {

      int n = sc.nextInt();

      int[][] map = new int[2][n+1];
      int[][] dp = new int[2][n+1];

      for (int i = 0; i < 2; i++) {
        for (int j = 1; j <= n; j++) {
          map[i][j] = sc.nextInt();
        }
      }

      dp[0][1] = map[0][1];
      dp[1][1] = map[1][1];

      for (int i = 2; i <=n; i++) {
        dp[0][i] = Math.max(dp[1][i-1], dp[1][i-2]) + map[0][i];
        dp[1][i] = Math.max(dp[0][i-1], dp[0][i-2]) + map[1][i];
      }

      System.out.println(Math.max(dp[0][n], dp[1][n]));
```

```
    }
  }
}
```

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.StringTokenizer;



public class BJ9465_스티커 {
  static int TC, N;
  static int[][] memo;
  static int ans;
  public static void main(String[] args) throws NumberFormatException, IOException {
    BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
    StringTokenizer st;
    StringBuilder sb = new StringBuilder();
    TC = Integer.parseInt(in.readLine());

    for(int tc = 1 ; tc <= TC ; tc++) {               // TC만큼 반복
      N = Integer.parseInt(in.readLine());
      memo = new int[2][N + 1];                 // 현재 위치의 스티커를 포함했을 때의 최댓값이 저장되도록
      for(int i = 0 ; i < 2 ; i++) {                // memo의 초기값은 스티커의 점수
        st = new StringTokenizer(in.readLine(), " ");
        for(int j = 1 ; j <= N ; j++) {
          memo[i][j] = Integer.parseInt(st.nextToken());
        }
      }

      if(N >= 2) {                          // N이 2보다 클 경우
        memo[0][2] = memo[0][2] + memo[1][1];         // 두번째 열의 값은 왼쪽 대각선에 위치한 스티커의 점수를 더함
        memo[1][2] = memo[1][2] + memo[0][1];
        for(int j = 3 ; j <= N ; j++) {
          for(int i = 0 ; i < 2 ; i++) {            // 상, 하 두개의 스티커에 대해
            int max = Math.max(memo[0][j - 2], memo[1][j - 2]);   // 비교값 - 왼왼쪽 상, 하, 왼쪽값 총 3개 비교
            max = Math.max(max, memo[1 - i][j - 1]);
            memo[i][j] += max;                // 현재 위치의 스티커 값에 구한 최댓값을 더함
          }
        }
      }

      ans = Math.max(memo[0][N], memo[1][N]);           // 가장 오른쪽의 스티커는 무조건 포함하므로 상 하 두개 중 최댓값이 결과

      sb.append(ans + "\n");
    }

    System.out.println(sb);
  }
}
```

문2. 백준 2579 계단오르기 실버3 - https://www.acmicpc.net/problem/2579

```
package day0928;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

public class Baekjoon_2579_계단오르기 {

  public static void main(String[] args) throws NumberFormatException, IOException {

    BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

    int N = Integer.parseInt(br.readLine());

    int[] dp = new int[N+1]; // dp 배열 선언
    int[] arr = new int[N+1]; // arr 배열 선언

    for (int i = 1; i <= N; i++) {
      arr[i] = Integer.parseInt(br.readLine()); // arr 배열에 값 넣기
    }
    dp[0] = arr[0] = 0;
    dp[1] = arr[1]; // dp[1]은 arr[1]값
    dp[2] = arr[1]+arr[2]; //dp[2]는 당연히 arr[2] 보다 arr[1] + arr[2]가 크다

    for (int i = 3; i <= N; i++) {
      dp[i] = Math.max(dp[i-2] + arr[i], dp[i-3]+arr[i-1]+arr[i]);
    }
```

```
    System.out.println(dp[N]);

  }
}
```

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;


public class BJ2579_계단오르기 {
  static int N;
  static int[] stair, memo;
  static int ans;
  public static void main(String[] args) throws NumberFormatException, IOException {
    BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
    N = Integer.parseInt(in.readLine());
    stair = new int[N + 1];       // 계단의 점수 저장
    memo = new int[N + 1];        // 해당 계단을 포함한(자기 자신을 포함한) 해당 계단까지의 최댓값 저장
    for(int i = 1 ; i <= N ; i++) {
      stair[i] = Integer.parseInt(in.readLine());
    }

    memo[1] = stair[1];
    if(N >= 2) {            // 두번째 계단에서는 첫번째 계단의 점수를 더함
      memo[2] = stair[1] + stair[2];
    }
    if(N >= 3) {            // 세번째 계단에서부터 연속 3개 계단이 안되므로 2개씩 더해 비교(자기 자신을 포함)
      memo[3] = Math.max(stair[1] + stair[3], stair[2] + stair[3]);
    }
    for(int i = 4 ; i <= N ; i++) {   // 네번째 계단부터는 두 가지를 비교해 저장
      memo[i] = Math.max(memo[i - 2], memo[i - 3] + stair[i - 1]) + stair[i];
    }
    System.out.println(memo[N]);
  }
}
```

```
package practice;

import java.util.LinkedList;
import java.util.Queue;
import java.util.Scanner;

public class Boj_2579_계단오르기 {

  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);

    int N = sc.nextInt();
    int[] arr = new int[N+1];
    for(int i=1;i<=N;i++) {
      arr[i] = sc.nextInt();
    }
    //입력 끝

    int[][] D = new int[3][N+1];      //한걸음으로 갔을 때의 최대 점수와 두걸음으로 갔을 때의 최대점수를 저장할 이차원 배열
    D[1][1] = arr[1];           //첫번째 계단까지의 최대 점수 초기화

    for(int i=2;i<=N;i++) {
      D[1][i]= Math.max(D[1][i-2], D[2][i-2])+arr[i]; //i번째 계단을 한걸음으로 가려면 i-2번째 계단에서 점프해서 올 수밖에 없으므로 i-2번째 계단에서의 최
      D[2][i]= D[1][i-1]+arr[i];            //i번째 계단을 두번째걸음으로 도착하려면 i-1번째 계단에서 걸어와야만 하므로 i-1번째 계단을 한걸음으로 왔을때의 최
    }
    int result = Math.max(D[1][N], D[2][N]);      //N번째 계단을 한걸음으로 오거나 두번째 걸음으로 온 점수 중 높은점수를 결과로
    System.out.println(result);

  }

  static class Info{
    int score;
    int walk;
    int now;
    public Info(int score, int walk, int now) {
      super();
      this.score = score;
      this.walk = walk;
      this.now = now;
    }
```

```
  }
}
```

문3. 백준 12865 평범한 배낭 골드5 https://www.acmicpc.net/problem/12865

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.StringTokenizer;


public class BJ12865_평범한배낭 {
  static int N, K;
  static int[] weights, values, dp;
  public static void main(String[] args) throws IOException {
    BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
    StringTokenizer st = new StringTokenizer(in.readLine(), " ");
    N = Integer.parseInt(st.nextToken());
    K = Integer.parseInt(st.nextToken());
    weights = new int[N + 1];
    values = new int[N + 1];
    dp = new int[K + 1];

    for(int i = 1 ; i <= N ; i++) {
      st = new StringTokenizer(in.readLine(), " ");
      weights[i] = Integer.parseInt(st.nextToken());
      values[i] = Integer.parseInt(st.nextToken());
    }

    for(int n = 1 ; n <= N ; n++) {           // 물건 개수만큼
      for(int k = K ; k >= weights[n] ; k--) {    // 뒤에서부터 계산할 경우 일차원 배열로 끝낼 수 있음
        // 무게 k가 되도록 2가지 경우를 만들었을 때 가치 비교 - 현재 물건을 포함했을 때, 현재 물건을 포함하지 않았을 때(이전 dp배열값)
        dp[k] = Math.max(dp[k], dp[k - weights[n]] + values[n]);
      }
    }

    System.out.println(dp[K]);              // 무게 K일 때의 최대 가치값을 출력
  }
}
```

```
package practice;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.StringTokenizer;

public class Boj_12865_평범한배낭 {

  public static void main(String[] args) throws IOException {
    BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

    StringTokenizer st = new StringTokenizer(br.readLine());
    int N = Integer.parseInt(st.nextToken());
    int K = Integer.parseInt(st.nextToken());

    int[] weight = new int[N+1];
    int[] value = new int[N+1];

    for(int i=1;i<=N;i++) {
      st = new StringTokenizer(br.readLine());
      weight[i] = Integer.parseInt(st.nextToken());
      value[i] =  Integer.parseInt(st.nextToken());
    }
    //입력 끝

    int[][] D = new int[N+1][K+1];        //N번째 물건까지 고려했을때, K무게까지 고려했을때 최대가치를 저장하기 위한 이차원 배열

    for(int i=1;i<=N;i++) {
      for(int j=1;j<=K;j++) {
        if(j>=weight[i]) {          //현재 추가된 물건을 담을 수 있다면
          D[i][j] = Math.max(D[i-1][j], D[i-1][j-weight[i]]+value[i]);//이전물건까지 고려된 최대무게에서 현재 추가된 물건의 무게를 빼고 현재 추가된 물건
        }else {           //현재 추가된 물건 담을 수 없다면
          D[i][j] = D[i-1][j];  //이전물건까지의 최대가치 그대로
        }
      }
    }

    System.out.println(D[N][K]);
  }
```

```
}
```