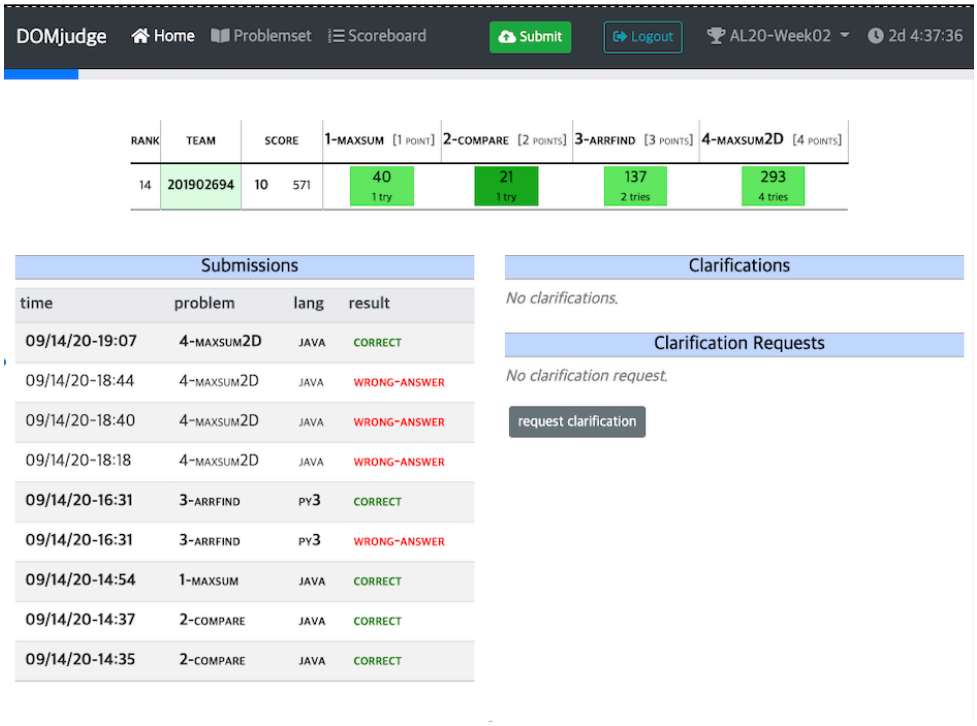
실습 3주차

201902694-박재우



문제1 : 최대부분합

```java
package algorithm.week3;

import java.util.Arrays;
import java.util.Scanner;

public class first {


    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String temp = sc.nextLine();
        String[] ar = temp.split(" ");

        int start, end,ans = 0;
        for(start = 0; start <=ar.length-1; start++)
        {
            int sum = 0;
            for(end = start; end<=ar.length-1;end++) {
                sum+=Integer.parseInt(ar[end]);
                if(ans < sum) ans = sum;
            }
        }
        System.out.print(ans);
    }

}
```

시간복잡도 :

for문이 이중으로 쓰였으므로 O(n^2) 이다.

자신의 생각 :

1차원의 구조와 원하는 결과에 맞는 효율적인 코딩을 원하는 문제였다고 생각한다.

질문 :

이보다 더 간단하게 짤 수도 있을것 같은데 우선 나는 잘 모르겠다.

느낀점 :

최대부분합 문제만으로도 많은 시간이 걸리고 남들보다 느려서 더많이 공부해야할것

문제2 : 비교와 정렬

```java
package algorithm.week3;

import java.util.Arrays;

public class second {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Scanner sc = new Scanner(System.in);
        int max = 0;
        int n = sc.nextInt();
        int[] integers = new int[n];
        for(int i = 0; i < n; i++)
            integers[i] = sc.nextInt();
        Arrays.sort(integers);

        for(int j=0;j<integers.length;j++) {
            max+=j;
        }
        System.out.println(max);

        for(int i=0;i<integers.length-1;i++)
            System.out.print(integers[i]+" ");
        System.out.print(integers[integers.length-1]);

    }
}
```

시간복잡도 :

for문이 1번씩만 쓰였으므로 시간복잡도는 O(n)이다.

자신의 생각 :

이문제는 생각보다 쉬워서 금방 끝낼 수 있었다. Sort를 사용해서 간단하게 정렬을 한뒤에 최대 비교갯수는 단순한 수학적 계산을 통해 따로 산출해 내었다. 따라서 비교적 쉽다고 생각이 들었다.

느낀점 :

최대비교횟수가 n이 증가할 수록 기하급수적으로 늘어간다는 것을 알게되었다. 보다 빠른 연산을 위해서는 최대 비교횟수가 적은 알고리즘을 사용해야겠다는것을 느꼈다.

문제3 : 배열탐색

```python
import heapq

def max(nums, k):
  heap = []
  for num in nums:
    heapq.heappush(heap, (-num,num))

  kth_max = None
  for _ in range(k):
    kth_max = heapq.heappop(heap)[1]
  return kth_max


nm = input()
nm = [int(i) for i in nm.split()]
b = input()
b = [int(i) for i in b.split()]
print(max(b, nm[1]+1))
```

시간복잡도 :

for문이 1번씩만 쓰였으므로 시간복잡도는 O(n)이다.

자신의 생각 :

이문제는 처음에는 heap sort를 사용해야 하는줄 알았지만 그렇지 않다는 것을 알고 난 뒤에는 생각보다 쉬워서 금방 끝낼 수 있었다. Sort를 사용해서 간단하게 정렬을 한뒤에 최대 비교갯수는 단순한 수학적 계산을 통해 따로 산출해 내었다. 따라서 비교적 쉽다고 생각이 들었다.

느낀점 :

처음에는 문제의 방향을 이해하지못해 헤맸지만 알고난 뒤에는 금방 해결 할 수 있었다. 처음에는 자바를 이용해서 풀려고 했지만 파이썬은 라이브러리에서 미리 제공을 하고 있기에 파이썬으로 풀었는데 파이썬을 사용하면 할 수록 매우 간단하고 편리하다는것을 느꼈다.

문제3 :2D 최대 부분합

(코드의 길이가 길어 코드는 하단에 첨부하였습니다)

시간복잡도 : for문이 4번 중첩되어있으므로 O(n^4)이다

자신의 생각 :

무엇보다 가장 어려웠던 문제라고 생각한다. 처음에 시작 좌표와 끝좌표를 설정해서 그사이의 값들을 전부 더해야겠다는 생각은 했지만 그것을 코드로 구현하는데에 많은 애를 먹었다. 최종적으로 해결하긴 했지만 생각보다 많이 어려웠다.

질문 :

이런 방법이외에 포함배제의 원리(?)를 통해 굉장히 간단하게 짤 수 있다는 것을 알게되었는데 그 방법에 대해서 자세히는 모르겠다.

느낀점 :

문제의 해결을 위한 알고리즘을 생각해 내는데 많은 시간이 걸리는데 많은 연습을 통해 이 부족한 점에 대해 조금더 열심히 공부를 해야겠다고 느꼈다.

```java
package algorithm.week3;

import java.util.Scanner;

public class forth {

  private static int error_prevent(int front,int end,int e_front,int e_end,int[][] sum) {
    int temp = sum[e_end][e_front];
    if(front > 0 && end > 0)
      temp = temp-sum[e_end][front-1]-sum[end-1][e_front]+sum[end-1][front-1];
    else if(front >0)
      temp = temp - sum[e_end][front-1];
    else if(end > 0)
      temp = temp-sum[end-1][e_front];
    return temp;
  }

  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    int n = sc.nextInt();
    int[][] set = new int[n][n];
    int[][] sum = new int[n][n];

    for(int i = 0; i < n; i++) {
      for(int j = 0; j< n; j++)
        set[i][j] = sc.nextInt();
    }
    for(int i = 0; i < n; i++) {
      for(int j = 0; j< n; j++)
        sum[i][j] = 0;
    }
    for(int i=0; i<n;i++)
      sum[0][i] = set[0][i];

    for (int i=1;i<n;i++) { //세로
      for(int j=0;j<n;j++) {
        sum[i][j] = sum[i-1][j]+set[i][j];
      }
    }

    for (int i=0;i<n;i++) {//세로합의 가로
      for (int j=1;j<n;j++) {
        sum[i][j] = sum[i][j-1] + sum[i][j];
      }
    }
    int aws = -2147483648;
    for (int front=0;front<n;front++) {
      for (int end=0;end<n;end++) {
        for (int e_front=front;e_front<n;e_front++) {
          for (int e_end=end;e_end<n;e_end++) {
            int temp = error_prevent(front,end,e_front,e_end,sum);
            if(aws < temp)
              aws = temp;
          }
        }
      }
    }

    System.out.print(aws);
  }
```