Fast campus Copyright FASTCAMPUS Corp.All Rights Reserved

JavaScript
투 포인터
투 포인터 알고리즘 문제 풀이

투 포인터 알고리즘 문제 풀이 | 코딩 테스트에서 자주 등장하는 투 포인터 이해하기

강사 나동빈

fast campus Copyright FASTCAMPUS Corp.All Rights Reserved

JavaScript
투 포인터

투 포인터 알고리즘 문제 풀이

fast campus Copyright FASTCAMPUS Corp.All Rights Reserved

두 수의 합: https://www.acmicpc.net/problem/3273

혼자 힘으로 풀어보기

JavaScript
투 포인터
투 포인터
문제 풀이

문제 제목: 두 수의 합

문제 난이도: ★★☆☆☆

문제 유형: 투 포인터

추천 풀이 시간: 40분

fast campus Copyright FASTCAMPUS Corp.All Rights Reserved

두 수의 합: https://www.acmicpc.net/problem/3273

JavaScript
투 포인터
투 포인터
문제 풀이

문제 풀이 핵심 아이디어

- 다음과 같이 원소가 주어진 경우를 가정한다.
- $arr = [5, 12, 7, 10, 9, 1, 2, 3, 11]$
- 원소를 오름차순 정렬한 결과는 다음과 같다.
- $sorted = [1, 2, 3, 5, 7, 9, 10, 11, 12]$

fast campus Copyright FASTCAMPUS Corp.All Rights Reserved

두 수의 합: https://www.acmicpc.net/problem/3273

JavaScript
투 포인터
투 포인터
문제 풀이

문제 풀이 핵심 아이디어

- 모든 정수가 양의 정수이고 서로 다르므로, **투 포인터**를 사용할 수 있다.
- **초기화**: $start = 0, end = N - 1$
- 오름차순 정렬이 되어 있다면, 다음이 성립한다.

1. 시작점(start)을 1 증가시키면, $sorted[s] + sorted[e]$가 증가한다.
2. 끝점(end)을 1 감소시키면, $sorted[s] + sorted[e]$가 감소한다.

- 따라서 현재의 합($sorted[s] + sorted[e]$)와 X를 매번 비교한다.

fast campus Copyright FASTCAMPUS Corp.All Rights Reserved

정답 코드 예시

```
let file = require('fs').readFileSync('/dev/stdin');
let input = file.toString().split('\n');

let n = Number(input[0]); // 원소의 개수(N)
let arr = input[1].split(" ").map(Number); // 전체 원소 입력
let x = input[2]; // X 입력

arr.sort((a, b) => a - b); // 오름차순 정렬
```

fast campus Copyright FASTCAMPUS Corp.All Rights Reserved

두 수의 합: https://www.acmicpc.net/problem/3273

JavaScript 투 포인터
투 포인터 문제 풀이

정답 코드 예시

```
// [현재 구현] start가 고정된 상태에서 end를 최대한 왼쪽으로 이동시키는 구현
let result = 0;
let start = 0; // 시작점(start)
let end = n - 1; // 끝점(end)
while (true) {
  // 현재 합이 x보다 크다면, 합을 줄이기 위해 end를 감소
  while (end > 0 && arr[start] + arr[end] > x) {
    end -= 1;
  }
  if (arr[start] + arr[end] == x) { // 현재 합이 x인 경우
    result += 1; // 카운트
    end -= 1; // 모든 정수가 다르다는 조건이 있으므로, end를 감소
  }
  start += 1;
  // [유의] 탈출 조건에 유의
  if (start >= end) break; // 서로 다른 2개의 정수를 고르는 [조합]이므로
}
console.log(result);
```

fast campus Copyright FASTCAMPUS Corp.All Rights Reserved

수들의 합 2: https://www.acmicpc.net/problem/2003

혼자 힘으로 풀어보기

문제 제목: 수들의 합 2

문제 난이도: ★★☆☆☆

문제 유형: 투 포인터

추천 풀이 시간: 40분

fast campus Copyright FASTCAMPUS Corp.All Rights Reserved

수들의 합 2: https://www.acmicpc.net/problem/2003

JavaScript
투 포인터
투 포인터
문제 풀이

문제 풀이 핵심 아이디어

- N개의 수로 된 수열 $A[1], A[2], \cdots, A[N]$이 있다.
- 이 수열의 i번째 수부터 j번째 수까지의 합을 고려하자.

→ $A[i] + A[i+1] + \cdots + A[j-1] + A[j]$가 M이 되는 경우의 수를 계산해야 한다.

- 이때, 다음의 두 가지 특성이 존재한다.

1. start가 증가하면 구간 합이 감소한다.
2. end가 증가하면 구간 합이 증가한다.

fast campus Copyright FASTCAMPUS Corp.All Rights Reserved

정답 코드 예시

```
let input = require('fs').readFileSync('/dev/stdin');
input = input.toString().split('\n');

let [n, m] = input[0].split(' ').map(Number);
let data = input[1].split(' ').map(Number);
```

fast campus Copyright FASTCAMPUS Corp.All Rights Reserved

수들의 합 2: https://www.acmicpc.net/problem/2003

정답 코드 예시

```
let cnt = 0;
let intervalSum = 0;
let end = 0;

// start를 차례대로 증가시키며 반복
for (let start = 0; start < n; start++) {
  // end를 가능한 만큼 이동시키기
  while (intervalSum < m && end < n) {
    intervalSum += data[end];
    end += 1;
  }
  // 부분합이 m일 때 카운트 증가
  if (intervalSum == m) cnt += 1;
  intervalSum -= data[start];
}
console.log(cnt);
```

fast campus Copyright FASTCAMPUS Corp.All Rights Reserved

집배원 한상덕: https://www.acmicpc.net/problem/2842

혼자 힘으로 풀어보기

문제 제목: 집배원 한상덕

문제 난이도: ★★★★☆

문제 유형: 투 포인터, 깊이 우선 탐색(DFS)

추천 풀이 시간: 70분

fast campus Copyright FASTCAMPUS Corp.All Rights Reserved

집배원 한상덕: https://www.acmicpc.net/problem/2842

JavaScript
투 포인터
투 포인터
문제 풀이

문제 풀이 핵심 아이디어

- 전체 공간은 $N \times N$의 행렬로 표현된다.
- 출발 지점(우체국)은 "P", 집은 "K", 목초지는 "."로 표현된다.
- 현재 위치에서 인접한 8칸으로 이동 가능하다.

[목표]

1. "P"에서 출발하여 모든 "K"를 방문한 뒤에 다시 "P"로 돌아온다.
2. 방문하는 칸 중 가장 높은 곳과 낮은 곳의 차이를 **최소화**한다.

fast campus Copyright FASTCAMPUS Corp.All Rights Reserved

집배원 한상덕: https://www.acmicpc.net/problem/2842

문제 풀이 핵심 아이디어

[예시 1]

- 아래의 경우 (0, 0): 3 → (0, 1): 2 → (1, 1): 4 → (1, 2): 2로 이동하여 **정답은 2**이다.

마을 정보

P	.	.
.	K	K
.	.	.

고도 정보

3	2	4
7	4	2
2	3	1

fast campus Copyright FASTCAMPUS Corp.All Rights Reserved

문제 풀이 핵심 아이디어

[예시 2]

- 아래의 경우 (0, 2): 4 → (0, 1): 3 → (0, 0): 3 → (0, 1): 3 → (1, 1): 5 → (2, 1): 3 → (2, 0): 8 → (2, 1): 3 → (2, 2): 7로 이동하여 **정답은 5**이다.

마을 정보

K	.	P
.	.	.
K	.	K

고도 정보

3	3	4
9	5	9
8	3	7

fast campus Copyright FASTCAMPUS Corp.All Rights Reserved

집배원 한상덕: https://www.acmicpc.net/problem/2842

JavaScript 투 포인터
투 포인터 문제 풀이

정답 코드 예시

```
let file = require('fs').readFileSync('/dev/stdin');
let input = file.toString().split('\n');

let n = Number(input[0]);
let arr = []; // 마을 정보
for (let i = 1; i <= n; i++) {
  arr.push(input[i].split(""));
}
let height = []; // 고도 정보
for (let i = n + 1; i <= n + n; i++) {
  height.push(input[i].split(" ").map(Number));
}

// 이동 가능한 인접한 8가지 칸
let dx = [-1, -1, -1, 0, 0, 1, 1, 1];
let dy = [-1, 0, 1, -1, 1, -1, 0, 1];
```

fast campus Copyright FASTCAMPUS Corp.All Rights Reserved

집배원 한상덕: https://www.acmicpc.net/problem/2842

JavaScript 투 포인터
투 포인터 문제 풀이

정답 코드 예시

```
function dfs(x, y) {
  visited[x][y] = true; // 방문 처리
  if (arr[x][y] == "K") cnt += 1; // 집(K)인 경우 카운트
  for (let i = 0; i < 8; i++) {
    let nx = x + dx[i];
    let ny = y + dy[i];
    // 범위를 벗어나는 경우 무시
    if (nx < 0 || nx >= n || ny < 0 || ny > n) continue;
    if (!visited[nx][ny]) { // 아직 방문한 적 없는 경우
      // 정해진 범위 [left, right] 안에 해당하는 고도인 경우
      if (height[nx][ny] >= uniqueHeight[left] && height[nx][ny] <= uniqueHeight[right]) {
        dfs(nx, ny);
      }
    }
  }
}
```

fast campus Copyright FASTCAMPUS Corp.All Rights Reserved

집배원 한상덕: https://www.acmicpc.net/problem/2842

JavaScript 투 포인터
투 포인터 문제 풀이

정답 코드 예시

```
let uniqueHeight = new Set();
let target = 0;
let result = 1e9;
for (let i = 0; i < n; i++) {
  for (let j = 0; j < n; j++) {
    uniqueHeight.add(height[i][j]);
    if (arr[i][j] == "P") { // 출발 지점(P)
      startX = i;
      startY = j;
    }
    if (arr[i][j] == "K") target += 1; // 전체 집(K)의 개수
  }
}
uniqueHeight = [...uniqueHeight]; // 고유한 고도(height) 값 정렬
uniqueHeight.sort((a, b) => a - b);
```

fast campus Copyright FASTCAMPUS Corp.All Rights Reserved

집배원 한상덕: https://www.acmicpc.net/problem/2842

정답 코드 예시

```
let left = 0;
let right = 0;
let middle = 0;
for (let i = 0; i < uniqueHeight.length; i++) {
  // 출발 지점의 높이(height)를 초기 right으로 설정
  if (uniqueHeight[i] == height[startX][startY]) {
    right = i;
    middle = i;
    break;
  }
}
```

fast campus Copyright FASTCAMPUS Corp.All Rights Reserved

JavaScript 투 포인터
투 포인터 문제 풀이

정답 코드 예시

```
while (true) {
  // 방문 가능한 집의 수(cnt)가 target보다 작다면, right을 증가
  while (true) {
    visited = [];
    for (let i = 0; i < n; i++) visited.push(new Array(n).fill(false));
    cnt = 0;
    dfs(startX, startY); // 깊이 우선 탐색(DFS)으로 방문 가능한 집 카운트
    if (right < uniqueHeight.length - 1 && cnt < target) right += 1;
    else break;
  }
  if (cnt == target) { // 모든 집을 방문 가능한 경우
    // 가장 높은 고도와 낮은 고도의 차이 [최솟값] 기록
    result = Math.min(result, uniqueHeight[right] - uniqueHeight[left]);
  }
  left += 1;
  // [유의] 탈출 조건에 유의
  if (left > middle || right >= uniqueHeight.length) break;
}
console.log(result);
```