

Devmon

알고리즘 문제 해결 도우미

김동연, 김민종

2024.12 LLM

01

Key Issues

02

Devmon's Solution

03

About Devmon

현존 문제

- 해설 찾기 어려움
 - 대부분의 경우 다른 사람의 블로그를 통해 해설을 보게 됨
 - 흔치 않은 문제일 땐 해설이 없다!
 - 찾더라도 이미 남의 코드(답)를 보게되어 학습의 효과가 떨어짐

해결책

- 자세한 해설 제공
 - 권장 알고리즘
 - 해당 알고리즘 설명
 - 풀이 접근 방법
 - 코드 제공(Pseudo, Python)
 - 단계별 코드 설명
 - 시간복잡도
- 관련 문제 제공

DEMO

알고리즘 문제 해결 도우미

<https://www.acmicpc.net/problem/16236>

분석하기

권장되는 알고리즘

너비 우선 탐색

알고리즘 설명

너비 우선 탐색(BFS)은 그래프나 트리의 탐색 알고리즘 중 하나로, 시작 정점에서부터 인접한 정점들을 먼저 탐색한 후, 점차 멀리 있는 정점들을 탐색하는 방식입니다.

BFS는 주로 큐(Queue)를 사용하여 구현되며, 최단 경로 문제나 연결 요소 찾기 등에 자주 사용됩니다.

풀이 접근 방법

이 문제는 아기 상어가 물고기를 먹으며 성장하는 과정을 시뮬레이션하는 문제로, BFS를 사용하여 아기 상어가 이동할 수 있는 최단 경로를 탐색합니다.

단계적으로 다음과 같은 방식으로 접근합니다:

초기 상태 설정: 주어진 공간에서 아기 상어의 위치를 찾고, 초기 크기와 먹은 물고기 수를 설정합니다.

BFS를 통한 탐색: 큐를 사용하여 상어가 이동할 수 있는 모든 경로를 탐색하며, 먹을 수 있는 물고기를 찾습니다.

물고기 선택: 거리가 가장 가까운 물고기를 우선으로 선택하며, 여러 마리가 있을 경우 가장 위쪽, 왼쪽에 있는 물고기를 선택합니다.

상어의 이동 및 성장: 상어가 물고기를 먹으면 해당 위치로 이동하고, 먹은 물고기 수가 상어의 크기와 같아지면 크기를 증가시킵니다.

반복: 더 이상 먹을 수 있는 물고기가 없을 때까지 위 과정을 반복합니다.

경과 출력: 상어가 물고기를 먹을 수 있는 경로를 출력합니다.

프롬프트 엔지니어링

역할 지시

```
const SYSTEM_PROMPT =`  
You are a programming expert dedicated to assisting educators who are teaching programming to beginners. When  
-`
```

"프로그래밍 전문가"로서의 역할 지시

출력 구조 명시

```
{  
    "required_algorithm": ${tag_name},  
    "key_concepts": "A broader step-by-step explanation of theoretical concepts required for solving the problem  
    "solution_approach": "A clear explanation of the reasoning and strategy behind solving the problem. Focus on  
    "solution": {  
        "pseudo_code": Pseudocode is an informal way to describe an algorithm. Please follow these rules:  
  
            Variable Declaration: Specify the variables and their initial values.  
            Example: Set count = 0  
  
            Conditional Statements: Use IF, ELSE, ELSE IF to describe conditions.  
            Example:  
                IF (condition)  
                    Do something  
                ELSE  
                    Do something else  
  
            Loops: Use FOR, WHILE, etc., to describe repetitive tasks.  
            Example:  
                FOR each item in list  
                    Do something  
  
            Function Calls: Mention the function name along with its input and output.  
            Example: Call function_name(input) -> result  
  
            Input and Output: Clearly describe the input and output processes.  
            Example:  
                INPUT: User's age  
                OUTPUT: Eligibility status  
                The pseudocode should be concise, properly indented, and include comments."  
,  
    "code": "Python code that implements the above pseudo_code. Include detailed comments for beginners to understand the logic.",  
    "explain": "Output in Markdown format, starting with ## Step 1. A step-by-step breakdown of the code, explaining the logic and key concepts.",  
    "time_complexity": "The time complexity of the solution, including both best-case and worst-case scenarios in Big O notation."}  
}
```

JSON 형태의 출력 구조 명시

퓨 샷 프롬프팅

Here is an example of the response format:

Example 1:

```
{  
    "required_algorithm": "동적 프로그래밍",  
    "key_concepts": "이 문제를 해결하기 위해서는 동적 프로그래밍의 기본 개념을 이해해야 합니다. 동적 프로그래밍은 큰 문제를 작은 문제로 나누어 해결하는 방법입니다.",  
    "solution_approach": "이 문제는 그리드 형태의 미로에서 이동할 수 있는 경로가 제한되어 있으므로, 각 위치에서 최대 사탕 수를 계산하기 위해 동적 프로그래밍을 사용합니다.",  
    "solution": {  
        "pseudo_code": "Set N, M = INPUT values\nCreate a 2D array dp of size (N+1) x (M+1) initialized to 0\nFor each cell in dp, calculate the maximum candy count by considering moves from adjacent cells.",  
        "code": "N, M = map(int, input().split())\ngrid = [list(map(int, input().split())) for _ in range(N)]\ndp = [[0] * (M + 1) for _ in range(N + 1)]\nfor i in range(1, N + 1):  
    for j in range(1, M + 1):  
        if grid[i - 1][j] == 1:  
            dp[i][j] = dp[i - 1][j] + 1  
        if grid[i][j - 1] == 1:  
            dp[i][j] = max(dp[i][j], dp[i][j - 1] + 1)  
print(dp[N][M])",  
        "explain": "## Step 1. 입력 받기\n첫 번째 줄에서 N과 M을 입력받아 미로의 크기를 설정합니다.\n## Step 2. 그리드와 DP 테이블을 초기화합니다.\nDP 테이블은 (N+1) x (M+1) 크기로 초기화됩니다. 모든 값은 0입니다.",  
        "time_complexity": "O(N * M)"  
    }  
}
```

Example 2:

```
{  
    "required_algorithm": "기하학",  
    "key_concepts": "이 문제를 해결하기 위해서는 기하학적 개념, 특히 원과 점의 관계를 이해해야 합니다. 주어진 출발점과 도착점이 각 행성계의 중심에 위치하는 원 안에 있는지 여부를 판단하여 출발점과 도착점이 원에 대해 어떤 관계인지를 판별합니다.",  
    "solution_approach": "이 문제는 주어진 출발점과 도착점이 각 행성계의 원 안에 있는지 여부를 판단하여 출발점과 도착점이 원에 대해 어떤 관계인지를 판별합니다.",  
    "solution": {  
        "pseudo_code": "Set T = INPUT value // 테스트 케이스 수\nFOR each test case\n    INPUT (x1, y1) // 출발점\n    INPUT (x2, y2) // 도착점\n    If distance((x1, y1), (x2, y2)) <= T, then print 'Yes'\n    Else print 'No'",  
        "code": "import math\n\ndef distance(x1, y1, cx, cy):  
    return math.sqrt((x1 - cx) ** 2 + (y1 - cy) ** 2)",  
        "explain": "## Step 1. 거리 계산 함수 정의\n'distance'라는 함수를 정의하여 두 점 간의 거리를 계산합니다. 이는 피타고라스 정리를 활용한 것입니다.",  
        "time_complexity": "O(T * n)"  
    }  
}
```

출력 구조에 알맞은 소수의 예시 제공

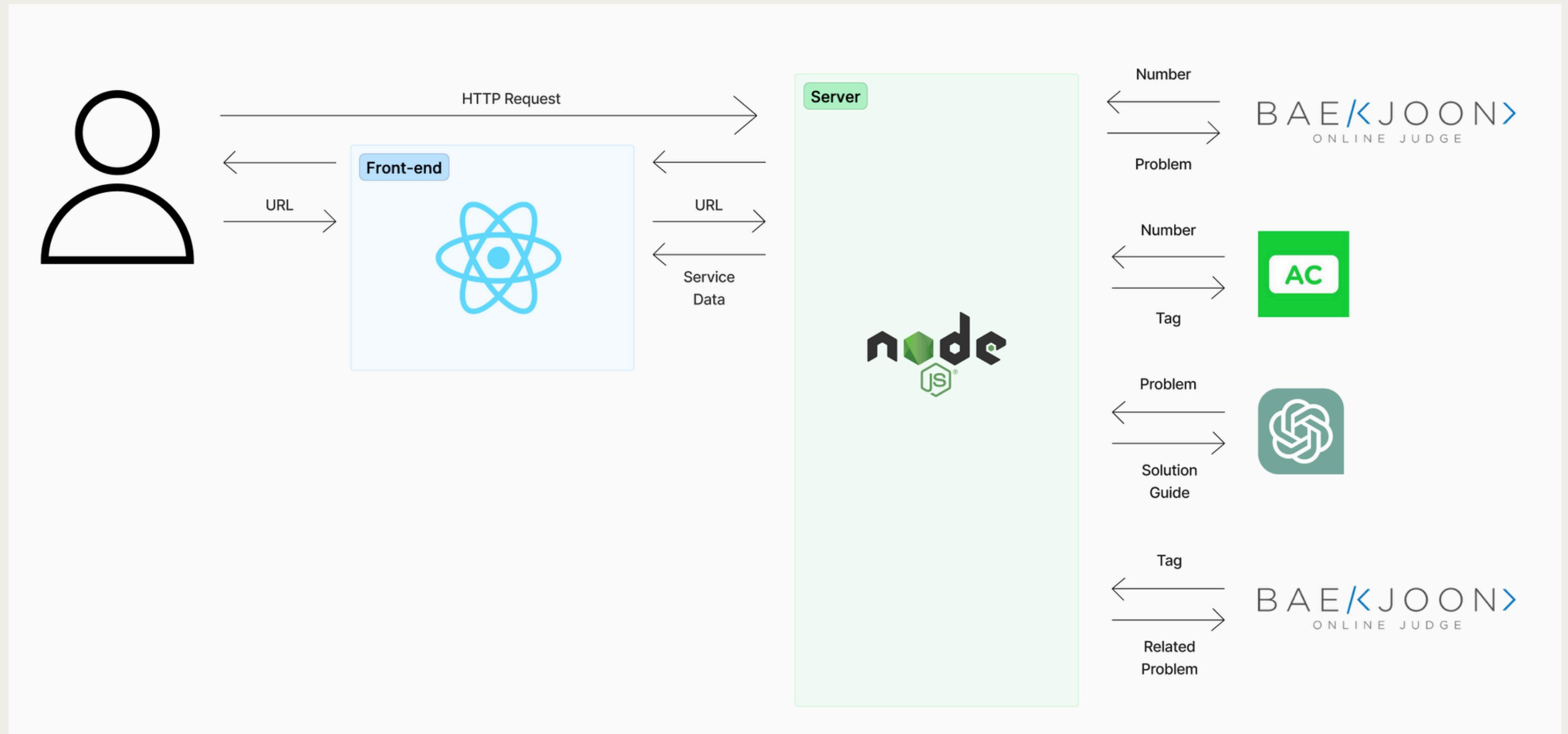
완성 기반 프롬프팅

```
Please explain this problem in Korean and complete the remaining items in the same format and structure as  
{  
    "required_algorithm": ${tag_name},  
    `;  
};
```

JSON의 일부만 제공, 나머지를 작성하도록 요청

아키텍쳐 다이어그램

아키텍쳐 다이어그램



정확도

성능 테스트

- Solved.ac 난이도 별 문제 5개씩 선정
- 문제 별 5번의 시도
- 정답률 및 소요시간 기록

성능 테스트 결과

	A	B	C	D	E	F	G	H	I	J
1	문제 티어	문제 번호	정답	시도	정답률	응답시간		문제 티어	정답률	응답시간
2	Bronze	2338	5	5	100.00%	20.58		Bronze	100.00%	16.452
3		1296	5	5	100.00%	17.59		Silver	92.00%	19.284
4		2884	5	5	100.00%	13.71		Gold	56.00%	16.818
5		1526	5	5	100.00%	13.66		Platinum	0.00%	23.9
6		1816	5	5	100.00%	16.72				
7								Total	62.00%	19.1135
8	Silver	1149	4	5	80.00%	13.62				
9		1158	5	5	100.00%	29.9				
10		1026	4	5	80.00%	21.88				
11		1002	5	5	100.00%	17.18				
12		1463	5	5	100.00%	13.84				
13										
14	Gold	7579	5	5	100.00%	11.97				
15		2252	4	5	80.00%	22.56				
16		11066	0	5	0.00%	13.21				
17		16236	4	5	80.00%	20.36				
18		11049	1	5	20.00%	15.99				
19										
20										
21	Platinum	1395	0	5	0.00%	23.55				
22		11377	0	5	0.00%	18.53				
23		1671	0	5	0.00%	28.29				
24		1017	0	5	0.00%	27.07				
25		1006	0	5	0.00%	22.06				

성능 테스트 - 얼마나 빨리 풀까?

H	I	J	
문제 티어	정답률	응답시간	
Bronze	100.00%	16.452	
Silver	92.00%	19.284	
Gold	56.00%	16.818	
Platinum	0.00%	23.9	
Total	62.00%	19.1135	

- 의외로 Gold의 문제가 응답시간이 가장 짧음
- 문제 길이에 영향을 받진 않음

성능 테스트 - 얼마나 잘 풀까?

H	I	J
문제 티어	정답률	응답시간
Bronze	100.00%	16.452
Silver	92.00%	19.284
Gold	56.00%	16.818
Platinum	0.00%	23.9
Total	62.00%	19.1135

- 티어의 난이도와 정답률은 반비례

개선점

- 난이도별 설명 제공
 - 초/중/고급 유저 맞춤형 설명 생성
- 코드 확인
 - 제공된 코드와 해설 비교, 틀린 부분 자동 분석
- GPT 모델 최적화
 - 난이도에 따라 Mini(빠름) 또는 4o(정밀) 활용
- 에러 대응
 - 자주 틀리는 경우(시간 초과, 런타임 에러 등) 버튼 클릭으로 새로운 코드 제공

부록

문제의 영향

- 시간 낭비
- 학습의 연속성
- 비효율적 학습 과정

왜 유용한 프롬프트인가?

- 단순한 코드 생성 도구를 넘어, 교육적인 도구로서의 프로그래밍
- 초보자부터 중급자까지 포괄적인 학습 제공
- 이론과 실습의 균형으로 문제 해결 능력 강화

기대효과

이 서비스가 주는 가치는 무엇인가?

- 단순한 코드 생성 도구를 넘어, 교육적인 도구로서의 프로그래밍
- 초보자부터 중급자까지 포괄적인 학습 제공
- 이론과 실습의 균형으로 문제 해결 능력 강화

기술적 구현 및 시스템 구조

Frontend

- URL 입력란
- 결과 및 해설 페이지
- 추천문제

Backend

- Node.js
 - 문제 내용 및 관련문제 crawling
 - 문제 태그 API
 - API를 통한 문제 해설

서비스의 가치

- 사용자의 학습 효율성
- 학습 과정의 연속성 유지
- 심화 학습과 학습 효과 향상

사용자의 학습 효율성

핵심

- 학습 흐름 유지와 시간 낭비의 최소화
- 시각적 예제로 즉각적인 학습 지원
- 인지적 부담 감소와 작업기억
 - 명확한 목표 설정과 집중력 향상

◦ <https://scienceon.kisti.re.kr/srch/selectPORsrchArticle.do?cn=DIKO000892751> 이주영(2007)

가치

- 문제 풀이에 집중할 수 있는 환경 조성 및 비효율 방지

심화 학습과 학습 효과 향상

핵심

- 체계적인 안내로 학습 유도
- 비효율적 학습 탈피와 깊이 있는 이해 지원
 - 인지적 부담 감소와 작업기억

◦ <https://scienceon.kisti.re.kr/srch/selectPORsrchArticle.do?cn=DIKO000892751> 김문수(1995)

가치

- 사고력과 코딩 실력 향상