단계별 모의고사 2회차 해설

문제		백준번호
1	트리 순회	BOJ22856
2	가장 긴 짝수 연속한 부분 수열 (small)	BOJ22857
3	원상 복구 (small)	BOJ22858
4	HTML 파싱	BOJ22859
5	폴더 정리 (small)	BOJ22860
6	폴더 정리 (large)	BOJ22861
7	가장 긴 짝수 연속한 부분 수열 (large)	BOJ22862
8	원상 복구 (large)	BOJ22863

1. 트리 순회

1. 트리 순회

- **해당 문제는 일반적인 중위 순회(Inorder)와 다릅니다.**
- 해당 문제는 주어진 대로 구현을 해도 풀리지만 관찰을 통해 더 간단한 풀이를 알려드리겠습니다.
- 유사 중위 순회의 끝은 일반적인 중위 순회를 할 때 맨 마지막에 방문하는 노드에서 멈추고 다시 위로 올라가지 않습니다.
- 이를 다시 말하면 맨 마지막 노드에서 출발하여 루트를 만날때까지 올라가지 않습니다. 올라갈 때 지나가야하는 간선을 제외한 나머지는 두 번씩 지나가야하지만 해당 간선은 단 한번만 지나간다는 것을 알 수 있습니다.
- 따라서, 간선의 개수 * 2 - 맨 마지막 노드에서 출발하여 루트까지 올라갈 때 지나가는 간선의 개수를 계산하면 됩니다.

1. 트리 순회

- 하지만, 맨 마지막 노드를 구하려면 중위 순회를 이용해야합니다. 여기서 조금 더 생각해보겠습니다.
- 중위 순회를 하여 맨 마지막에 있는 노드는 루트에서 가장 오른쪽에 존재하는 것을 알 수 있습니다. 다시 말해서, 루트에서 오른쪽 자식으로만 이동하면 맨 마지막 노드를 구할 수 있습니다.
- 그러면 전 슬라이드에서 말했던 식을 “간선의 개수 * 2 - 루트에서 오른쪽 자식으로만 이동하는 횟수”와 같이 더 간소화 시킬 수 있습니다.
- 간선의 개수는 항상 트리에서는 정점의 개수 -1 입니다.

2. 가장 긴 짝수 연속한 부분 수열 (small)

2. 가장 긴 짝수 연속한 부분 수열 (small)

- ✓ 해당 문제는 투 포인터 문제를 연습하신 분이라면 투 포인터부터 보이실겁니다. 투 포인터로 푸신 분이 계신다면 해설을 보기 전에 다이나믹 프로그래밍으로도 풀어보세요.
- ✓ 위에서 말했듯이 다이나믹 프로그래밍으로 풀이를 설명해보겠습니다. 가장 긴 짝수 연속한 부분 수열 (large)가 투 포인터로만 풀 수 있으니 투 포인터 풀이는 “가장 긴 짝수 연속한 부분 수열 (large)”에서 하도록 하겠습니다.

2. 가장 긴 짝수 연속한 부분 수열 (small)

- ✓ 다이나믹 프로그래밍 문제면 점화식을 한번 세워봐야합니다.
- ✓ DP$[i][j]$: i번 삭제하고 j번째까지 연속된 부분 수열 중 가장 긴 값

$$\text{DP}[i][j] = DP[i-1][j-1] + 1 \text{ (if } S[j] \text{ is odd)}$$
$$\text{DP}[i][j] = DP[i][j-1] \text{ (if } S[j] \text{ is even)}$$

- ✓ 엣지 케이스를 잘 생각하여 코딩에 옮기면 됩니다.

3. 원상 복구 (small)

3. 원상 복구 (small)

- 해당 문제는 지문에서 설명한대로 구현을 하면 되는 문제입니다.
- 근데, N, K 가 좀 커보이는데 과연 가능할까요?
- 카드를 총 K 번 섞었다는 것은 $K-1$ 번 섞었을때로 돌아가려면 각 위치에 있는 카드를 원래 섞는 방식의 반대로 돌려야합니다. 이때, 시간복잡도는 $O(N)$ 이 발생하게 되고 카드를 K 번 섞었기 때문에 총 시간복잡도는 $O(NK)$ 가 됨을 알 수 있습니다.
- 시간제한은 1초이고 N 은 최대 10^4, K 는 최대 10^3 이기 때문에 10^7 로 시간안에 충분히 돌아갈 수 있음을 미리 계산해볼 수 있습니다.

4. HTML 파싱

4. HTML 파싱

- HTML 형식으로 저장된 문서를 지문에서 주어진 방식으로 파싱을 하는 문제입니다.
- 해당 문제는 지문에서 설명하는대로 차근차근 구현하면 되는 문제로 파싱을 어떻게 진행을 하는지에 따라 구현이 조금 간단해지거나 복잡해집니다.
- title=”를 먼저 찾고 그 뒤에 를 찾아 로 열리는 곳과 으로 닫히는 구간을 찾습니다.
- 그 다음에 태그 div안에 있는 title의 정보를 파싱하고 div 태그 안에 있는 p 닫힌 태그를 기준으로 문장을 나눕니다.
- 그 이후에 p 태그로 감싸져 있는 안에 있는 다른 태그들을 없애면 문제를 해결할 수 있습니다.
- 이는 다른 사람들은 어떤식으로 구현을 했는지 참고하면 많은 도움이 될 것입니다.

5. 폴더 정리 (small)

5. 폴더 정리 (small)

- 주어진 파일들의 디렉토리를 트리로 잘 표현하고 쿼리마다 파일의 종류(파일의 이름이 같은 경우)와 파일의 총 개수를 구하는 문제입니다.
- 주어지는 폴더와 폴더 간의 관계와 폴더와 파일 관계를 트리로 구축한 후 트리를 탐색하며 문제를 해결할 수 있습니다.
- 맨 처음에 주어진 정보로 트리를 구축 후 한번 DFS를 돌아 모든 폴더에 대한 답을 계산한 후 주어지는 쿼리마다 바로 답을 출력하는 방식으로 풀어도 됩니다.
- 또는, 쿼리가 주어질 때마다 구축된 트리를 탐색하는 풀이도 가능합니다. 시간복잡도를 생각해보면 쿼리마다 답을 찾는 과정은 $O(QN)$ 만에 할 수 있기 때문입니다.

6. 폴더 정리 (large)

6. 폴더 정리 (large)

- Small 버전과 거의 유사하지만 해당 문제에는 하나가 추가된 것이 있습니다. 바로 폴더를 다른 폴더로 이동시키는 것입니다.
- 폴더 이동을 하는 경우를 제외하면 small 버전의 풀이와 동일합니다.
- 폴더를 이동하는 경우를 잘 생각해보면 옮길 폴더 안에 있는 폴더와 파일들의 부모만 바뀌는 것을 생각할 수 있습니다.
- 여기서 조심해야하는 것은 파일들을 옮길 때, 같은 종류의 파일이 있는 경우 두 번 계산하지 않도록 해야합니다.
- Java 기준 TreeSet, HashMap 등 다양한 자료구조를 활용하여 구현하면 됩니다.
- Small 버전과 Large 버전 풀이는 단순하지만 이를 구현하는건 상당히 어려운 편에 속하기 때문에 실수가 많이 발생할 수 있습니다.

7. 가장 긴 짝수 연속한 부분 수열 (large)

7. 가장 긴 짝수 연속한 부분 수열 (large)

- ✓ Small 버전과 입력 제한을 제외한 나머지는 같은 문제입니다.
- ✓ Small 버전에서 설명드린 다이나믹 프로그래밍으로 푸는 경우는 시간복잡도가 $O(NK)$ 이기 때문에 Large에서는 $O(NK)$ 인 경우는 시간제한 안에 돌 수 없음을 예측할 수 있습니다.
- ✓ 이 문제는 두 포인터라는 알고리즘을 이용하여 시간복잡도 $O(N+K)$ 만에 해결할 수 있습니다.
- ✓ 한쪽 시작점을 기준으로 최대한 오른쪽에 있는 연속된 원소들을 선택하는데 선택된 원소들이 다음을 만족하도록 선택하면 됩니다.
- ✓ "홀수인 수가 최대 K개 있고 나머지는 짝수인 수로 선택된 경우"

8. 원상 복구 (large)

8. 원상 복구 (large)

- 원상 복구 (small) 문제와 입력제한만 다르고 같은 문제입니다.
- N 의 최대가 10^6 이고 K 의 최대가 10^{15} 입니다. 이 제한을 보면 $O(NK)$ 는 시간제한안에 못돌아간다는 것을 예측할 수 있습니다.
- 어떻게 하면 $O(NK)$ 보다 더 빠르게 할 수 있을까요?
- 매번 카드를 셔플할 때 카드가 어떻게 바뀌었는지 그려보면 주기성을 보인다는 것을 알 수 있습니다.
- 셔플하는 과정을 그래프로 그려보면 한 개 이상의 사이클이 있는데 각 사이클의 길이를 구하여 K 로 나눈 나머지를 이용하면 처음 카드가 어떤 것인지 좀 더 빠르게 알 수 있습니다.

8. 원상 복구 (large)

- ✓ 존재하는 사이클들을 찾는데 걸리는 시간복잡도는 $O(N)$ 이고 주기성을 이용하여 초기 카드를 찾는 과정은 $O(1)$ 로 할 수 있습니다.
- ✓ 따라서, $O(NK)$ 보다 훨씬 빠른 $O(N)$ 으로 문제를 해결할 수 있습니다.
- ✓ N 은 10^6 밖에 안되기 때문에 시간제한 안에 충분히 돌 수 있음을 알 수 있습니다.