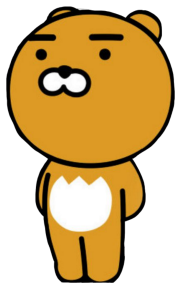
CS1 Mock Exam Editorial

2022 Spring

Automata

KSA of KAIST

2022.06.04



1. Stressed Students - 김연후

- 리스트의 값을 처음부터 확인하면서, 값이 0인 방을 찾습니다.

1. Stressed Students - 김연후

- 리스트의 값을 처음부터 확인하면서, 값이 0인 방을 찾습니다.
- 0인 방의 양 옆 방 번호의 합을 계산합니다. 이 값이 해당 방이 주는 스트레스의 크기입니다.

1. Stressed Students - 김연후

- 리스트의 값을 처음부터 확인하면서, 값이 0인 방을 찾습니다.
- 0인 방의 양 옆 방 번호의 합을 계산합니다. 이 값이 해당 방이 주는 스트레스의 크기입니다.
- 스트레스의 크기가 지금까지의 최대값보다도 크다면, 해당 방 번호를 저장합니다. 최대값도 갱신해줍니다.

1. Stressed Students - 김연후

- 리스트의 값을 처음부터 확인하면서, 값이 0인 방을 찾습니다.
- 0인 방의 양 옆 방 번호의 합을 계산합니다. 이 값이 해당 방이 주는 스트레스의 크기입니다.
- 스트레스의 크기가 지금까지의 최대값보다도 크다면, 해당 방 번호를 저장합니다. 최대값도 갱신해줍니다.
- 양 끝 방의 번호가 0인 경우 IndexError가 나지 않게끔 주의합시다.

2. Satisfy Me with Minimum Cost - 박예성

- 어떤 수를 1이 아닌 두 수의 곱으로 나눌 때, 두 수의 합이 원래 수보다 작아지는 것은 자명합니다.

2. Satisfy Me with Minimum Cost - 박예성

- 어떤 수를 1이 아닌 두 수의 곱으로 나눌 때, 두 수의 합이 원래 수보다 작아지는 것은 자명합니다.
- 따라서, 최소한의 합을 가진 수들의 곱으로 N을 나타내기 위해서는 N을 소인수분해해 주면 됩니다.

2. Satisfy Me with Minimum Cost - 박예성

- 어떤 수를 1이 아닌 두 수의 곱으로 나눌 때, 두 수의 합이 원래 수보다 작아지는 것은 자명합니다.
- 따라서, 최소한의 합을 가진 수들의 곱으로 N을 나타내기 위해서는 N을 소인수분해해 주면 됩니다.
- 2부터 시작해서 N의 소인수들을 모두 찾아주고, 그 합을 구해 주면 됩니다.

2. Satisfy Me with Minimum Cost - 박예성

- 어떤 수를 1이 아닌 두 수의 곱으로 나눌 때, 두 수의 합이 원래 수보다 작아지는 것은 자명합니다.
- 따라서, 최소한의 합을 가진 수들의 곱으로 N을 나타내기 위해서는 N을 소인수분해해 주면 됩니다.
- 2부터 시작해서 N의 소인수들을 모두 찾아주고, 그 합을 구해 주면 됩니다.
- 보통은 N을 나누는 수 i를 찾는다면 N을 i로 나누고, 결과값에 i를 더하는 방식으로 구현합니다.

2. Satisfy Me with Minimum Cost - 박예성

- 어떤 수를 1이 아닌 두 수의 곱으로 나눌 때, 두 수의 합이 원래 수보다 작아지는 것은 자명합니다.
- 따라서, 최소한의 합을 가진 수들의 곱으로 N을 나타내기 위해서는 N을 소인수분해해 주면 됩니다.
- 2부터 시작해서 N의 소인수들을 모두 찾아주고, 그 합을 구해 주면 됩니다.
- 보통은 N을 나누는 수 i를 찾는다면 N을 i로 나누고, 결과값에 i를 더하는 방식으로 구현합니다.
- N이 i에 나누어떨어지지 않을 때까지 반복해서 나눠야 함에 주의합시다.

2. Satisfy Me with Minimum Cost - 박예성

- 어떤 수를 1이 아닌 두 수의 곱으로 나눌 때, 두 수의 합이 원래 수보다 작아지는 것은 자명합니다.
- 따라서, 최소한의 합을 가진 수들의 곱으로 N을 나타내기 위해서는 N을 소인수분해해 주면 됩니다.
- 2부터 시작해서 N의 소인수들을 모두 찾아주고, 그 합을 구해 주면 됩니다.
- 보통은 N을 나누는 수 i를 찾는다면 N을 i로 나누고, 결과값에 i를 더하는 방식으로 구현합니다.
- N이 i에 나누어떨어지지 않을 때까지 반복해서 나눠야 함에 주의합시다.
- 원래는 문제에 스토리가 있었지만, 이해하기 어려워서 제거했습니다. 그러나 함수명과 문제명은 마땅히 수정할 것이 없어 그대로 두었습니다.

3. Max Consecutive Composite - 김동언

- 수업 시간에 배웠을 소수 판별법을 사용합니다. isPrime 함수를 그대로 활용해도 좋습니다.

3. Max Consecutive Composite - 김동언

- 수업 시간에 배웠을 소수 판별법을 사용합니다. isPrime 함수를 그대로 활용해도 좋습니다.
- 2부터 시작해서 N까지 각각 소수인지를 판별해 줍니다. 만약 소수가 아니라면 합성수일 것입니다.

3. Max Consecutive Composite - 김동언

- 수업 시간에 배웠을 소수 판별법을 사용합니다. isPrime 함수를 그대로 활용해도 좋습니다.
- 2부터 시작해서 N까지 각각 소수인지를 판별해 줍니다. 만약 소수가 아니라면 합성수일 것입니다.
- 만약 어떤 수가 합성수라면, 연속 몇 번째 합성수인지를 나타내는 변수에 1을 더해 줍니다.

3. Max Consecutive Composite - 김동언

- 수업 시간에 배웠을 소수 판별법을 사용합니다. isPrime 함수를 그대로 활용해도 좋습니다.
- 2부터 시작해서 N까지 각각 소수인지를 판별해 줍니다. 만약 소수가 아니라면 합성수일 것입니다.
- 만약 어떤 수가 합성수라면, 연속 몇 번째 합성수인지를 나타내는 변수에 1을 더해 줍니다.
- 합성수가 아니라면, 연속 몇 번째 합성수인지를 나타내는 변수를 0으로 초기화합니다.

3. Max Consecutive Composite - 김동언

- 수업 시간에 배웠을 소수 판별법을 사용합니다. isPrime 함수를 그대로 활용해도 좋습니다.
- 2부터 시작해서 N까지 각각 소수인지를 판별해 줍니다. 만약 소수가 아니라면 합성수일 것입니다.
- 만약 어떤 수가 합성수라면, 연속 몇 번째 합성수인지를 나타내는 변수에 1을 더해 줍니다.
- 합성수가 아니라면, 연속 몇 번째 합성수인지를 나타내는 변수를 0으로 초기화합니다.
- 그 변수가 최대값을 갱신할 때마다 따로 저장해 주고, 마지막에 저장한 최대값을 리턴합니다.

4. Number of Observable Skyscrapers - 이재환

- 모든 행에 대해 다음 연산을 시행해 줍니다.

4. Number of Observable Skyscrapers - 이재환

- 모든 행에 대해 다음 연산을 시행해 줍니다.
- 우선 볼 수 있는 건물 중 가장 높은 건물의 높이를 0으로 설정합니다.

4. Number of Observable Skyscrapers - 이재환

- 모든 행에 대해 다음 연산을 시행해 줍니다.
- 우선 볼 수 있는 건물 중 가장 높은 건물의 높이를 0으로 설정합니다.
- 각 건물에 대해, 만약 가장 높은 건물보다 그 건물이 높다면 해당 변수를 갱신하고 그 행에서 볼 수 있는 건물의 수에 1을 더해줍니다.

4. Number of Observable Skyscrapers - 이재환

- 모든 행에 대해 다음 연산을 시행해 줍니다.
- 우선 볼 수 있는 건물 중 가장 높은 건물의 높이를 0으로 설정합니다.
- 각 건물에 대해, 만약 가장 높은 건물보다 그 건물이 높다면 해당 변수를 갱신하고 그 행에서 볼 수 있는 건물의 수에 1을 더해줍니다.
- 그렇지 않다면, 그 건물은 볼 수 없다는 뜻이므로 건너뛰어 줍니다.

4. Number of Observable Skyscrapers - 이재환

- 모든 행에 대해 다음 연산을 시행해 줍니다.
- 우선 볼 수 있는 건물 중 가장 높은 건물의 높이를 0으로 설정합니다.
- 각 건물에 대해, 만약 가장 높은 건물보다 그 건물이 높다면 해당 변수를 갱신하고 그 행에서 볼 수 있는 건물의 수에 1을 더해줍니다.
- 그렇지 않다면, 그 건물은 볼 수 없다는 뜻이므로 건너뛰어 줍니다.
- 모든 행에 대한 연산을 마치면, 결과값들을 리턴합니다.

5. Pop Balloon - 김수기

- 문제 조건을 그대로 구현하면 되지만, 실수가 나오기 쉬운 문제입니다.

5. Pop Balloon - 김수기

- 문제 조건을 그대로 구현하면 되지만, 실수가 나오기 쉬운 문제입니다.
- 모든 풍선이 터질 때까지 아래 시행을 반복합니다.

5. Pop Balloon - 김수기

- 문제 조건을 그대로 구현하면 되지만, 실수가 나오기 쉬운 문제입니다.
- 모든 풍선이 터질 때까지 아래 시행을 반복합니다.
- 각 행 또는 열에 대해, 풍선이 있는지를 확인해 주고, 있다면 가장 앞의 것을 없앱니다.

5. Pop Balloon - 김수기

- 문제 조건을 그대로 구현하면 되지만, 실수가 나오기 쉬운 문제입니다.
- 모든 풍선이 터질 때까지 아래 시행을 반복합니다.
- 각 행 또는 열에 대해, 풍선이 있는지를 확인해 주고, 있다면 가장 앞의 것을 없앱니다.
- 만약 모든 행 또는 열에 풍선이 없었다면, 모든 풍선이 터진 것이므로 종료합니다. 처음에 풍선의 개수를 저장해 놓고 풍선을 터트릴 때마다 1씩 뺀 뒤, 0일 때 종료하는 식으로 구현하면 더 편리합니다.

5. Pop Balloon - 김수기

- 문제 조건을 그대로 구현하면 되지만, 실수가 나오기 쉬운 문제입니다.
- 모든 풍선이 터질 때까지 아래 시행을 반복합니다.
- 각 행 또는 열에 대해, 풍선이 있는지를 확인해 주고, 있다면 가장 앞의 것을 없앱니다.
- 만약 모든 행 또는 열에 풍선이 없었다면, 모든 풍선이 터진 것이므로 종료합니다. 처음에 풍선의 개수를 저장해 놓고 풍선을 터트릴 때마다 1씩 뺀 뒤, 0일 때 종료하는 식으로 구현하면 더 편리합니다.
- 그때까지 총 몇 번의 시행을 거쳤는지를 확인해 주고, 리턴해 주면 됩니다.

6. Report Card - 박유민

- 직전 문제와는 반대로, 구현은 단순하지만 아이디어가 필요합니다.

6. Report Card - 박유민

- 직전 문제와는 반대로, 구현은 단순하지만 아이디어가 필요합니다.
- 힌트에서 제시한 대로, 연속된 F들의 개수는 어떤 연산을 사용하던 1개씩만 없앨 수 있습니다.

6. Report Card - 박유민

- 직전 문제와는 반대로, 구현은 단순하지만 아이디어가 필요합니다.
- 힌트에서 제시한 대로, 연속된 F들의 개수는 어떤 연산을 사용하던 1개씩만 없앨 수 있습니다.
- 따라서, 연속된 F들의 개수만큼 두 연산 중 소요 시간이 더 짧은 것만 반복해 주면 됩니다.

6. Report Card - 박유민

- 직전 문제와는 반대로, 구현은 단순하지만 아이디어가 필요합니다.
- 힌트에서 제시한 대로, 연속된 F들의 개수는 어떤 연산을 사용하던 1개씩만 없앨 수 있습니다.
- 따라서, 연속된 F들의 개수만큼 두 연산 중 소요 시간이 더 짧은 것만 반복해 주면 됩니다.
- 단, 마지막 한 번은 반드시 연산 A를 사용해야 하는 점에 주의합시다.

6. Report Card - 박유민

- 직전 문제와는 반대로, 구현은 단순하지만 아이디어가 필요합니다.
- 힌트에서 제시한 대로, 연속된 F들의 개수는 어떤 연산을 사용하던 1개씩만 없앨 수 있습니다.
- 따라서, 연속된 F들의 개수만큼 두 연산 중 소요 시간이 더 짧은 것만 반복해 주면 됩니다.
- 단, 마지막 한 번은 반드시 연산 A를 사용해야 하는 점에 주의합시다.
- 그러므로 정답은 ((연속된 F의 개수 - 1) * min(A,B)) + A라는 식을 통해 구할 수 있습니다. 단, F가 하나도 없다면 0을 리턴해야 합니다.

7. QXQ2RUN1 - 박기윤

- 문제 제목은 $\mathbb{Q} \times \mathbb{Q} \to \mathbb{R} \cup \{\text{None}\}$에서 왔습니다.

7. QXQ2RUN1 - 박기윤

- 문제 제목은 $\mathbb{Q} \times \mathbb{Q} \to \mathbb{R} \cup \{\text{None}\}$에서 왔습니다.
- 문제에서 제시된 함수 f는 다소 복잡해 보일 수 있지만 간단한 분석을 통해 친숙한 함수라는 걸 알아차릴 수 있을 것입니다.

7. QXQ2RUN1 - 박기윤

- 문제 제목은 $\mathbb{Q} \times \mathbb{Q} \to \mathbb{R} \cup \{\text{None}\}$에서 왔습니다.
- 문제에서 제시된 함수 f는 다소 복잡해 보일 수 있지만 간단한 분석을 통해 친숙한 함수라는 걸 알아차릴 수 있을 것입니다.
- 문제에 제시된 대로 조건 분기를 활용해 결과값을 찾아냅니다.

7. QXQ2RUN1 - 박기윤

- 문제 제목은 $\mathbb{Q} \times \mathbb{Q} \to \mathbb{R} \cup \{\text{None}\}$에서 왔습니다.
- 문제에서 제시된 함수 f는 다소 복잡해 보일 수 있지만 간단한 분석을 통해 친숙한 함수라는 걸 알아차릴 수 있을 것입니다.
- 문제에 제시된 대로 조건 분기를 활용해 결과값을 찾아냅니다.
- 두 분수를 먼저 약분하고 시작해야 함에 주의합시다. 유클리드 호제법을 사용해도 되고 다른 방법을 사용해도 됩니다.

7. QXQ2RUN1 - 박기윤

- 문제 제목은 $\mathbb{Q} \times \mathbb{Q} \to \mathbb{R} \cup \{\text{None}\}$에서 왔습니다.
- 문제에서 제시된 함수 f는 다소 복잡해 보일 수 있지만 간단한 분석을 통해 친숙한 함수라는 걸 알아차릴 수 있을 것입니다.
- 문제에 제시된 대로 조건 분기를 활용해 결과값을 찾아냅니다.
- 두 분수를 먼저 약분하고 시작해야 함에 주의합시다. 유클리드 호제법을 사용해도 되고 다른 방법을 사용해도 됩니다.
- 경우의 수가 약간 많은 편이므로, `if`문을 대량으로 쓰다가 조건들의 우선 순위가 꼬이거나 몇 가지 경우를 빠뜨리는 일이 없도록 조심해야 합니다.

7. QXQ2RUN1 - 박기윤

- 문제 제목은 $\mathbb{Q} \times \mathbb{Q} \to \mathbb{R} \cup \{\text{None}\}$에서 왔습니다.
- 문제에서 제시된 함수 f는 다소 복잡해 보일 수 있지만 간단한 분석을 통해 친숙한 함수라는 걸 알아차릴 수 있을 것입니다.
- 문제에 제시된 대로 조건 분기를 활용해 결과값을 찾아냅니다.
- 두 분수를 먼저 약분하고 시작해야 함에 주의합시다. 유클리드 호제법을 사용해도 되고 다른 방법을 사용해도 됩니다.
- 경우의 수가 약간 많은 편이므로, `if`문을 대량으로 쓰다가 조건들의 우선 순위가 꼬이거나 몇 가지 경우를 빠뜨리는 일이 없도록 조심해야 합니다.
- 조금 실수해도 부분점수를 얻을 수 있을 수도 있습니다.

7. QXQ2RUN1 - 박기윤

- 문제 제목은 $\mathbb{Q} \times \mathbb{Q} \to \mathbb{R} \cup \{\text{None}\}$에서 왔습니다.
- 문제에서 제시된 함수 f는 다소 복잡해 보일 수 있지만 간단한 분석을 통해 친숙한 함수라는 걸 알아차릴 수 있을 것입니다.
- 문제에 제시된 대로 조건 분기를 활용해 결과값을 찾아냅니다.
- 두 분수를 먼저 약분하고 시작해야 함에 주의합시다. 유클리드 호제법을 사용해도 되고 다른 방법을 사용해도 됩니다.
- 경우의 수가 약간 많은 편이므로, `if`문을 대량으로 쓰다가 조건들의 우선 순위가 꼬이거나 몇 가지 경우를 빠뜨리는 일이 없도록 조심해야 합니다.
- 조금 실수해도 부분점수를 얻을 수 있을 수도 있습니다.
- 자세한 조건 분기는 모범 답안을 확인해 주세요.

Thanks!