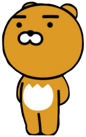
CS2 Mock Exam Editorial

2022 Fall

Automata

KSA of KAIST

2022.12



1. Basic Problems - 박유민

- 1번부터 3번까지의 문제는 python 기본 문법을 숙지하고 있는지 확인하는 문제였습니다. 정수를 리턴해야 하는 문제에 소수를 리턴하지 않도록 주의합시다.

1. Basic Problems - 박유민

- 1번부터 3번까지의 문제는 python 기본 문법을 숙지하고 있는지 확인하는 문제였습니다. 정수를 리턴해야 하는 문제에 소수를 리턴하지 않도록 주의합시다.
- 4번 문제는, 힌트대로 A와 B를 각각 리스트로 바꿔준 뒤 sort 함수를 사용해 정렬한 뒤 같은지 비교해주면 간단하게 구현이 가능합니다.

1. Basic Problems - 박유민

- 1번부터 3번까지의 문제는 python 기본 문법을 숙지하고 있는지 확인하는 문제였습니다. 정수를 리턴해야 하는 문제에 소수를 리턴하지 않도록 주의합시다.
- 4번 문제는, 힌트대로 A와 B를 각각 리스트로 바꿔준 뒤 sort 함수를 사용해 정렬한 뒤 같은지 비교해주면 간단하게 구현이 가능합니다.
- A의 각 문자가 B에 존재하는지만 확인한다면, $A=$"aab",$B=$"abb"와 같은 반례가 발생합니다.

1. Basic Problems - 박유민

- 1번부터 3번까지의 문제는 python 기본 문법을 숙지하고 있는지 확인하는 문제였습니다. 정수를 리턴해야 하는 문제에 소수를 리턴하지 않도록 주의합시다.
- 4번 문제는, 힌트대로 A와 B를 각각 리스트로 바꿔준 뒤 sort 함수를 사용해 정렬한 뒤 같은지 비교해주면 간단하게 구현이 가능합니다.
- A의 각 문자가 B에 존재하는지만 확인한다면, $A=$"aab",$B=$"abb"와 같은 반례가 발생합니다.
- 5번 문제는, N의 각 글자를 K번씩 next_alphabet 함수를 돌려준 뒤, 결과 문자열에 추가해 주면 됩니다.

2. Chocolate Cake - 박유민

- $1 < i, j, k < N - 1$ 를 만족하는 $P[i][j][k]$를 모두 확인하면 됩니다.

2. Chocolate Cake - 박유민

- $1 < i, j, k < N - 1$ 를 만족하는 $P[i][j][k]$를 모두 확인하면 됩니다.
- $P[i \pm 1][j][k]$, $P[i][j \pm 1][k]$, $P[i][j][k \pm 1]$이 모두 1이라면 $P[i][j][k] = 1$일 때 조건을 만족하는 칸입니다.

2. Chocolate Cake - 박유민

- $1 < i, j, k < N - 1$ 를 만족하는 $P[i][j][k]$를 모두 확인하면 됩니다.
- $P[i \pm 1][j][k]$, $P[i][j \pm 1][k]$, $P[i][j][k \pm 1]$이 모두 1이라면 $P[i][j][k] = 1$일 때 조건을 만족하는 칸입니다.
- 조건을 만족하는 칸의 개수를 구해서 반환하면 됩니다.

3. Ribbons (Easy) - eric00513

- 짧게 요약하면, $|X_i - X_j| \leq L_i + L_j$와 $C_i \neq C_j$를 만족하는 쌍 (i, j)를 반환하면 됩니다.

3. Ribbons (Easy) - eric00513

- 짧게 요약하면, $|X_i - X_j| \leq L_i + L_j$와 $C_i \neq C_j$를 만족하는 쌍 (i, j)를 반환하면 됩니다.
- for문을 사용하여 가능한 모든 쌍 (i, j) $(0 \leq i, j < N)$에 대해 확인해 보다가 조건을 만족하는 쌍이 나오면 바로 반환합니다.

3. Ribbons (Easy) - eric00513

- 짧게 요약하면, $|X_i - X_j| \leq L_i + L_j$와 $C_i \neq C_j$를 만족하는 쌍 (i, j)를 반환하면 됩니다.
- for문을 사용하여 가능한 모든 쌍 (i, j) $(0 \leq i, j < N)$에 대해 확인해 보다가 조건을 만족하는 쌍이 나오면 바로 반환합니다.
- 어떤 수 a의 절댓값 $|a|$는 `abs(a)` 함수나 if문을 이용하면 구할 수 있습니다.

3. Ribbons (Easy) - eric00513

- 짧게 요약하면, $|X_i - X_j| \le L_i + L_j$와 $C_i \ne C_j$를 만족하는 쌍 (i, j)를 반환하면 됩니다.
- for문을 사용하여 가능한 모든 쌍 (i, j) $(0 \le i, j < N)$에 대해 확인해 보다가 조건을 만족하는 쌍이 나오면 바로 반환합니다.
- 어떤 수 a의 절댓값 $|a|$는 `abs(a)` 함수나 if문을 이용하면 구할 수 있습니다.
- 이때, 모든 쌍이 $i < j$를 만족하도록 순회하면, 문제 조건에 의해 $X_i < X_j$이므로 $X_i - X_j$의 절댓값은 $X_j - X_i$가 되기 때문에 절댓값을 고려할 필요가 없습니다.

4. Joker - 박유민

- 총 52장의 카드 중, 원래 가지고 있던 4장을 제외한 48장의 카드를 확인합니다.

4. Joker - 박유민

- 총 52장의 카드 중, 원래 가지고 있던 4장을 제외한 48장의 카드를 확인합니다.
- 주어진 핸드 판별 함수들을 사용해 48가지의 경우 중 가장 강한 핸드를 찾아냅니다.

4. Joker - 박유민

- 총 52장의 카드 중, 원래 가지고 있던 4장을 제외한 48장의 카드를 확인합니다.
- 주어진 핸드 판별 함수들을 사용해 48가지의 경우 중 가장 강한 핸드를 찾아냅니다.
- 더 강한 핸드를 찾을 때마다 결과값을 갱신해주고, 탐색이 끝났을 때의 결과값을 반환하면 간단합니다.

5. Mentoring Matching - 송은하(runnie0427)

- 전형적인 Stable Matching 문제입니다.

5. Mentoring Matching - 송은하(runnie0427)

- 전형적인 Stable Matching 문제입니다.
- KSA 학생들의 선호도는 바로 주어집니다.

5. Mentoring Matching - 송은하(runnie0427)

- 전형적인 Stable Matching 문제입니다.
- KSA 학생들의 선호도는 바로 주어집니다.
- 까다로운 부분은 KAIST 학생들의 선호도인데, 여러 가지 방법으로 선호하는 순서대로 list를 만드는 걸 구현할 수 있습니다.

5. Mentoring Matching - 송은하(runnie0427)

- 전형적인 Stable Matching 문제입니다.
- KSA 학생들의 선호도는 바로 주어집니다.
- 까다로운 부분은 KAIST 학생들의 선호도인데, 여러 가지 방법으로 선호하는 순서대로 list를 만드는 걸 구현할 수 있습니다.
- 예시 코드는 가장 선호하지 않는 KSA 학생(=자신과 학번이 같은 학생)을 시작으로 차례대로 리스트에 넣고, 그 리스트를 뒤집는 구현을 통해서 KAIST 학생들의 선호도도 만들었습니다.

5. Mentoring Matching - 송은하(runnie0427)

- 전형적인 Stable Matching 문제입니다.
- KSA 학생들의 선호도는 바로 주어집니다.
- 까다로운 부분은 KAIST 학생들의 선호도인데, 여러 가지 방법으로 선호하는 순서대로 list를 만드는 걸 구현할 수 있습니다.
- 예시 코드는 가장 선호하지 않는 KSA 학생(=자신과 학번이 같은 학생)을 시작으로 차례대로 리스트에 넣고, 그 리스트를 뒤집는 구현을 통해서 KAIST 학생들의 선호도도 만들었습니다.
- 그 이후로 Perfect Matching을 찾아주면 됩니다. N이 20 이하이기 때문에 제한 시간은 아주 여유롭습니다.

Bonus 1. Diverse Array - 박유민

- M이 $min(R - L + 1)$ 보다 커질 수 없음은 당연합니다.

Bonus 1. Diverse Array - 박유민

- M이 $min(R - L + 1)$ 보다 커질 수 없음은 당연합니다.
- 따라서, $M = min(R - L + 1)$일 때가 존재한다면 그것이 최선의 경우 중 하나가 됩니다.

Bonus 1. Diverse Array - 박유민

- M이 $min(R - L + 1)$ 보다 커질 수 없음은 당연합니다.
- 따라서, $M = min(R - L + 1)$일 때가 존재한다면 그것이 최선의 경우 중 하나가 됩니다.
- 이를 만족하는 수열은 매우 많습니다. $1, 2, \cdots, M$을 계속 반복했을 때, 즉 $X[i] = (i$를 M으로 나눈 나머지$) + 1$로 정했을 때 나오는 수열도 간단한 정답 중 하나입니다.

Bonus 2. Prevent Cheating - 박유민

- 가장 학생 수가 많은 반을 X이라고 합시다. 이들을 최대한 멀리 떨어뜨려야 합니다.

Bonus 2. Prevent Cheating - 박유민

- 가장 학생 수가 많은 반을 X이라고 합시다. 이들을 최대한 멀리 떨어뜨려야 합니다.
- 그보다 적은 학생 수를 가진 반의 학생들은 X반 학생 사이사이에 들어갔을 때의 간격이 X반 학생들 사이 간격보다 항상 같거나 크기 때문입니다.

Bonus 2. Prevent Cheating - 박유민

- 가장 학생 수가 많은 반을 X이라고 합시다. 이들을 최대한 멀리 떨어뜨려야 합니다.
- 그보다 적은 학생 수를 가진 반의 학생들은 X반 학생 사이사이에 들어갔을 때의 간격이 X반 학생들 사이 간격보다 항상 같거나 크기 때문입니다.
- 즉, X반 학생들이 최대한 멀리 떨어지게끔 배치한 후, 그 사이사이 간격마다 다른 반의 학생들을 최대 한 명씩 집어넣으면 됩니다.

Bonus 2. Prevent Cheating - 박유민

- 가장 학생 수가 많은 반을 X이라고 합시다. 이들을 최대한 멀리 떨어뜨려야 합니다.
- 그보다 적은 학생 수를 가진 반의 학생들은 X반 학생 사이사이에 들어갔을 때의 간격이 X반 학생들 사이 간격보다 항상 같거나 크기 때문입니다.
- 즉, X반 학생들이 최대한 멀리 떨어지게끔 배치한 후, 그 사이사이 간격마다 다른 반의 학생들을 최대 한 명씩 집어넣으면 됩니다.
- 학생 수가 X반보다 한 명 적은 반의 경우 배치 방법에 특히 주의해야 합니다.

Bonus 2. Prevent Cheating - 박유민

- 가장 학생 수가 많은 반을 X이라고 합시다. 이들을 최대한 멀리 떨어뜨려야 합니다.
- 그보다 적은 학생 수를 가진 반의 학생들은 X반 학생 사이사이에 들어갔을 때의 간격이 X반 학생들 사이 간격보다 항상 같거나 크기 때문입니다.
- 즉, X반 학생들이 최대한 멀리 떨어지게끔 배치한 후, 그 사이사이 간격마다 다른 반의 학생들을 최대 한 명씩 집어넣으면 됩니다.
- 학생 수가 X반보다 한 명 적은 반의 경우 배치 방법에 특히 주의해야 합니다.
- 학생 수가 최대인 반이 여러 개일 때는, $X_1X_2X_3 \cdots X_1X_2X_3 \cdots\cdots X_1X_2X_3$ 와 같이 배치하면 항상 (X반의 학생 수 $-$ 1)개의 간격을 만들 수 있습니다.

Bonus 3. Ribbons (Hard) I - eric00513

- Easy 버전의 풀이는 N으로 10^6과 같이 매우 큰 수가 주어지면 실행 시간이 제한을 초과합니다.

Bonus 3. Ribbons (Hard) I - eric00513

- Easy 버전의 풀이는 N으로 10^6과 같이 매우 큰 수가 주어지면 실행 시간이 제한을 초과합니다.
- 이는 (대략) N^2에 비례하는 개수의 쌍을 일일이 확인하기 때문입니다. 그러면 더 빠른 풀이를 생각해봅시다.

Bonus 3. Ribbons (Hard) I - eric00513

- Easy 버전의 풀이는 N으로 10^6과 같이 매우 큰 수가 주어지면 실행 시간이 제한을 초과합니다.
- 이는 (대략) N^2에 비례하는 개수의 쌍을 일일이 확인하기 때문입니다. 그러면 더 빠른 풀이를 생각해봅시다.
- 문제를 재해석하면, N개의 구간 $A_0[X_0 - L_0, X_0 + L_0], A_1[X_1 - L_1, X_1 + L_1], \cdots, A_{N-1}[X_{N-1} - L_{N-1}, X_{N-1} + L_{N-1}]$ 중 교집합이 $\varnothing$이 아닌 두 구간을 고르면 됩니다.

Bonus 3. Ribbons (Hard) I - eric00513

- Easy 버전의 풀이는 N으로 10^6과 같이 매우 큰 수가 주어지면 실행 시간이 제한을 초과합니다.
- 이는 (대략) N^2에 비례하는 개수의 쌍을 일일이 확인하기 때문입니다. 그러면 더 빠른 풀이를 생각해봅시다.
- 문제를 재해석하면, N개의 구간 $A_0[X_0 - L_0, X_0 + L_0], A_1[X_1 - L_1, X_1 + L_1], \cdots, A_{N-1}[X_{N-1} - L_{N-1}, X_{N-1} + L_{N-1}]$ 중 교집합이 $\varnothing$이 아닌 두 구간을 고르면 됩니다.
- 두 구간 $A_i[X_i - L_i, X_i + L_i], A_j[X_j - L_j, X_j + L_j]$ $(i < j)$가 $\varnothing$이 아닌 교집합을 가지기 위해서는 $X_i + L_i \geq X_j - L_j$를 만족해야 합니다(지문에서 언급된 부등식과 동일한 부등식입니다).

Bonus 3. Ribbons (Hard) I - eric00513

- Easy 버전의 풀이는 N으로 10^6과 같이 매우 큰 수가 주어지면 실행 시간이 제한을 초과합니다.
- 이는 (대략) N^2에 비례하는 개수의 쌍을 일일이 확인하기 때문입니다. 그러면 더 빠른 풀이를 생각해봅시다.
- 문제를 재해석하면, N개의 구간 $A_0[X_0 - L_0, X_0 + L_0], A_1[X_1 - L_1, X_1 + L_1], \cdots, A_{N-1}[X_{N-1} - L_{N-1}, X_{N-1} + L_{N-1}]$ 중 교집합이 $\varnothing$이 아닌 두 구간을 고르면 됩니다.
- 두 구간 $A_i[X_i - L_i, X_i + L_i]$, $A_j[X_j - L_j, X_j + L_j]$ $(i < j)$가 $\varnothing$이 아닌 교집합을 가지기 위해서는 $X_i + L_i \geq X_j - L_j$를 만족해야 합니다(지문에서 언급된 부등식과 동일한 부등식입니다).
- 이제 본격적으로 쌍 (i, j)를 어떻게 구하는지 알아봅시다.

Bonus 3. Ribbons (Hard) II - eric00513

- j를 고정하고 A_j와의 교집합이 $\varnothing$이 아닌 $A_i(0 \leq i < j)$를 구하면 됩니다.

Bonus 3. Ribbons (Hard) II - eric00513

- j를 고정하고 A_j와의 교집합이 $\varnothing$이 아닌 $A_i (0 \leq i < j)$를 구하면 됩니다.
- A_j는 변하지 않으므로 $A_0, A_1, \cdots, A_{j-1}$ 중 오른쪽 끝점, 즉 $X_i + L_i$가 가장 큰 A_i를 고르는 것이 유리합니다.

Bonus 3. Ribbons (Hard) II - eric00513

- j를 고정하고 A_j와의 교집합이 $\varnothing$이 아닌 $A_i (0 \le i < j)$를 구하면 됩니다.
- A_j는 변하지 않으므로 $A_0, A_1, \cdots, A_{j-1}$ 중 오른쪽 끝점, 즉 $X_i + L_i$가 가장 큰 A_i를 고르는 것이 유리합니다.
- 따라서 $i = 0, 1, \cdots, j-1$에서의 $X_i + L_i$의 최댓값이 $X_j - L_j$ 보다 크면 답을 반환하고, 만약 아니라면 다음 j로 넘어갑니다.

Bonus 3. Ribbons (Hard) II - eric00513

- j를 고정하고 A_j와의 교집합이 $\varnothing$이 아닌 $A_i (0 \leq i < j)$를 구하면 됩니다.
- A_j는 변하지 않으므로 $A_0, A_1, \cdots, A_{j-1}$ 중 오른쪽 끝점, 즉 $X_i + L_i$가 가장 큰 A_i를 고르는 것이 유리합니다.
- 따라서 $i = 0, 1, \cdots, j - 1$에서의 $X_i + L_i$의 최댓값이 $X_j - L_j$ 보다 크면 답을 반환하고, 만약 아니라면 다음 j로 넘어갑니다.
- 이때 최댓값은 계속 $X_i + L_i$ 값을 최댓값 변수에 누적하는 방식으로 구하고, 해당 최댓값의 i값(인덱스)도 같이 저장합니다.

Bonus 3. Ribbons (Hard) II - eric00513

- j를 고정하고 A_j와의 교집합이 $\varnothing$이 아닌 $A_i (0 \leq i < j)$를 구하면 됩니다.
- A_j는 변하지 않으므로 $A_0, A_1, \cdots, A_{j-1}$ 중 오른쪽 끝점, 즉 $X_i + L_i$가 가장 큰 A_i를 고르는 것이 유리합니다.
- 따라서 $i = 0, 1, \cdots, j-1$에서의 $X_i + L_i$의 최댓값이 $X_j - L_j$보다 크면 답을 반환하고, 만약 아니라면 다음 j로 넘어갑니다.
- 이때 최댓값은 계속 $X_i + L_i$ 값을 최댓값 변수에 누적하는 방식으로 구하고, 해당 최댓값의 i값(인덱스)도 같이 저장합니다.
- 아직 색깔을 고려하지 않았습니다. 이는 A_j와 앞에서 찾은 A_i가 다른 색깔이라는 보장이 없으므로 각 색깔마다 최댓값과 인덱스를 따로 저장하면 해결됩니다.

Bonus 3. Ribbons (Hard) II - eric00513

- j를 고정하고 A_j와의 교집합이 $\varnothing$이 아닌 $A_i(0 \le i < j)$를 구하면 됩니다.
- A_j는 변하지 않으므로 $A_0, A_1, \cdots, A_{j-1}$ 중 오른쪽 끝점, 즉 $X_i + L_i$가 가장 큰 A_i를 고르는 것이 유리합니다.
- 따라서 $i = 0, 1, \cdots, j-1$에서의 $X_i + L_i$의 최댓값이 $X_j - L_j$보다 크면 답을 반환하고, 만약 아니라면 다음 j로 넘어갑니다.
- 이때 최댓값은 계속 $X_i + L_i$ 값을 최댓값 변수에 누적하는 방식으로 구하고, 해당 최댓값의 i값(인덱스)도 같이 저장합니다.
- 아직 색깔을 고려하지 않았습니다. 이는 A_j와 앞에서 찾은 A_i가 다른 색깔이라는 보장이 없으므로 각 색깔마다 최댓값과 인덱스를 따로 저장하면 해결됩니다.
- 위 풀이에서는 쌍의 후보 개수가 (대략) N에 비례하므로 적은 시간으로 쌍 (i, j)를 구할 수 있습니다.

Thanks!