KUPC 2022 문제 풀이

Official Solutions

by

KUPC 2022 출제진

2022년 12월 3일

KUPC 2022

문제		의도한 난이도	출제자
A	슬픈 건구스	**Beginner**	이동훈 aru0504
B	만쥬의 식사	**Easy**	김명기 riroan
C	비숍 여행	**Easy**	김명기 riroan
D	시험자리 배정하기	**Medium**	김명기 riroan
E	즐거운 XOR	**Medium**	김명기 riroan
F	킥보드로 등교하기	**Medium**	김태현 kth990303
G	보물찾기 2	**Hard**	이동훈 aru0504
H	볼링장 아르바이트	**Hard**	이승엽 delena0702
I	문자열 게임	**Hard**	이승엽 delena0702
J	압도적 XOR수	**Challenging**	김태현 kth990303

A. 슬픈 건구스

`string`, `implementation`
출제진 의도 – **Beginner**

- ✓ 출제자: 이동훈aru0504

건구스는 항상 goo...se와 같이 웁니다.

- ✓ 첫 글자가 g인지 확인합니다.
- ✓ 마지막 두 글자가 se인지 확인합니다.
- ✓ 첫 글자와 마지막 글자를 제외한 나머지 글자가 2개 이상의 o로 이루어져있는지 확인합니다.
- ✓ 단순히 문자의 개수를 세는 것으로는 통과할 수 없습니다.

B. 만쥬의 식사

greedy
출제진 의도 – **Easy**

✓ 출제자: 김명기riroan

- ✓ 밥그릇에 들어있는 츄르의 개수는 늘어나지 않습니다.
- ✓ 그래서 모든 츄르가 같아지기 위해 가장 작은 값으로 맞춰야 합니다.
- ✓ 츄르의 최솟값을 m 이라고 하면 $\sum_{i=1}^{N}(a_i - m)$ 의 값을 구하면 정답이 됩니다.
- ✓ 총 시간복잡도는 $\mathcal{O}(N)$ 입니다.

C. 비숍 여행

`implementation`, `math`

출제진 의도 – **Easy**

✓ 출제자: 김명기 `riroan`

- ✓ 비숍은 한번 이동할 때마다 x좌표와 y좌표의 홀짝이 각각 변합니다.
- ✓ 예를들어 비숍이 (짝수, 홀수)좌표에 있다면 한번 이동했을 때 (홀수, 짝수)좌표로 이동합니다.
- ✓ 하지만 이동을 하더라도 **x좌표 + y좌표**의 홀짝은 변하지 않습니다.
- ✓ 따라서 비숍의 시작좌표 합의 홀짝과 같은 동전좌표 합의 홀짝인 개수를 구하면 됩니다.
- ✓ 총 시간복잡도는 $\mathcal{O}(N)$ 입니다.

D. 시험자리 배정하기

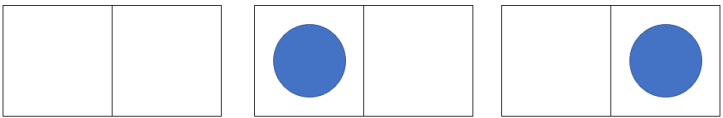
`math`, `dynamic_programming`, `combinatorics`
출제진 의도 – **Medium**

- ✓ 출제자: 김명기 riroan

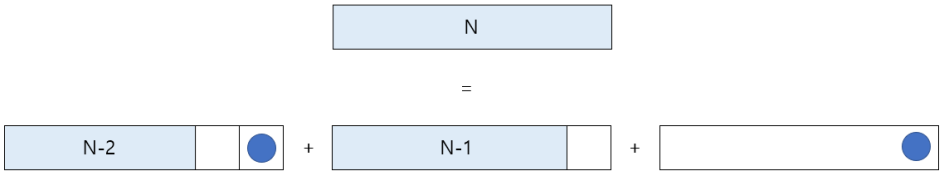
D. 시험자리 배정하기

- ✓ 이 문제는 dp 풀이와 조합식을 이용한 풀이가 있습니다.
- ✓ 여기에서는 dp 풀이를 소개하겠습니다.

- ✓ 문제의 정답을 $A(n)$으로 정의하겠습니다.
- ✓ $A(1) = 1, A(2) = 2$임을 직관적으로 알 수 있습니다.
- ✓ $k \geq 3$에서 $A(k)$의 마지막 자리 2개만 봤을 때 가능한 경우는 아래 3가지 뿐입니다.



D. 시험자리 배정하기



- ✓ 위와 같이 구성하면 $A(n)$의 모든 경우를 구할 수 있습니다.
- ✓ 따라서 점화식은 $A(n) = A(n-1) + A(n-2) + 1$이 됩니다.
- ✓ 총 시간복잡도는 $\mathcal{O}(n)$입니다.

E. 즐거운 XOR

`pigeonhole_principle`, `brute_force`

출제진 의도 – **Medium**

- 출제자: 김명기 riroan

- ✓ a_i 의 제한이 작음에 주목합니다.
- ✓ 비둘기집 원리에 의해 배열 크기가 100이 넘어가면 중복되는 원소가 적어도 하나 존재합니다.
- ✓ 원소의 개수를 세는 배열을 A라고 정의하겠습니다.

- ✓ 이제 $(0,0,0),(0,0,1),\cdots,(100,100,100)$ 까지 모두 탐색을 하면 됩니다.
- ✓ (a,b,c) 인 경우 $A[a]=A[a]-1, A[b]=A[b]-1, A[c]=A[c]-1$ 를 했을 때 배열 A 에 음수가 없는 경우에만 탐색합니다.
- ✓ 탐색이 완료된 후 가장 큰 xor값을 출력하면 됩니다.
- ✓ 총 시간복잡도는 $\mathcal{O}\left(100^3\right)$ 입니다.

F. 킥보드로 등교하기

`binary_search`, `parametric_search`
출제진 의도 – **Medium**

✓ 출제자: 김태현 kth990303

- ✓ 가장 용량이 작은, 즉 가장 싼 킥보드를 사기 위해서는 방문하는 충전소 간 거리가 짧을수록 유리합니다.
- ✓ K 가 작다면 모든 경우를 완전탐색하거나 다이나믹 프로그래밍으로 최적의 해를 구할 수 있지만, 충전 가능 횟수가 매우 큽니다.
- ✓ 어떻게 하면 좋을까요?

- 킥보드의 용량이 클수록 충전소에 방문해야 하는 횟수는 감소하게 됩니다.
- 다시 말해 킥보드의 용량이 특정 값 이상이면 조건에 만족하게 되며, 단조함수 형태로 증감하므로 킥보드의 용량을 매개변수로 두어 접근할 수 있습니다.
- 이 때, 킥보드의 용량 L에 대해 모든 경우를 순차적으로 탐색하면, 특정 용량일 때 통학 가능 여부를 판단하는 로직 $\mathcal{O}(N)$을 최대 L번 수행하므로 시간초과를 받습니다.

- ✓ 따라서 킥보드 용량과 충전 횟수 관계가 서로 단조증감함수 성질을 지닌다는 점을 이용해 이분 탐색으로 $\mathcal{O}(N \log L)$ 로 해결해야 합니다.
- ✓ 집과 학교의 위치도 탐색 범위에 포함시켜야된다는 점을 유의합시다.

G. 보물찾기 2

`dijkstra`, `0_1_bfs`

출제진 의도 – **Hard**

- ✓ 출제자: 이동훈aru0504

- 시작점부터 끝점까지의 최단거리를 구하는 문제입니다.
- 가중치가 일정하지 않아서, 단순한 bfs를 적용할 수 없습니다.
- 가중치는 총 0, 1로 두 가지입니다. 이를 이용할 수 없을까요?

- ✓ 풀이 방법 중 하나는 데이크스트라 `dijkstra` 입니다.
- ✓ 음이 아닌 정수를 가중치로 가지므로, 해당 알고리즘을 사용해서 최단거리를 구할 수 있습니다.
- ✓ 이보다 더 간단하고 효율적인 방법이 있을까요?

- ✓ 가중치는 0 또는 1 로 두 가지입니다. 간단한 최단 거리 bfs를 생각해 봅시다.
- ✓ 큐에서 정점을 뽑았을 때, 해당 정점은 큐에 있는 가중치보다 작거나 같습니다.
- ✓ 이 성질을 활용해서, 덱dequeue 을 사용합시다.

- ✓ 시작점부터 탐색합시다. 현재 간선의 가중치가 0 인 경우에는 덱의 앞쪽에, 1 인 경우에는 덱의 뒤쪽에 삽입합니다.
- ✓ 탐색하는 동안 덱 내부 원소들의 가중치의 종류는 최대 두 가지이며, 가중치를 기준으로 정렬돼 있습니다.
- ✓ 우선순위 큐`priority-queue` 의 경우, 원소를 뽑는 데 $\mathcal{O}(\log N)$ 이 들지만, 덱을 사용하면 $\mathcal{O}(1)$ 만에 할 수 있습니다.

- ✓ 이와 같은 풀이 방법을 `0-1 bfs`라고 합니다. 주로 가중치의 종류가 0 또는 1로 두 가지뿐인 경우에 사용합니다.
- ✓ 데이크스트라와 구현이 비슷합니다. 정점이 HW 개, 간선이 $\mathcal{O}(4HW)$ 개이므로 총 시간복잡도는 $\mathcal{O}(HW)$ 입니다.

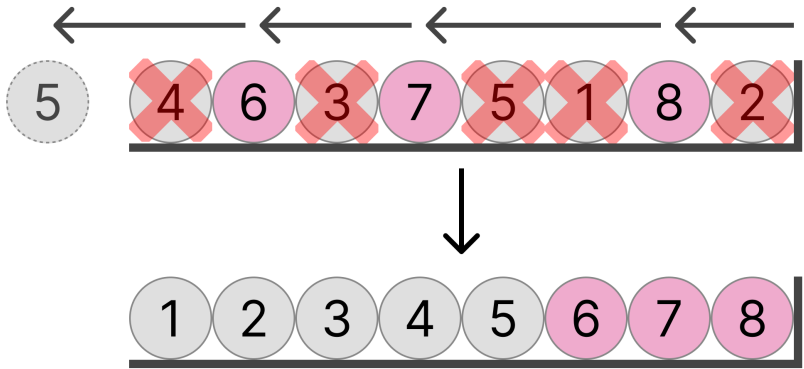
H. 볼링장 아르바이트

`ad_hoc`, `sorting`
출제진 의도 – **Hard**

✓ 출제자: 이승엽 `delena0702`

- ✓ 모든 볼링공을 정렬하기 위해서는 각 볼링공마다 최대 1번만 이동하면 정렬할 수 있습니다.
- ✓ 예시로 가장 무거운 공을 제외한 나머지 볼링공을 무거운 순서대로 이동하면 $N-1$ 번에 정렬할 수 있습니다.
- ✓ 따라서 반드시 이동이 필요한 볼링공들의 개수를 세면 최소 이동 횟수를 구할 수 있습니다.

- ✓ 최소 이동 횟수로 볼링공을 정렬하기 위해서는 가장 무거운 공보다 뒤에 있는 공들은 반드시 이동해야 합니다.
- ✓ 그다음으로, 2번째로 무거운 공과 1번째로 무거운 공 사이의 모든 공이 이동해야 하고, 3번째와 2번째 사이 역시 마찬가지입니다.
- ✓ 이때, 만약 2번째로 무거운 공이 1번째로 무거운 공보다 뒤에 있다면, 이미 배열의 앞으로 이동할 예정이므로 1번째 무거운 공 앞의 모든 공들이 이동해야 하는 공이 됩니다.


5
4
6
3
7
5
1
8
2
1
2
3
4
5
6
7
8

- ✓ 이렇게 모든 공들을 확인할 때까지 공이 무거운 순서대로 확인하게 된다면, 반드시 이동해야 할 공들의 개수를 구할 수 있습니다.
- ✓ 따라서 원래 배열에서 역순으로 탐색하여 가장 무거운 공부터 순서대로 몇 개가 있는지 확인하면, 이동할 필요가 없는 공의 개수를 구할 수 있습니다.
- ✓ 따라서 답은 N – [이동할 필요가 없는 공의 개수] 입니다.
- ✓ 이 풀이의 시간복잡도는 정렬하는데 걸리는 시간인 $\mathcal{O}\left(N \log N\right)$ 입니다.

I. 문자열 게임

`dynamic_programming`
출제진 의도 – **Hard**

- ✓ 출제자: 이승엽 `delena0702`

- ✓ 단어의 마지막이 j 번째 보드에서 끝나게 될 때의 최대 매칭 횟수에 대해 생각해봅시다.
- ✓ 이는 마지막 문자를 제외한 단어를 매칭시켰을 때, $j-1$ 또는 $j+1$ 에서 최대 매칭 횟수를 이용하여 구할 수 있습니다.
- ✓ 이 문제는 부분 문제를 이용하여 전체 문제를 해결할 수 있으므로, 2차원 dp를 이용하여 해결할 수 있습니다.

- ✓ $dp[i][j]$ 를 "단어의 i 번째 문자까지 매칭하고, 마지막 문자가 보드의 j 번째에 매칭됐을 때, 최대 매칭 횟수" 라고 정의하겠습니다.
- ✓ $dp[i][j]$ 즉, 마지막으로 보드의 j 번째에 매칭되기 위해서는 $i-1$ 번째 문자에서 $j-1$ 또는 $j+1$ 에 매칭되어야 합니다.
- ✓ $dp[i][j] = max(dp[i-1][j-1], dp[i-1][j+1]) + \begin{cases} 1, & \text{if } word[i] = board[j] \\ 0, & \text{otherwise} \end{cases}$
- ✓ 보드의 양 끝의 예외처리에 유의해야합니다.

- ✓ dp의 첫 항은 단어의 첫 문자와 보드의 문자가 일치하면 1, 아니면 0으로 초기화 할 수 있습니다.
- ✓ 답은 dp의 마지막 행의 최대값이 정답이 됩니다.
- ✓ 이 풀이의 총 시간복잡도는 $\mathcal{O}(NM)$ 입니다.

J. 압도적 XOR 수

`math`, `bit_mask`

출제진 의도 – **Challenging**

✓ 출제자: 김태현 `kth990303`

- ✓ 먼저, 압도적 xor 수의 정의부터 살펴봅시다.
- ✓ 압도적 xor 수의 정의를 파악하려면 A와 B의 상관관계에 대해 살펴볼 필요가 있습니다.
- ✓ P는 A보다 큰 2의 제곱수 중 가장 작은 값이므로, $P-1$은 A보다 크거나 같은 2^i-1 (i는 자연수)임을 알 수 있습니다.

- ✓ $2^i - 1$을 2진수로 나타내보면 1로만 이루어짐을 알 수 있습니다. 그러므로 xor 성질에 따라 $B = A \oplus (P-1)$에서 B는 A와 서로 완전히 다른 비트를 가지는 수임을 파악할 수 있습니다.
- ✓ 다시 말해 A가 가지는 비트 성질을 완전히 제외(반전)시키므로 $B = P - 1 - A$입니다.
- ✓ 예를 들어 $A = 13(1101_{(2)})$이면 $P - 1 = 15(1111_{(2)})$이고, $B = A \oplus (P-1)$은 $2(0010_{(2)})$입니다.
- ✓ P는 조건을 만족하는 값 중 가장 작은 값이므로 $B = P - 1 - A < A$을 만족합니다.

- ✓ 이제 압도적 xor 수의 정의를 살펴봅시다.
- ✓ $A \geq B \cdot (2^K - 1)$ 일 때 A가 압도적 xor 수라고 합니다.
- ✓ A와 B는 2진수일 때 같은 길이를 가지고 완전히 반전된 비트를 가진다는 점을 이용하여 둘의 상관관계를 다시 한 번 살펴봅시다.
- ✓ A, B를 2진수로 나타낼 때 같은 길이를 가지는 범위 내에서 A와 B의 차이가 가장 클 때는 A는 1로만 구성돼있을 때이고 B는 0으로만 구성돼있을 때입니다.
- ✓ 이 때는 $B = 0$이므로 어떠한 길이이든 상관없이 A는 압도적 xor 수입니다. 다시 말해 $1, 3, 7, 15, \ldots$ 과 같이 $2^i - 1$꼴은 반드시 압도적 xor 수입니다.

- ✓ A, B를 2진수로 나타낼 때 같은 길이를 가지는 범위 내에서 A와 B의 차이가 가장 작을 때는 A는 맨 앞을 제외하고 0으로 구성돼있을 때입니다.
- ✓ $2^i > (2^{i-1} + 2^{i-2} + \ldots + 2^0)$($i$는 자연수)임이 성립한다는 점을 이용해봅시다.
- ✓ $A = 100000_{(2)}, B = 011111_{(2)}$일 때, A는 B의 1배보다 큽니다. 그리고 A, B의 맨 앞에서 2번째 비트가 정반되기 전까지는 $(1 + 2)$배보다 작음이 보장됩니다.

- ✓ $A = 110000_{(2)}, B = 001111_{(2)}$ 일 때, A는 B의 $(1+2)$ 배보다 큽니다.
- ✓ 그리고 A, B의 맨 앞에서 3번째 비트가 정반되기 전까지는 $(1+2+4)$ 배보다는 작음이 보장됩니다.
- ✓ 따라서 $A \geq B \cdot (2^K - 1)$이기 위해서는 A, B를 이진수로 나타냈을 때 맨 앞에서 K 번째 비트까지 A는 1로, B는 0으로 구성돼야 합니다.

- ✓ A, B를 이진수로 나타냈을 때 서로 같은 길이일 경우, A가 특정 값 이상이면 압도적 xor수임이 보장됨을 이용해 수학 또는 이분탐색으로 $\mathcal{O}(\log N)$ 시간복잡도에 답을 구할 수 있습니다.
- ✓ 수학으로 접근한다면 $[2^i, 2^{i+1}-1]$ 범위의 정수(i는 음이 아닌 정수)에서 압도적 xor 수의 개수는 $2^{i+2}-1$개임을 이용하여 등비수열 합의 공식을 이용해 구할 수 있습니다.
- ✓ 이분탐색으로 접근한다면 $[2^i, 2^{i+1}-1]$ 범위의 정수(i는 음이 아닌 정수)에서 압도적 xor 수가 특정 값 이상임이 되는 단조함수 꼴을 이룬다는 점을 이용해 $\mathcal{O}\left(\log 2^i\right)$에 구할 수 있습니다.
- ✓ 이 과정을 $i \leq \log N$이 될 때까지 반복해서 더해주면 됩니다.
- ✓ $[2^i, 2^{i+1}-1]$ 범위 경계값이 2^K-1보다 작을 때에도 반드시 압도적 xor 수가 1개는 있다는 점을 주의합시다.