2022 SCCC 봄 스터디 중급 #1

A. 21967 세워라 반석 위에

수열의 길이를 N, 수열의 최댓값을 X라고 합시다. 최댓값과 최솟값의 차이가 2 이하인 가장 긴 구간의 길이를 구해야 합니다. X가 매우 작다는 것에 주목합시다.

임의의 자연수 x에 대해 $x \le A_i \le x+2$를 만족하는 구간의 최대 길이는 $O(N)$에 구할 수 있습니다. 확인해야 하는 x는 $X-2$가지이므로 $O(NX)$에 문제를 해결할 수 있습니다.

B. 22994 이미지 축소

설명의 편의를 위해 입력으로 주어지는 ni, mj를 N, M이라고 합시다.

원본 이미지의 세로 길이는 N의 약수가 되어야 하고, 가로 길이는 M의 약수가 되어야 합니다. 그러므로 N, M의 약수 n, m에 대해 $A[0:n][0:m]$로 A를 만들 수 있는지 확인하면 됩니다. 어떤 수 X의 약수는 최대 $2\sqrt{X}$개이므로 $O(NM\sqrt{NM})$에 해결할 수 있습니다.

사실 $A[i][*] = A[i-n][*]$를 확인하는 작업과 $A[*][j] = A[*][j-m]$를 확인하는 작업을 따로 수행해도 되고, 이 풀이의 시간 복잡도는 $O(NM(\sqrt{N}+\sqrt{M}))$입니다.

C. 22968 균형

높이가 i인 AVL Tree를 만들기 위한 정점의 최소 개수를 $D[i]$라고 정의하면,
$D[i] = D[i-1] + D[i-2] + 1$을 만족합니다.

$D[i] \le V$를 만족하는 가장 큰 i를 찾으면 되고, D 배열의 값은 지수적으로 증가하기 때문에 naive하게 i를 찾아도 됩니다.

D. 22967 구름다리

간선을 적당히 추가해서 지름을 1로 만드는 것은 Clique를 만들면 됩니다. 정점이 N개인 Clique는 $N(N-1)/2$개의 간선이 존재해야 하므로 $2(N-1) \ge N(N-1)/2$일 때만 지름을 1로 만들 수 있습니다. 부등식을 풀면 $N \le 4$를 얻을 수 있습니다.

그 밖의 경우 항상 지름을 2로 만들 수 있습니다. 1번 정점과 $2, 3, \cdots, N$번 정점을 연결하면 추가한 간선들만 이용해도 지름이 2가 됩니다.

E. 24888 노트조각

모든 최단 경로의 합집합, 즉 Shortest Path DAG에서 적당히 이동해 노트 조각을 모두 방문하고 N번 정점까지 갈 수 있는지 확인하면 되고, Shortest Path DAG는 Dijkstra's Algorithm을 이용해 $O(M \log N)$에 구할 수 있습니다.

DAG에서 특정 정점들을 모두 지나는 경로가 있는지 확인하는 것은 DP를 이용해 확인할 수 있습니다. $D[v]$를 시작점에서 v까지 이동했을 때 방문한 노트 조각의 개수라고 정의하면, $D[N]$이 노트 조각의 개수와 동일한지 확인하면 됩니다.

F. 24970 균형 수

수를 반으로 잘라서 앞부분과 뒷부분의 합이 같도록 만들어야 합니다. 즉, (앞부분 절반의 합을 S로 만드는 경우의 수)와 (뒷부분 절반의 합을 S로 만드는 경우의 수), 그리고 이 조건을 만족하는 수들의 합을 알고 있다면 이들을 활용해 정답을 계산할 수 있습니다.

$C(len, sum, start)$와 $S(len, sum, start)$를 각각 숫자 len개를 사용했을 때 합이 sum이고, 제일 처음에 사용한 숫자가 $start$인 **경우의 수**와 **그러한 수들의 합**으로 정의합시다. 앞부분 절반은 0으로 시작하면 안 되기 때문에 이를 판별하기 위해 $start$가 필요합니다. $a \cdots b$꼴의 수 맨 뒤에 c를 추가해서 $a \cdots bc$를 만드는 방식으로 상태 전이를 하면 C배열과 S배열을 모두 계산할 수 있습니다.

- $C(0,0,0,0) = 1, S(0,0,0,0) = 0$
- $0 \le i < 10$인 i에 대해, $C(1,i,i,i) = 1, S(1,i,i,i) = i$
- $C(len, sum, a) \leftarrow C(len-1, sum-c, a)$
- $S(len, sum, a) \leftarrow S(len-1, sum-c, a) \cdot 10 + C(len-1, sum-c, a) \cdot c$

이제 실제로 정답을 구해봅시다. $f(n)$을 길이가 n인 균형 수들의 합이라고 정의하면, $\sum_{i=1}^{N} f(i)$를 계산하면 됩니다.

n이 짝수라면 앞 $n/2$개의 숫자와 뒤 $n/2$개의 숫자의 합이 동일해야 합니다. 모든 s에 대해 아래 식을 더하면 됩니다.

- $\sum_{i=1}^{9} S(n/2, s, i) \cdot \sum_{i=0}^{9} C(n/2, s, i) \cdot 10^{n/2}$
  - leading zero 없이 합이 s인 수들의 합 / 뒷부분으로 가능한 경우의 수 / 뒤에 숫자 $n/2$개 붙음
- $\sum_{i=0}^{9} S(n/2, s, i) \cdot \sum_{i=1}^{9} C(n/2, s, i)$
  - 합이 s인 수들의 합 / 앞부분으로 가능한 경우의 수

n이 홀수라면 앞 $\lfloor n/2 \rfloor$개의 숫자와 뒤 $\lfloor n/2 \rfloor$개의 숫자의 합이 동일하고, 그 사이의 숫자 하나가 들어가야 합니다. 모든 $0 \le s < 10 \cdot \lfloor n/2 \rfloor$에 대해 아래 식을 더하면 됩니다.

- $\sum_{i=1}^{9} S(n/2, s, i) \cdot \sum_{i=0}^{9} C(n/2, s, i) \cdot 10 \cdot 10^{n/2+1}$
  - leading zero 없이 합이 s인 수들의 합 / 뒷부분으로 가능한 경우의 수 / 가운데 들어갈 수 있는 숫자 개수 / 뒤에 숫자 $\lfloor n/2 \rfloor + 1$개 붙음
- $\sum_{i=1}^{9} C(n/2, s, i) \cdot \sum_{i=0}^{9} C(n/2, s, i) \cdot 45 \cdot 10^{n/2}$
  - 앞부분으로 가능한 경우의 수 / 뒷부분으로 가능한 경우의 수 / 가운데 들어갈 수 있는 수들의 합 / 뒤에 숫자 $\lfloor n/2 \rfloor$개 붙음
- $\sum_{i=0}^{9} S(n/2, s, i) \cdot \sum_{i=1}^{9} C(n/2, s, i) \cdot 10$
  - 합이 s인 수들의 합 / 앞부분으로 가능한 경우의 수 / 가운데 들어갈 수 있는 숫자 개수

$C(len, sum, *)$의 합과 $S(len, sum, *)$의 합도 같이 전처리하면 빠르게 정답을 계산할 수 있습니다.

G. 25012 마법의 다이얼

먼저, M개의 다이얼 중 하나는 돌리지 않더라도 최적해를 찾을 수 있음을 관찰해야 합니다. 즉, 원래 점이 있던 위치 중 하나로 M개의 점을 모으면 됩니다.

$f_i(j)$를 i번째 다이얼의 j번째 줄에 점이 오도록 만드는 최소 비용이라고 정의합시다. $f(j) = \sum f_i(j)$의 최솟값을 구하면 됩니다. 함수의 개형을 살펴봅시다.

i번째 다이얼의 $a_1 < a_2 < \cdots < a_k$번째 줄에 점이 있다고 가정합시다. $a_x \le j \le a_{x+1}$인 j에 대해,
$j \le (a_x + a_{x+1})/2$이면 $f_i(j) = j - a_x$이고, $j \ge (a_x + a_{x+1})/2$이면 $f_i(j) = a_{x+1} - j$입니다.
$j < a_1 \ or \ j > a_k$인 경우도 비슷하게 처리할 수 있습니다.

$f_i(j)$의 개형은 기울기가 1 또는 -1인 `\/\/\/\...` 형태가 되고, 기울기가 바뀌는 지점은 점의 개수에 비례한다는 것을 알 수 있습니다. 그러므로 $f_i(j)$의 기울기가 바뀌는 지점과 기울기의 변화량을 모두 구한 다음, 스위핑을 통해 그 정보들을 합치면 $f(j)$를 얻을 수 있습니다. 기울기가 바뀌는 지점에 대해서만 $f(j)$를 구해도 최솟값을 알 수 있음에 유의하세요.

기울기가 바뀌는 지점이 정수로 떨어지지 않아서 구현이 귀찮을 수 있는데, 항상 반정수 형태이므로 2를 곱하면 쉽게 처리할 수 있습니다. `std::map`이나 `std::priority_queue` 등을 이용해 $O(N \log N)$에 해결할 수 있습니다.

H. 25011 칠하기

가로 방향으로 연속한 칸 그룹과 세로 방향으로 연속한 칸 그룹을 정점으로 생각하면 방향 그래프를 만들 수 있습니다. 만들어진 모든 정점을 지나는 보행(walk)이 존재하는지 판별하는 문제로 생각할 수 있습니다.

어떤 정점 v를 방문할 수 있다면 v와 같은 SCC에 속한 정점을 모두 방문할 수 있기 때문에 SCC를 압축한 DAG에서 문제를 해결해도 됩니다. DAG의 한 정점에서 출발해 모든 정점을 방문하는 walk가 있는지 판별하면 되고, 압축한 그래프가 선형인지 확인하면 됩니다. Kosaraju’s Algorithm는 위상정렬 순서대로 SCC를 찾기 때문에 i번째 SCC에서 $i+1$번째 SCC로 가는 간선이 있는지 확인하면 됩니다.