2022 SCCC 봄 스터디 중급 #3

A. 25045 비즈마켓

A는 오름차순, B는 내림차순으로 정렬한 뒤, $\max(0, A_i - B_i)$의 합을 구하면 됩니다.

B. 24023 아기 홍윤

or 연산을 하면 값은 항상 단조 증가하므로 투포인터를 사용할 수 있습니다. 각 비트를 몇 번 사용했는지 기록하면 $O(30N)$ 정도에 문제를 해결할 수 있습니다.

C. 3673 나눌 수 있는 부분 수열

$S_i = \sum_{k=1}^{i} A_k$라고 정의하면, 수열의 구간 합 $A_s + A_{s+1} + \cdots + A_e$는 $S_e - S_{s-1}$로 나타낼 수 있습니다. 즉, $S_{s-1} \equiv S_e \pmod{d}$인 순서쌍 (s, e)의 개수를 구하면 됩니다.

D. 23258 밤편지

2^C 이상 먹을 수 없다는 것은 $1 \cdots C-1$번 정점만 경유할 수 있다는 것을 의미합니다. 그러므로 플로이드 와샬 알고리즘의 중간 과정을 모두 저장하면 전처리를 $O(N^3)$ 만큼의 메모리를 사용해 $O(N^3)$ 시간에 할 수 있고, 이후 쿼리를 상수 시간에 처리할 수 있습니다.

E. 8916 이진 검색 트리

이진 트리를 위상 정렬하는 경우의 수를 구하는 문제입니다.

현재 정점 v의 자식 정점 c_1, c_2를 루트를 하는 서브트리를 위상 정렬하는 경우의 수를 각각 f_1, f_2, 서브트리의 크기를 각각 s_1, s_2라고 합시다. 첫번째 서브트리의 정점에 모두 1이라는 라벨을 붙이고, 두번째 서브트리의 정점에 모두 2라는 라벨을 붙이면, (중복이 있는 순열의 개수) * f_1 * f_2를 구하면 됩니다.

그러므로 v를 루트로 하는 서브트리를 위상 정렬하는 경우의 수는 $\frac{(s_1+s_2)!}{s_1!s_2!} f_1 f_2$이고, 트리 DP를 이용해 문제를 해결할 수 있습니다.

F. 21725 더치페이

각 사람이 지출한 금액과 실제로 지출해야 하는 금액을 어떻게 잘 계산했다고 가정하고, 송금 과정을 구하는 방법부터 알아봅시다. 한 명이 **"은행"** 역할을 해서 돈을 적게 낸 사람은 은행에게 송금하고, 돈을 많이 낸 사람은 은행에게 돈을 받으면 $n-1$번의 송금만 필요합니다.

Union-Find를 이용해 그룹이 합칠 때마다 새로운 정점을 추가하면 그룹이 합쳐지는 과정을 트리로 나타낼 수 있습니다. 그룹이 돈을 지불할 때마다 그룹을 나타내는 정점에 지불한 돈을 더하면, 각 사람이 지불해야 하는 금액은 트리에서 자신의 조상 정점의 가중치를 모두 더한 것이 됩니다. 이는 DFS를 이용해 계산할 수 있습니다.

Union-Find 과정에서 $O(N \log N)$, DFS를 이용해 지불해야 하는 금액을 계산하는데 $O(N)$, 송금 과정을 복원하는데 $O(N)$이 걸리므로 전체 시간 복잡도는 $O(N \log N)$입니다.

G. 22306 트리의 색깔과 쿼리 2

컴포넌트에 **아이디**라는 것을 붙여서, 각 정점이 어떤 컴포넌트에 속한 상태인지 관리합시다. 각 정점이 몇 번 컴포넌트에 속했는지 알고 있다면, std::map 등을 이용해 컴포넌트의 색깔을 관리할 수 있습니다.

간선이 끊어질 때마다 컴포넌트의 아이디를 갱신해야 하는데, 분할되는 두 컴포넌트 중 작은 컴포넌트에 속한 정점들의 아이디만 바꾸면 small to large의 원리로 정점의 아이디를 최대 $O(N \log N)$번만 바꿀 수 있습니다. 실제로 각 컴포넌트의 크기를 직접 구하는 것은 어렵고, 두 컴포넌트에서 동시에 DFS를 돌리면서 한쪽이 끝날 때까지 탐색을 구행하면 됩니다.

DFS를 하고 std::map을 이용해 컴포넌트의 색깔을 관리하는데 $O(N \log^2 N)$이 걸리므로 문제를 해결할 수 있습니다.

Dynamic Tree(ex. Euler Tour Tree)를 이용해 컴포넌트의 크기를 amortized $O(\log N)$ 시간에 구하고, std::map 대신 hash table을 사용하면 $O(N \log N)$에 해결할 수 있습니다.

H. 19855 3분 그래프 리턴즈

한 지점에 3개의 구간이 있다면 사이클이 만들어집니다. 그러므로 각 지점을 2개 이하의 구간만 지나도록 구간을 적절히 선택하는 문제라고 생각할 수 있습니다. 2개 이하 조건은 어려워보이니 1개 이하 조건 문제를 먼저 풀어봅시다.

각 지점에 최대 1개의 구간만 지나도록 구간을 선택하는 것은, $0, 1, 2, \cdots, 500\,000$까지의 지점을 만들어서 한 칸씩 뒤로 이동하다가, 구간 $[s, e]$를 선택하면 $s-1$에서 e로 점프하는 것이라고 생각할 수 있습니다. 이때 점프하는 구간의 가중치의 합을 최대화하면 됩니다. 이는 $i-1$에서 i로 가는 가중치가 0인 간선과, $s-1$에서 e로 가는 가중치가 t_i인 간선을 만든 다음 0에서 50만으로 가는 **최장 경로**를 구하는 것과 동일합니다. DAG이므로 DP를 이용해 $O(N)$에 구할 수 있습니다.

2개 이하 조건은 MCMF로 해결할 수 있습니다. 0에서 50만까지 유량을 2만큼 흘릴 때의 최대 가중치를 구하면 됩니다. 이때 $i-1$에서 i로 가는 간선의 용량은 ∞, $s-1$에서 e로 가는 간선의 용량은 1입니다. MCMF를 그대로 구현하면 이므로 시간 초과를 받게 되므로 그래프의 형태를 이용해 최적화해야 합니다.

DAG이므로 첫 유량은 DP를 이용해 $O(N)$에 구할 수 있습니다. 첫 유량에서 사용한 간선을 뒤집어야 하는데, 간선을 뒤집으면 DAG 성질이 없어지기 때문에 DP를 사용하지 못하고, 음수 가중치가 있기 때문에 최단 경로를 빠르게 찾지도 못합니다.

MCMF에서 Johnson's Algorithm을 이용해 음수 가중치를 없애는 방법은 방법은 잘 알려져 있습니다. 마침 첫 번째 유량을 구하면서 각 정점까지의 최단 거리를 구했기 때문에 추가적인 과정 없이 음수 가중치를 없앨 수 있습니다. 음수 가중치를 없앤 다음, Dijkstra's Algorithm을 이용하면 두 번째 유량도 빠르게 찾을 수 있습니다.