ICPC Sinchon #11

문제 목록

- **BOJ 3178 코코스** (Croatian OI 2006)
- **BOJ 5875 오타** (USACO 2012 November Bronze)
- **BOJ 6051 시간 여행** (USACO 2010 US Open Silver)
- **BOJ 16758 NLO** (COCI 2018/2019 #3)
- **BOJ 25337 Merge the Tree and Sequence** (제5회 천하제일 코딩대회)
- **BOJ 1979 극적인 곱셈** (ICPC Tehran Site 2005)
- **BOJ 25207 바벨탑의 저주** (2022 인하대학교 프로그래밍 경진대회)
- **BOJ 25008 문자열 찾기** (2020 1차 국가대표 선발고사)
- **BOJ 25011 칠하기** (2020 1차 국가대표 선발고사)
- **BOJ 17675 램프** (2018/2019 JOI Spring Camp Day 3)
- **BOJ 24942 Jail** (2021/2022 JOI Spring Camp Day 1)
- **BOJ 8987 수족관 3** (2013 한국 정보 올림피아드 고등부)

BOJ 3178 코코스

문제 요약

- 길이가 $2K$인 문자열 N개 주어짐
- T_1을 앞에 있는 K개의 문자로 구성된 문자열들로 만든 트라이
- T_2를 뒤에 있는 K개의 문자로 구성된 문자열을 뒤집은 문자열로 만든 트라이라고 할 때
- T_1과 T_2의 정점 개수의 합을 구하는 문제

풀이

- 트라이를 직접 만들면 메모리 초과를 받게 됨
- 트라이의 각 정점은 트라이를 구성하는 문자열들의 접두사에 대응됨
- 따라서 서로 다른 접두사의 개수를 구하는 문제라고 생각할 수 있음
  - NK에서 중복으로 등장하는 접두사의 개수를 빼는 방식으로 문제를 풀어보자.
- 임의의 문자열 s를 접두사로 갖는 문자열들은 정렬했을 때 인접한 위치에 있음
  - 트라이를 구성하는 문자열을 정렬하고 인접한 문자열만 봐도 중복 접두사를 모두 구할 수 있음
  - $S[i-1]$과 $S[i]$의 최장 공통 접두사의 길이가 x이면 중복 접두사가 x개이므로 정답에서 x를 빼면 됨

BOJ 5875 오타

문제 요약

- 괄호 문자열이 주어짐
- 문자 하나만 고쳐서 올바른 괄호쌍으로 만드는 경우의 수

풀이

- 올바른 괄호 문자열인지 판별하는 방법
  - 여는 괄호를 +1, 닫는 괄호를 -1로 생각했을 때
  - 합이 0이고 prefix sum이 모두 0 이상이면 됨
- 수식으로 표현

    - i번째 문자가 여는 괄호면 $A[i] = +1$, 닫는 괄호면 $A[i] = -1$로 정의하자.
    - $S[i] = A[1] + A[2] + \cdots + A[i]$라고 정의하자.
    - 어떤 문자열이 올바른 괄호 문자열이라는 것은 $S[N] = 0 \wedge \min S = 0$이라는 것과 동치이다.
- 여는 괄호를 닫는 괄호로 바꾸는 경우
    - $A[i]$가 2 만큼 감소하므로 $S[i]$부터 $S[N]$까지 모두 2씩 감소
    - $S[N] - 2 = 0$이고 $\min S[1 \cdots i-1] \geq 0, \min S[i \cdots N] - 2 \geq 0$인지 확인하면 됨
- 닫는 괄호를 여는 괄호로 바꾸는 경우
    - $A[i]$가 2 만큼 증가하므로 $S[i]$부터 $S[N]$까지 모두 2씩 증가
    - $S[N] + 2 = 0$이고 $\min S[1 \cdots i-1] \geq 0, \min S[i \cdots N] + 2 \geq 0$인지 확인하면 됨
- S의 prefix min과 suffix min을 관리하면 된다.

코멘트 - 추천 문제

- BOJ 14476 최대공약수 하나 빼기

BOJ 6051 시간 여행

문제 요약

- 3가지 쿼리를 처리하는 문제
    - 스택의 맨 뒤에 x 추가
    - 스택의 맨 뒤에 있는 원소 제거
    - K번째 쿼리를 수행하기 직전으로 돌아감

풀이

- persistent stack을 구현하는 문제
    - persistent data structure: 과거의 상태를 모두 보존하고 있는 자료구조
    - persistent stack: 과거의 상태를 모두 보존하고 있는 스택
    - persistent segment tree: 과거의 상태를 모두 보존하고 있는 세그먼트 트리
- 스택을 연결 리스트로 직접 구현
- 연결리스트의 노드에서는 원소의 값과 바로 밑에 있는 원소의 포인터를 저장
- Top[i] = i번째 쿼리를 처리한 뒤 스택의 맨 뒤에 있는 원소의 포인터를 관리하면 됨

BOJ 16758 NLO

문제 요약

- $N \times M$ 격자에서 쿼리를 Q번 처리해야 함
    - (X_i, Y_i, R_i): $(x - X_i)^2 + (y - Y_i)^2 \leq R_i$을 만족하는 $A[x][y]$을 0으로 만들고, 다른 모든 칸은 1 증가시킴
- Q개의 쿼리를 처리한 뒤 격자에 적힌 수들의 합을 구하는 문제

풀이

- 각 칸마다 마지막으로 0이 되는 쿼리 q를 찾으면 그 칸의 최종 값은 $Q - q$라는 것을 알 수 있음
- 만약 2차원이 아니라 1차원이라면?
    - $X_i - R_i \leq x \leq X_i + R_i$인 x에 대해, $A[x] \leftarrow \max(A[x], i)$를 수행하면 됨
    - 세그먼트 트리를 사용해도 되고 스위핑을 해도 됨
    - x를 1부터 N까지 차례대로 보면서
    - $x = X_i - R_i$를 만나면 집합 S에 i를 넣고, $x = X_i + R_i$를 만나면 S에서 i를 제거
    - $A[x] = \max S$
    - $O(N + Q \log Q)$에 할 수 있음

- 쿼리 개수가 100개밖에 없다는 점을 이용하면 2차원에서도 동일하게 해결할 수 있음
- 각 행마다 쿼리에 해당되는 열들의 구간을 구한 다음, 1차원 풀이를 그대로 적용하면 됨

BOJ 25337 Merge the Tree and Sequence

문제 요약

- N개의 수로 구성된 수열 B와 N개의 정점으로 구성된 트리가 주어짐
- 트리의 정점마다 가중치 A_i가 있고, 간선은 각각 한 가지 색깔로 칠해져 있음
- 같은 정점을 공유하면서 색이 같은 간선들을 한 그룹으로 묶자.
- B의 원소를 트리의 정점에 하나씩 배정했을 때
  - (그룹에 속한 간선들의 끝점의 가중치 A_i의 합) × (그룹에 속한 간선들의 끝점에 배정된 B_i의 합)
  - 으로 가능한 최솟값과 최댓값을 구하는 문제

풀이

- 그룹의 가중치는 $(A_1 + A_2 + \cdots + A_k) \times (B_1 + B_2 + \cdots + B_k)$ 꼴로 표현할 수 있다.
  - 트리의 정점에 B를 배정하는 문제이므로 B를 떼어내는 게 좋음
  - 모든 그룹에 대해 $B_1 \sum A_i + B_2 \sum A_i + \cdots$의 합
  - (i번 정점을 포함하는 그룹의 $\sum A_i$의 합) × B_i의 합으로 가능한 최솟값과 최댓값을 구하는 문제
- 간선을 묶는 작업은 인접 리스트를 색깔 기준으로 정렬한 다음 Union Find로 묶을 수 있음
- Union Find를 이용해 그룹에 속한 정점들의 가중치 합과 각 정점이 속한 그룹의 가중치 합을 구할 수 있음
- $A'_1, A'_2, \cdots, A'_N$과 $B_1, B_2, \cdots, B_N$을 적당히 매칭해서 만들 수 있는 최솟값과 최댓값을 구해야 함
  - 둘 다 오름차순으로 정렬해서 매칭하면 최댓값
  - 하나 내림차순으로 바꾸고 매칭하면 최솟값

코멘트 - 추천 문제

- BOJ 25341 인공 신경망

BOJ 1979 극적인 곱셈

문제 요약

- 어떤 양의 정수 x에 n을 곱한 결과가 x의 숫자들을 오른쪽으로 한 칸씩 옮긴 수라면 x를 n-극적인 수라고 하자.
  - ex. 102564 * 4 = 410256 이므로 102564는 4-극적인 수
- n, k가 주어지면 n-극적인 수 중 마지막 자릿수가 k인 가장 작은 수를 구하는 문제

풀이

- 일단 수의 길이 L을 고정하자.
- 길이가 $L-1$인 어떤 양의 정수 t에 대해, 우리가 구해야 하는 x는 $x = 10t + k$꼴
- $xn = (10t + k)n = k \times 10^{L-1} + t$
- $(10n - 1)t = 10^{L-1}k - nk$
- $t = (10^{L-1} - n)k / (10n - 1)$
- 지금까지 나온 문자들을 정리해 보면
  - n, k는 문제에서 주어진 상수

    - t, L은 양의 정수인 변수, 가장 작은 (L, t) pair를 찾아야 함
        - 위 식에 의해 t가 정수가 되는 가장 작은 L만 구하면 됨
- t가 정수가 되기 위해서는 $(10^{L-1} - n)k \equiv 0 \pmod{10n-1}$
- $10^{L-1}k \equiv nk \pmod{10n-1}$
- $n \neq 1$이면 n, k는 서로소이므로 양변에서 k를 없앨 수 있음
    - $n = 1$이면 정답은 k임
- $10^{L-1} \equiv n \pmod{10n-1}$
- $10^{L-1} \pmod{10n-1}$의 주기는 최대 $10n-1$이므로 $L \in [1, 10n-1]$만 보면 됨

코멘트 - 주의 사항

- 수의 길이를 고정 → 수식 전개 → 정수론적 지식
- 상수, 변수, 최적화할 변수, 변수의 조건, k를 소거할 수 있는 조건을 놓치지 않고 잘 따라가야 함

BOJ 25207 바벨탑의 저주

문제 요약

- 정점마다 정수 A_i가 적혀 있음
- 각 정점을 루트로 하는 서브 트리마다, 서브 트리에 속한 모든 정점의 자식 순서를 뒤집었을 때
- 정점에 적힌 A_i가 뒤집기 전과 동일한지 판별하는 문제

풀이

- 서브 트리가 팰린드롬인지 판별하는 문제
- Euler Tour Technique를 사용하면 in(v) ~ out(v)가 팰린드롬인지 판별하는 문제
    - 팰린드롬 판별은 Manacher algorithm 또는 해싱을 이용하면 됨
    - A_i만 저장하면 틀리고 깊이까지 함께 저장해야 함

BOJ 25008 문자열 찾기

문제 요약

- 두 문자열 A, B가 아래 세 조건을 모두 만족할 때, 두 문자열을 **사실상 같다**고 한다.
    - $|A| = |B|$
    - $A[i] = A[j]$이면 $B[i] = B[j]$
    - $A[i] \neq A[j]$이면 $B[i] \neq A[j]$
- 문자열 T, P가 주어지면 P와 **사실상 같은** T의 부분 문자열의 개수를 구하는 문제

풀이

- KMP로 해결할 수 있다.
- KMP에서 문자열을 매칭하는 과정과 실패 함수를 구할 때 수행하는 작업은 한 가지밖에 없음
    - 문자열 A와 B가 **동치**일 때 $A + a$와 $B + b$가 **동치**인지 확인
    - 일반적인 문자열에서는 $a = b$인지 확인하면 됐지만 이 문제는 조금 다름
- A, B가 동치일 때 $A + a, B + b$가 동치인지 확인하는 방법
    - A에 이미 a가 등장했다면 a는 매칭되어야 하는 짝이 정해져 있음
    - B에 이미 b가 등장했다면 b는 매칭되어야 하는 짝이 정해져 있음
    - T, P에서 각 문자 c마다, c보다 앞에 있으면서 가장 최근에 c가 등장한 위치를 전처리하면 쉽게 해결할 수 있음
- 더 자세한 설명은 https://justicehui.github.io/review/2021/08/08/scpc-qual/ 의 2차 예선 4번 패턴 매칭 문제 풀이 참고

코멘트 - 추천 문제

- BOJ 3308 Matching, BOJ 20298 파인애플 피자, BOJ 23576 Stock Price Prediction

BOJ 25011 칠하기

문제 요약

- $N \times M$ 격자의 한 지점에서 출발
- 상하좌우 4방향 중 한 방향을 골라서 벽을 만날 때까지 계속 이동하는 것을 반복함
- 가로 방향으로 이동하면서 지나는 칸은 파란색, 세로 방향으로 이동하면서 지나는 칸은 노란색으로 칠함
- 모든 칸에 파란색과 노란색을 모두 칠하는 방법이 있는지 판별하는 문제

풀이

- 벽과 인접한 칸에서만 진행 방향을 바꿀 수 있음
- 벽을 기준으로 가로 방향 그룹과 세로 방향 그룹을 묶으면 방향 그래프를 만들 수 있음
- 모든 정점을 방문하는 walk가 존재하는지 판별하는 문제
- SCC를 묶은 DAG가 선형인지 판별하는 문제
- Kosaraju's algorithm은 위상 정렬 순서대로 SCC를 찾기 때문에, 단순히 i번째 SCC에서 $i+1$번째 SCC로 가는 간선이 있는지 확인하면 됨

코멘트 - 추천 문제

- BOJ 14737 Dev, Please Add This!

BOJ 17675 램프

문제 요약

- 0과 1로만 구성된 길이 N짜리 수열 A, B가 주어짐
- 아래 3가지 연산을 이용해서 A를 B로 만들 때 필요한 연산의 최소 횟수
  - 구간 $[l, r]$을 0으로 변경
  - 구간 $[l, r]$을 1로 변경
  - 구간 $[l, r]$의 상태를 반전

풀이

- 기본적인 관찰
  - 한 지점에 1, 2번 연산을 모두 사용할 필요 없음
  - 한 지점에 3번 연산을 여러 번 사용할 필요 없음
  - 3번 연산은 1, 2번 연산을 모두 끝낸 다음에 해도 됨
  - 1, 2번 연산을 적용하는 구간은 서로 겹치지 않음
  - 3번 연산을 적용하는 구간은 서로 겹치지 않음
- 중요한 관찰
  - 1번 연산을 사용한 구간과 2번 연산을 사용한 구간이 인접하지 않는 최적해가 존재
  - 만약 인접하는 최적해가 존재한다면, 두 구간의 합집합에 1번 연산을 적용한 뒤, 2번 연산을 사용할 구간에 3번 연산을 사용하면 됨
- 전략
  - 수열에 0, 1, x를 적당히 깔아놓자. (x는 그대로)
  - xx00xx00xx11xx00xx11 같은 형태
  - 이 상황에서 3번 연산을 적당히 적용하면 됨
- 점화식

- $D(i, j) := A[1 \cdots i]$만 신경썼을 때, i번째 값에 j를 깔아놓고 $A[1 \cdots i]$를 $B[1 \cdots i]$와 동일하게 만들 때 필요한 연산의 최소 횟수
- $C(i, j) := i$번째 값에 j를 깔았을 때 $B[i]$와 동일하면 0, 다르면 1 (3번 연산이 필요한지 확인)
- 상태 $3N$개, 각 상태의 답을 $O(1)$에 구할 수 있으므로 $O(N)$에 해결할 수 있음

코멘트 - 추천 문제

- 1, 2번 연산은 그 전까지의 모든 작업을 덮어버리는 연산이라는 점을 생각하면 풀이를 생각하기 쉬움
- BOJ 11934 Fortune Telling 2

BOJ 24942 Jail

문제 요약

- N개의 정점으로 구성된 트리에 M명의 사람이 있음
- i번째 사람은 S_i에서 T_i로 최단 경로를 통해 이동해야 함
- 각 사람은 간선을 타고 한 정점에서 다른 정점으로 이동할 수 있음
- 두 사람이 한 정점에서 만나지 않도록 사람들의 이동을 스케줄링할 수 있는지 판별하는 문제

풀이

- 선형인 경우를 먼저 생각해 보자.
  - 역전이 있으면 불가능, 없으면 가능
  - 역전이 없는 경우에는 각 사람을 중간에 멈추지 않고 한 번에 옮길 수 있음
  - 잘 생각해 보면 이건 트리에서도 성립한다는 것을 알 수 있음
- 두 경로 (S_1, T_1)와 (S_2, T_2)를 생각해 보자.
  - 만약 경로에 겹치는 부분이 없으면 두 경로의 이동 순서는 상관 없음
  - 경로의 중간 지점만 겹치는 상황에서도 두 경로의 순서는 상관 없음
  - S 또는 T가 다른 경로 위에 놓인 경우만 신경써도 됨
  - $S_1 - S_2 - T_1$ 경로가 존재하면 (S_2, T_2)가 먼저 이동해야 함
  - $S_1 - T_2 - T_1$ 경로가 존재하면 (S_1, T_1)이 먼저 이동해야 함
  - 위상 정렬
- Sparse Table을 이용해 위상 정렬 그래프 만들면 100점 가능
  - https://justicehui.github.io/tutorial/2020/09/05/graph-with-segment-tree/ 참고
  - 세그먼트 트리는 HLD가 필요해서 log가 하나 더 붙음

코멘트 - 서브태스크 관련

- subtask 1. 선형 - 역전 유무 판별
- subtask 3. $M \leq 6$ - 트리에서도 한 번에 옮길 수 있다고 추측한 다음 $M!$가지 시도해서 서브태스크로 검증

BOJ 8987 수족관 3

문제 요약

- 카르테시안 트리과 비슷한 방식으로 수족관을 이진 트리로 만들자.
- 정점에 가중치가 있는 트리에서 정점을 최대 K개 골라서
- 선택한 정점들의 조상들의 가중치 합을 최대화하는 문제

풀이

- 리프 정점만 선택해도 됨
  - 만약 리프 정점이 아닌 정점 v를 선택하는 것이 최적이라고 가정하면
  - v의 자식 c를 선택해도 v의 조상이 모두 포함되고, c를 추가로 가져갈 수 있으므로 더 좋은 해를 만들 수 있음
- $D(v, k)$: v를 루트로 하는 서브 트리에서 리프를 k개 선택했을 때의 최댓값
- $D(u, *)$와 $D(v, *)$를 효율적으로 합칠 수 있으면 문제를 해결할 수 있음
- $D(v, *)$는 볼록함
  - MCMF로 모델링된다는 것을 이용해 증명할 수 있음
- $D(r, k_1 + k_2) \leftarrow D(u, k_1) + D(v, k_2)$를 해야 함
- $D(v, *)$는 볼록하니까 기울기가 큰 것부터 하나씩 끼워넣으면 우선순위 큐를 이용해 $O(S \log S)$에 합칠 수 있음.
  - $S = \min\{|D(u)|, |D(v)|\}$
  - DP값 대신 기울기를 저장하는 것이 좋음. $D(v, 0) = 0$이므로 기울기만 알아도 정답을 계산할 수 있음
- small to large하면 $O(N \log^2 N)$에 해결할 수 있음

코멘트 - 사전지식 유무에 따른 풀이까지의 도달 과정

- MCMF를 알고 있는지 여부에 따라 풀이를 찾는 과정이 다름
  - MCMF를 모르는 경우: 리프 정점만 선택해도 된다 → Tree DP 식을 세운다 → 두 정점의 DP값을 어떻게 합치지? → 볼록하니까 잘 합칠 수 있음
  - MCMF를 아는 경우: MCMF로 모델링되네 → 플로우를 하기에는 제한이 너무 큼 → 볼록하니까 두 서브 트리를 빠르게 합칠 수 있음 → Tree DP