# 2020 SciOI #1 후기

SciCom

# 문제별 출제자

| 문제 | 제목 | 출제자 | 검수자 |
|---|---|---|---|
| A | OR과 쿼리 | blackking26 | gs18115, imeimi, cs71107 |
| B | 이상한 편집기 | blackking26 | 79brue, gs18115, imeimi |
| C | 도로 공사 | blackking26 | 79brue, gs18115, imeimi, cs71107 |
| D | 댐 | blackking26 | blackking26, imeimi |
| E | Bridge Construction | 79brue | 79brue, gs18115, imeimi |
| F | 방역 | blackking26 | blackking26, gs18115, imeimi |
| G | 대안적 사실 | blackking26 | blackking26, imeimi |
| H | Tree and Easy Queries | 79brue | 79brue, gs18115, dennisstar, imeimi, cs71107, maruii |
| I | 와일드 카드 | blackking26 | blackking26, gs18115, imeimi |

# A. OR과 쿼리

- 만점자: 21명
- 가장 첫 만점: cheetose (21분)
- 출제자: blackking26

- 특별상: ainta

# A. OR과 쿼리

- OR 쿼리는 단조 증가 연산이기 때문에, 수가 감소하는 일은 일어나지 않습니다.
- 문제의 제한 덕분에, 각 수가 바뀌는 횟수는 logX회 이하입니다. (X=maxAi)
- 따라서 수가 바뀌는 총 횟수는 NlogX회 이하이므로, 어떤 수가 바뀌는지만 빠르게 알 수 있다면, 수를 일일이 바꿔줄 수 있습니다.

# A. OR과 쿼리 : $O(N(logN + logX))$

- 1번 쿼리가 주어질 때마다 쿼리에 의해서 바뀌는 수를 빠르게 찾을 수 있어야 합니다.
- 1번 쿼리에서 주어진 수 x가 어떤 수들을 모두 bitwise AND한 값의 subset 이라면, 그 수들은 모두 쿼리에 영향을 받지 않는 수들입니다.
- bitwise AND 연산으로 segment tree를 만들어서 x가 구간의 bitwise AND 값의 subset이라면 바로 종료합시다.

# A. OR과 쿼리 : $O(N(logN + logX))$

- 세그먼트 트리의 각 노드에 있는 값 역시 최대 logX번만 바뀔 수 있습니다.
- 이로 인해 1번 쿼리에 걸리는 총 시간은 NlogN+NlogX를 넘지 않게 됩니다.
- 이제 어떤 수가 바뀌는지를 빠르게 알 수 있게 되었으므로 정확히 K와 동일한 수의 개수를 다른 세그먼트 트리 등을 이용해 관리하면 문제를 풀 수 있습니다.

# A. OR과 쿼리 : $O(NlogNlogX)$

- 시간복잡도를 조금 포기한다면 더 간단하게도 풀 수 있습니다.
- 30개의 비트에 대해 각각 i번째 비트가 꺼져있는 수의 목록을 std::set 등으로 관리하면 1번 쿼리가 들어올 때마다 변하는 비트를 모두 찾아낼 수 있습니다.
- 시간 제한은 1.5초로 이 풀이는 일반적으로 통과되지 않아야 하지만, std::set이나 세그먼트 트리 대신 펜윅 트리 등을 사용하여 잘 커팅을 하면 시간 안에 들어옵니다.

# A. OR과 쿼리 : O(???)

- 같은 풀이를 유니온-파인드 자료구조 혹은 monotone queue를 이용해서 구현할 수도 있습니다.
- 당연히도 이 경우에는 시간제한 내에 잘 들어옵니다.

# B. 이상한 편집기

- 만점자: 10명
- 가장 첫 만점: kdh9949 (16분)
- 출제자: blackking26

- 특별상: jhwest2

# B. 이상한 편집기 : $O(N^4)$

- $DP[i][j]$: j번째 문자까지 출력하되, 마지막 출력에서는 i~j번 문자를 출력하기 위한 연산의 최소 횟수로 정의합시다.
- $DP[i][j]$를 naive하게 계산하면, $O(N^4)$의 시간 복잡도에 계산할 수 있습니다.

# B. 이상한 편집기 : $O(N^3)$

- $DP[i][j]$는 아래와 같은 식으로 나타낼 수 있습니다.

$$DP[i][j] = \min_{k<i}\left(DP[k][i-1] + (\mathrm{j} - \mathrm{k} + 1) - 2 \cdot \min(LCP[k][i], i-k, j-i+1)\right) + 1$$

- LCP를 모든 i, j 쌍에 대해 전처리해 준다면, $O(N^3)$의 시간 복잡도에 DP 배열을 채워 줄 수 있습니다. LCP 배열은 $O(Nlog^2N)$에도 전처리할 수 있지만 이 문제에서는 $O(N^2)$이면 충분하겠군요.

# B. 이상한 편집기 : $O(N^2 logN)$

- 점화식을 다시 봅시다

$$DP[i][j] = \min_{k<i}\left(DP[k][i-1] + (\mathrm{j} - \mathrm{k} + 1) - 2 \cdot \min(LCP[k][i], i-k, j-i+1)\right) + 1$$

- k의 결정에 j가 영향을 미치는 부분은 $\min(LCP[k][i], i-k, j-i+1)$ 뿐입니다.
- i가 고정되어 있고 j가 커질 때 $LCP[k][i]$는 단조증가 해야 합니다.
- 이를 이용해 DnC optimization을 적용하면 $O(N^2 logN)$에 문제를 풀 수 있고 시간 안에 잘 작동합니다.

# B. 이상한 편집기 : $O(N^2)$

- $O(N^2)$으로도 여러가지 방법이 존재합니다.
- 하지만 귀찮으므로 여러분을 위한 challenge로 남기겠습니다.

# C. 도로 공사

- 만점자: 3명
- 가장 첫 만점: cki86201 (294분)
- 출제자: blackking26

- 특별상: sebinkim

# D. 댐

- 만점자: 1명
- 가장 첫 만점: ainta (1419분)
- 출제자: blackking26

- 특별상: ainta

# E. Bridge Construction

- 만점자: 5명
- 가장 첫 만점: cki86201 (159분)
- 출제자: 79brue

- 특별상: ainta

# E. Bridge Construction : $O(N^2)$

- 문제를 간단히 요약하면, 정점이 N개이고 각 정점의 차수가 $4$ 이하인 트리 구조의 경우의 수를 세는 문제입니다.
- 모든 트리에는 하나 또는 두 개의 센트로이드가 있기 때문에, 센트로이드를 루트로 잡으면 편리해집니다.

# E. Bridge Construction : $O(N^2)$

- $DP[n]$: 정점이 n개이고, 각 정점의 차수가 4 이하이며, 루트 노드의 차수는 3 이하인 rooted-tree의 개수로 정의합시다. (단, $DP[0] = 1$입니다.) 루트 노드의 차수를 3 이하로 하는 이유는, 루트 노드가 나중에 부모 노드와 연결되는 경우를 고려하기 위함입니다.
- $DP2[k][n]$: 위에서 설명한 형태의 rooted-tree를 $k$개 선택해, 그 정점의 수의 합이 $n$이 되는 경우의 수로 정의합시다. 단, 선택하는 순서만 다른 것은 한 가지로 셉니다.

# E. Bridge Construction : $O(N^2)$

- $DP$ 배열과 $DP2$ 배열을 동시에 관리해 봅시다.
- $DP[i] = DP2[3][i-1]$입니다.
- $DP2$ 배열은 $DP[i]$를 구한 직후에 Knapsack-DP와 비슷한 방식으로 업데이트해줄 수 있습니다. 다만 중복 처리를 하기 위해 중복조합을 사용해야 합니다. 자세한 설명은 코드로 대체합니다.

# E. Bridge Construction : $O(N^2)$

```
for(int i=1; i<=(n-1)/2; i++){
    DP[i] = DP2[3][i-1];
    vector<ll> v = {DP[i], DP[i]*(DP[i]+1)%MOD*inv2%MOD,
                    DP[i]*(DP[i]+1)%MOD*(DP[i]+2)%MOD*inv3%MOD,
                    DP[i]*(DP[i]+1)%MOD*(DP[i]+2)%MOD*(DP[i]+3)%MOD*inv4%MOD};
    for(int k=4; k>=1; k--){
        for(int l=k-1; l>=0; l--){
            for(int j=n; j>=i*(k-l); j--){
                DP2[k][j] += DP2[l][j-i*(k-l)] * v[k-l-1] % MOD;
                DP2[k][j] %= MOD;
            }
        }
    }
}
```

# E. Bridge Construction : $O(N^2)$

- 답은 $DP2[4][n-1]$과 같습니다. 눈치채셨을 지도 모르겠지만, 센트로이드와 연결된 서브트리는 크기가 $\frac{n}{2}$ 이하여야 하기 때문에, DP 배열을 딱 $\frac{n}{2}$까지만 업데이트해 줘야 합니다.
- 다만 $n$이 짝수인 경우에는 센트로이드가 두 개인 경우도 고려해 줘야 하는데, 이 경우는 답에 $\binom{DP2[3]\left[\frac{n}{2}-1\right]}{2}$을 더합니다.
- 시간 복잡도는 $O(N^2)$입니다.

# F. 방역

- 만점자: 2명
- 가장 첫 만점: cki86201 (774분)
- 출제자: blackking26

- 특별상: cki86201

# G. 대안적 사실

- 만점자: 1명
- 가장 첫 만점: ainta (972분)
- 출제자: blackking26

- 특별상: ainta

# H. Tree and Easy Queries

- 만점자: 5명
- 가장 첫 만점: cki86201 (41분)
- 출제자: 79brue

- 특별상: ainta

# H. Tree and Easy Queries:

- 해법이 다양하며, 출제진 및 검수진이 검수에 사용한 해법은 3가지입니다.
- 첫 번째 해법은 Centroid Decomposition + Euler Tour Trick + Segment Tree with Lazy Propagation을 이용한 풀이입니다. 시간 복잡도는 $O(N \log N + Q \log^2 N)$입니다.
- Centroid Decomposition Tree를 만든 뒤, 트리의 각 정점별로 세그먼트 트리를 배정해 쿼리를 처리하면 가능합니다.

# H. Tree and Easy Queries

- 앞에서 설명한 방법 이외에도 Link-cut Tree를 이용한 방법이 있으며, 시간복잡도는 앞의 해법과 비슷하지만 상수가 더 작기 때문에 실제 수행 시간은 훨씬 빠릅니다.
- 또 Dynamic Diameter (BOJ 17642, CEOI 2019) 의 $O(Q \log N)$ 해법을 응용해서 $O(Q \log N)$의 시간복잡도로 문제를 푸는 방법도 존재합니다.
- 시간제한은 2.5초로, 첫 번째 방법을 최적화하거나, 두 번째 또는 세 번째 방법을 이용하면 AC를 받을 수 있습니다.

blackking26 주: 첫 번째 방법은 최적화를 조금 '많이' 해야 합니다.

# I. 와일드 카드

- 만점자: 없음
- 가장 첫 만점: 없음
- 출제자: blackking26

- 특별상: 없음