2022 선린 정보 알고리즘경시대회

Official Solutions

출제 및 검수

✓ 김준겸	ryute	구글 코리아	고려대학교 컴퓨터학과
✓ 김준서	junseo		한양대학교 컴퓨터소프트웨어학부
✓ 나정휘	jhnah917		숭실대학교 컴퓨터학부
✓ 박세훈	prarie		성균관대학교 소프트웨어학과
✓ 박찬솔	chansol		숭실대학교 컴퓨터학부
✓ 심준	wesley2003		국민대학교 소프트웨어학부
✓ 오주원	kyo20111		숭실대학교 소프트웨어학부
✓ 윤시우	cgiosy		서울사이버대학교 컴퓨터공학과
✓ 이성현	hibye1217		한양대학교 컴퓨터소프트웨어학부
✓ 정현서	jhwest2		서울대학교 컴퓨터공학부
✓ 최준석	stonejjuin03		고려대학교

Sponsors

문제		의도한 난이도	출제자
A	재귀의 귀재	**Easy**	나정휘
B	등차수열? 등비수열?	**Easy**	오주원
C	순열 뒤집기	**Medium**	이성현
D	최적 경로와 쿼리	**Medium**	나정휘
E	공통 부분 문자열 쿼리	**Hard**	정현서

A. 재귀의 귀재

`recursion, implementation`
출제진 의도 – **Easy**

- ✓ 처음 푼 사람: **장태환**, 2분
- ✓ 처음 푼 사람(Open Contest): **man_of_learning**, 2분
- ✓ 출제자: 나정휘

A. 재귀의 귀재

- ✓ 문제 지문에서 제공한 코드를 그대로 사용하면 됩니다.

B. 등차수열? 등비수열?

`math`, `hash_set`
출제진 의도 – **Easy**

- ✓ 처음 푼 사람: **장태환**, 11분
- ✓ 처음 푼 사람(Open Contest): **hyperbolic**, 7분
- ✓ 출제자: 오주원

B. 등차수열? 등비수열?

- ✓ 수열이 양의 등차 수열인지 판별하는 것은 인접한 항의 차이가 모두 동일한지 확인하면 됩니다.
- ✓ 인접한 항의 차이를 key-value 구조로 관리하면 효율적으로 해결할 수 있습니다.
- ✓ 공비가 1인 등비 수열은 모든 항이 동일합니다.
- ✓ 그렇지 않은 경우, 등비 수열의 길이는 최대 $O(\log \max A_i)$ 입니다.
- ✓ 따라서 최대 60개의 항만 확인해도 양의 등비 수열인지 판별할 수 있습니다.

C. 순열 뒤집기

ad_hoc
출제진 의도 – **Medium**

- ✓ 처음 푼 사람: **장태환**, 25분
- ✓ 처음 푼 사람(Open Contest): **aeren**, 16분
- ✓ 출제자: 이성현

C. 순열 뒤집기

- ✓ 뒤집는 구간들의 포함 관계는 트리 형태로 표현할 수 있습니다.
- ✓ 따라서 인접한 두 구간을 합치는 것을 반복해 모든 수열을 합칠 수 있는지 판별하면 됩니다.
- ✓ 두 구간에 속한 수들의 범위가 각각 $[min_l, max_l], [min_r, max_r]$ 이라고 합시다.
- ✓ 이때 두 구간은 각각 오름차순 또는 내림차순으로 정렬되어 있으므로
  $max_l + 1 = min_r$ 이거나 $max_r + 1 = min_l$ 이면 두 구간을 합칠 수 있습니다.
- ✓ Stack을 사용하면 $O(N)$, Union-Find를 사용하면 $O(N \log N)$ 에 해결할 수 있습니다.

D. 최적 경로와 쿼리

`graphs, dp, sqrt_decomposition`
출제진 의도 – **Medium**

- ✓ 처음 푼 사람: **장태환**, 120분
- ✓ 처음 푼 사람(Open Contest): **edenooo**, 103분
- ✓ 출제자: 나정휘

D. 최적 경로와 쿼리

- ✓ 간선을 1개만 사용하는 경우는 쉽게 해결할 수 있습니다.
- ✓ 간선을 2개 또는 3개를 사용하는 경우의 답을 구하는 가장 단순한 방법은 다음과 같습니다.
  - 쿼리로 주어진 두 정점 u, v의 모든 이웃 c_u, c_v를 보면서
  - $c_u = c_v$이면 간선을 2개 사용하는 경우
  - 간선 (c_u, c_v)가 존재하면 간선을 3개 사용하는 경우
- ✓ 시간 복잡도는 $O(deg(u) \times deg(v))$로 시간 초과를 받게 됩니다.
- ✓ 하지만 정점의 차수가 아주 작을 때는 이 풀이가 효율적이기 때문에
- ✓ 차수가 큰 몇 개의 정점만 다른 방식으로 해결하는 것을 시도해 볼 수 있습니다.

D. 최적 경로와 쿼리

- ✓ 적당한 상수 K에 대해, 차수가 K 초과인 정점을 Large, 이하인 정점을 Small이라고 합시다.
- ✓ 정점들의 차수를 모두 더하면 $2M$이기 때문에, Large 정점은 최대 $2M/K$개 존재합니다.
- ✓ 쿼리로 주어진 두 정점이 모두 Small이라면 $O(K^2)$에 문제를 해결할 수 있습니다.
- ✓ 따라서 두 정점이 모두 Small인 쿼리를 모두 처리하는데 $O(QK^2)$이 걸립니다.

D. 최적 경로와 쿼리

- ✓ 일반성을 잃지 않고, $deg(u) > deg(v)$ 이고, 쿼리 (u, v) 가 모두 다르다고 가정합시다.
- ✓ 만약 쿼리를 $deg(v)$ 에 비례하는 시간에 해결할 수 있다면
- ✓ 정점 u 에 대해서 처리해야 하는 v 가 모두 다르므로
- ✓ u 에 대한 모든 쿼리를 $O(M)$ 에 해결할 수 있습니다.
- ✓ Large 정점은 최대 $O(M/K)$ 개이므로 시간 복잡도는 $O(M^2/K)$ 입니다.
- ✓ 쿼리를 $deg(v)$ 에 비례하는 시간에 해결하는 방법을 알아봅시다.

D. 최적 경로와 쿼리

- ✓ $deg(v)$에 비례하는 시간에 쿼리를 해결하기 위해서는 v의 이웃만 고려해야 합니다.
- ✓ u에서 간선을 2개 이하만 사용해서 갈 수 있는 정점들을 미리 전처리하면
- ✓ v의 이웃에서 전처리된 정점으로 간선이 있는지만 확인하면 정답을 구할 수 있습니다.
- ✓ 전처리는 DP나 SPFA를 이용해 각 정점마다 $O(M)$에 할 수 있습니다.
- ✓ Large 정점은 최대 $O(M/K)$개 존재하므로 전처리하는데 $O(M^2/K)$ 만큼 걸립니다.
- ✓ 따라서 u가 Large 정점인 경우, 모든 쿼리를 $O(M^2/K)$에 처리할 수 있습니다.

D. 최적 경로와 쿼리

- ✓ 두 정점이 모두 Small이면 $O(QK^2)$, 하나라도 Large이면 $O(M^2/K)$ 입니다.
- ✓ 따라서 전체 시간 복잡도는 $O(QK^2 + M^2/K)$ 입니다.
- ✓ $K = M^{1/3}$ 으로 잡으면 $O(QM^{2/3} + M^{5/3})$ 이 되어서 문제를 해결할 수 있습니다.

E. 공통 부분 문자열 쿼리

`suffix_array`, `sparse_table`, `sqrt_decomposition`
출제진 의도 – **Hard**

- ✓ 처음 푼 사람: **N/A**, N/A분
- ✓ 처음 푼 사람(Open Contest): **aeren**, 91분
- ✓ 출제자: 정현서

E. 공통 부분 문자열 쿼리

- ✓ 주어진 문자열을 $S_1, S_2, \cdots, S_N$ 이라고 할 때
  $S = S_1 + \# + S_2 + \# + \cdots + \# + S_N$ 이라고 정의합시다.
- ✓ S 의 접미사들을 사전순으로 정렬했을 때 i 번째로 오는 접미사를 $sa[i]$
  $sa[i-1]$ 과 $sa[i]$ 의 최장 공통 접미사를 $lcp[i]$ 라고 정의합시다. (0-based)
- ✓ sa, lcp 배열은 $O(|S| \log^2 |S|)$ 또는 $O(|S| \log |S|)$ 에 구할 수 있습니다.
- ✓ 쿼리로 주어진 두 문자열 S_i 와 S_j 가 같다면 정답은 $|S_i|(|S_i|+1)/2 - \sum lcp[i]$ 입니다.

E. 공통 부분 문자열 쿼리

- ✓ $N = 2$ 이면 $S = S_1 + \# + S_2$ 입니다. 쿼리로 (S_1, S_2) 가 주어진 경우를 생각해 봅시다.
- ✓ 만약 $sa[i-1] < |S_1|$ and $sa[i] > |S_1|$ 이면 $S_1[sa[i-1]]$ 에서 시작하는 길이가 $lcp[i]$ 인 공통 부분 문자열이 존재합니다.
- ✓ $sa[i-1] > |S_1|$ and $sa[i] < |S_1|$ 인 경우에는 $S_1[sa[i]]$ 에서 시작하는 문자열이 존재합니다.
- ✓ 조건을 만족하는 $(\min(sa[i-1], sa[i]), lcp[i])$ 는 $O(|S_1| + |S_2|)$ 에 모두 구할 수 있습니다.
- ✓ $\min(sa[i-1], sa[i])$ 가 증가하는 순서대로 $cand[0], cand[1], \cdots$ 라고 합시다.

E. 공통 부분 문자열 쿼리

- 단순히 $cand[i][1]$ 들의 합을 구하면 정답을 구할 수 없습니다.
- 여러 후보가 공통으로 갖는 문자열이 있기 때문입니다.
- 두 후보 $cand[i-1], cand[i]$ 가 공통으로 갖는 문자열은 $cand[i]$ 에서 반영하면 안 됩니다.
- 두 후보가 공통으로 갖고 있는 문자열의 길이 c 는 다음과 같이 계산할 수 있습니다.

$$c = \min(cand[i-1][1], cand[i][1], lcp(cand[i-1][0], cand[i][0]))$$ 입니다.

- 따라서 i 번째 후보는 정답에 $cand[i][1] - c$ 만큼 기여합니다.
- $lcp(s,t)$ 는 s 와 t 의 최장 공통 접두사의 길이로, $lcp[s+1 \cdots t]$ 의 최솟값과 동일합니다.
- Sparse Table을 이용하면 구간의 최솟값을 $O(1)$ 에 구할 수 있습니다.
- 따라서 쿼리를 $O(|S_1| + |S_2|)$ 에 처리할 수 있습니다.
- KOI 2021 중등부 4번 공통 괄호 문자열 사전 문제를 비슷한 방식으로 해결할 수 있습니다.

E. 공통 부분 문자열 쿼리

- ✓ 이제 $N > 2$ 이고, 쿼리로 (S_i, S_j) 가 주어진 경우를 생각해 봅시다.
- ✓ S 의 두 접미사 s, t 의 최장 공통 접두사는 lcp 배열의 구간 최솟값으로 구할 수 있습니다.
- ✓ 따라서 sa 배열에서 S_i 의 접미사와 앞/뒤에서 가장 가까운 S_i/S_j 의 접미사를 구할 수 있다면 동일한 방법으로 해결할 수 있습니다.
- ✓ $idx[i]$ 를 S_i 의 접미사의 sa 배열에서의 위치라고 합시다.
- ✓ S_i 의 접미사 $idx[i][s]$ 와 가장 가까운 S_i 의 접미사는 $idx[i][s-1], idx[i][s+1]$ 입니다.
- ✓ $idx[i][s]$ 와 가장 가까운 S_j 의 접미사는 $idx[j]$ 에서 이분 탐색을 이용해 구할 수 있습니다.
- ✓ 따라서 쿼리를 $O(|S_i| \log |S_j|)$ 에 해결할 수 있습니다.

E. 공통 부분 문자열 쿼리

- 시간 복잡도가 $O(|S_i| \log |S_j|)$ 이므로 일반성을 잃지 않고 $|S_i| \leq |S_j|$ 이라고 생각합시다.
- 만약 중복된 쿼리가 없다면 전체 시간 복잡도는 $O(|S|\sqrt{|S|} \log |S|)$ 입니다.
- 중복된 쿼리가 주어지는 경우, 이전에 계산한 정답을 그대로 반환하면 됩니다.
- 시간 복잡도는 $|S_i| \leq \sqrt{|S|}$ 인 경우와 아닌 경우로 나눠서 증명할 수 있습니다.
- 하지만 더 최적화해야 합니다.

E. 공통 부분 문자열 쿼리

- $idx[j]$ 배열에서 이분 탐색을 하기 때문에 시간 복잡도에 log가 붙게 됩니다.
- 만약 $|S_j| \leq \sqrt{|S|}$ 이면 투 포인터로 $O(|S_1| + |S_2|) \leq O(\sqrt{|S|})$ 에 해결할 수 있습니다.
- 그렇지 않은 경우, $|S_j| > \sqrt{|S|}$ 인 문자열은 최대 $\sqrt{|S|}$ 개입니다.
- 각 지점의 lower bound 값을 문자열마다 $O(|S|)$, 총 $O(|S|\sqrt{|S|})$ 에 전처리하면
- 각 쿼리를 $O(|S_i|)$ 시간에 해결할 수 있습니다.
- 따라서 전체 시간 복잡도는 $O(|S|\sqrt{|S|})$ 입니다.
- 이 밖에도 Suffix Automaton 등을 이용해 $O(|S|\sqrt{|S|})$ 에 해결하는 풀이도 있습니다.

- ✓ 모범 코드는 https://github.com/justiceHui/Sunrin-Contest/tree/main/Sunrin-OI-2022 에서 확인할 수 있습니다.
- ✓ 감사합니다.