Sqrt Decomposition

구재현 (gs14004)

Table of Contents

Query Sorting 1

- Problem : http://koistudy.net/?mid=prob_page&NO=1069

- Naive = O(QN) (Counting Sort)
- 쿼리마다 큰 원소 기준으로 카운팅 정렬
- 그 후, Sum(a[i] * i) 를 계산하면 됨 (a[i] < a[i+1])
- 데이터 약해서 이걸로 AC가 뜸 (…)

Query Sorting 2

- 쿼리 정렬을 하기 전에, [L,R] -> [L,R+1]을 lgN에 할 수 있는지에 대해서 고민해 보자. (반대도)
- 방법은 혼자 생각해 보길.. 할 수 있다.
- 이제 다음 함수로 쿼리를 정렬하자. (SQ = sqrt(n))
- Cmp(Query a, Query b) :
- return a.s / SQ == b.s / SQ ? (a.e < b.e) : (a.s < b.s)

Query Sorting 3

- 앞서 집합 [L,R] 에서 앞 뒤 원소를 빼거나 더하는 것이 lgN 이라 언급했다.

- 쿼리 [L1,R1] -> 쿼리 [L2,R2] 로 움직이는 것은 그러한 연산들을 몇번씩 반복하면 쉽게 가능하다.

- 이제, 앞 함수 같이 쿼리를 정렬하면 **그러한 연산이 총 N*Q^0.5 번만 이루어 진다**는 것을 증명하겠다.

Query Sorting 4

- End 포인트의 이동을 보자.
- N / SQRT번 1 ~ N으로 이동한다.
- Start 포인트의 이동을 보자.
- 한번 1 ~ N으로 이동하며, Q번동안 최대 SQRT 거리만큼 변화를 가진다.
- SQRT = N / Q^0.5~~~
- **Q * SQRT + N / SQRT * N = N * Q ^ 0.5**

Query Sorting 5

- 이 아름다운 Sqrt Decomposition 알고리즘은 오프라인 쿼리 처리에서 **극단적으로** 강력한 방법 중 하나다.
- 농담이 아니라, 쿼리 문제는 어지간하면 Naive가 NQ^0.5lgN 일 정도.
- 거기다가 이 방법을 쓰면 어지간하면 버킷을 구현할 필요가 없어서 많은 버킷 문제가 산으로 갔다.
- 알아만 두면 코드포스 등지에서 **개꿀빨수 있다!**
- ~~물론 멀쩡한 문제 쿼리 정렬로 풀고 TLE 볼수도 있다.~~

Query Sorting 6

- 연습 문제가 좀 많다. 3/4/5는 또 보게 될거니 꼭 고민해보길.
- http://koistudy.net/?mid=prob_page&NO=1069 (easy)
- http://codeforces.com/problemset/problem/86/D (easy)
- http://oj.uz/problems/statistics/JOI14_historical (medium)
- https://www.acmicpc.net/problem/7469 (medium)
- http://codeforces.com/problemset/problem/375/D (hard)

Query Caching 1

- http://koistudy.net/?mid=prob_page&NO=1072
- Naive = O(QNlgN)
- 너무 느린 Naive를 보고 빡친 재민이는 map으로 중복 쿼리를 잡기로 결심했다.
- **그런데 AC가 떴다????**

Query Caching 2

- 각설하고 문제 풀이부터 알려주겠다.
- 1) 큰 집합 A, 작은 집합 B를 고려하고 있다면, 쿼리를 O(BlgA) 에 처리해야만 한다.
- 2) 중복 쿼리는 map을 사용해서 메모이제이션 한다.
- **복잡도는 O(NQ^0.5lgN) 이다.**

Query Caching 3

- 1) 작은 집합의 크기가 B 이하이다
- - 이 때 연산이 BlgN 이하임은 자명하다.
- 2) 작은 집합의 크기가 B 이상이다.
- - 큰 집합은 많아야 N/B 개 존재한다.
- - (N/B) * (N/B) 개의 계산 시간을 모두 더해보자. 이러한 연산의 계산 복잡도의 총 합은 N^2/BlgN 이다.
- => O(QBlgN + N^2/BlgN) = O(NQ^0.5lgN)

Query Caching 4

- 1번 방법만큼 유명하고 유용하지는 않지만 여전히 강력한 방법이다.
- 일반적으로 서로소 합집합의 크기가 적당히 bounded 되어 있을때 자주 사용하는 방법.
- 2차선발 4번 풀이가 이게 아닐까 추측한다.
- http://koistudy.net/?mid=prob_page&NO=1072 (easy)
- http://codeforces.com/problemset/problem/506/D (medium)
- http://183.106.113.109/pool/ioi_regions/ioi_regions.php?pname=ioi_regions (hard)

Explicit Bucket 1

- http://koistudy.net/?mid=prob_page&NO=1049
- 쿼리 정렬이 상대적으로 버킷보다 느리다. (아님말구..)

- 버킷은 기본적으로 앞 테크닉들의 상위 호환이라, 다 안 되면 이걸로 짜야 됨.

- 사실 나도 이걸 잘 못 짜서 많은 걸 설명하진 못한다.

Explicit Bucket 2

- 풀이를 설명하긴 그렇고 그냥 힌트만 주고 끝내려 한다.
- 버킷을 사용해서 쿼리마다 SQRT개의 원소만을 볼 수 있게 하는 방법이 무엇일까?
- 버킷이 후보를 N개에서 SQRT개로 줄이는 데 무슨 역할을 할까?
- 잘 생각해보세요..
- 연습 문제 : http://www.codechef.com/problems/QCHEF
- (난 O(NQ^0.5lg^0.5N) 에 풀었지만, O(NQ^0.5) 가 된다고 함)