Sqrt Decomposition

나정휘

https://justiceHui.github.io/

목차

- 산술 기하 부등식
- 수열에 대한 버킷
- 쿼리 캐싱
- 쿼리에 대한 버킷

산술 기하 부등식

산술 기하 부등식

- 음이 아닌 실수 $x_1, x_2, \cdots, x_n$ 에 대해 아래 식이 성립함
  - $\frac{x_1+x_2\cdots+x_n}{n} \geq \sqrt[n]{x_1 x_2 \cdots x_n}$
  - 등호가 성립할 조건은 $x_1 = x_2 = \cdots = x_n$

- $n = 2$ 인 경우
  - $a + b \geq 2\sqrt{ab}$
  - $a = b$ 일 때 등호 성립
  - 두 식의 곱이 일정할 때 합의 최솟값을 찾을 수 있음

산술 기하 부등식

- 간단한 정렬 알고리즘
  - 길이가 N인 배열을 정렬하는 알고리즘
    - 배열을 크기가 B인 N/B개의 덩어리로 분할
    - 각 덩어리를 $O(N^2)$ 알고리즘으로 정렬
    - N/B개의 덩어리는 각각 정렬되어 있는 상태

    - 각 덩어리의 맨 앞에 있는 원소들의 최솟값을 정답에 추가
    - N번 반복하면 전체 배열이 정렬됨

  - 시간 복잡도는?

산술 기하 부등식

- 간단한 정렬 알고리즘 – 시간 복잡도
  - 길이가 N인 배열을 정렬하는 알고리즘
    - 배열을 크기가 B인 N/B개의 덩어리로 분할
    - 각 덩어리를 $O(N^2)$ 알고리즘으로 정렬 $B^2 * N/B = NB$
    - N/B개의 덩어리는 각각 정렬되어 있는 상태

    - 각 덩어리의 맨 앞에 있는 원소들의 최솟값을 정답에 추가 $N/B * N = N^2/B$
    - N번 반복하면 전체 배열이 정렬됨

  - 시간 복잡도는? $NB + N^2/B$

산술 기하 부등식

- 간단한 정렬 알고리즘 – 시간 복잡도
  - 길이가 N인 배열을 정렬하는 알고리즘
    - 배열을 크기가 B인 N/B개의 덩어리로 분할
    - 각 덩어리를 $O(N^2)$ 알고리즘으로 정렬 — $B^2 * N/B = NB$
    - N/B개의 덩어리는 각각 정렬되어 있는 상태
    - 각 덩어리의 맨 앞에 있는 원소들의 최솟값을 정답에 추가 — $N/B * N = N^2/B$
    - N번 반복하면 전체 배열이 정렬됨
  - 시간 복잡도는? — $NB + N^2/B$
    - 곱이 일정하므로 $NB = N^2/B$일 때 최소가 됨
    - $B = \sqrt{N}$ 이면 전체 시간 복잡도는 $O(N\sqrt{N})$

산술 기하 부등식

- 곱이 일정하지 않은 경우
  - 대부분의 문제는 각 변수의 최댓값끼리 큰 차이가 없음
    - ex. $N \le 50'000$, $M \le 100'000$, $Q \le 100'000$
  - 두 식의 차수를 동일하게 만든다고 생각해도 됨
    - ex. 시간 복잡도가 $O(QB^2 + M^2/B)$ 이면 $B = M^{1/3}$일 때 $O(QM^{2/3} + M^{5/3})$, 각 항의 차수는 5/3

질문?

수열에 대한 버킷

수열에 대한 버킷

- BOJ 14438 수열과 쿼리 17
  - 1: A_i를 v로 바꾼다.
  - 2: A_i, A_{i+1}, … , A_j에서 크기가 가장 작은 값을 출력한다.

  - 세그먼트 트리로 풀 수 있지만 다른 방법으로 풀어보자.

수열에 대한 버킷

- BOJ 14438 수열과 쿼리 17
  - 정렬 알고리즘에서 본 방식을 사용해 보자.
    - 배열을 크기가 B인 N/B개의 버킷으로 분할
    - 각 버킷의 최솟값을 전처리
      - O(N)에 가능
    - 1: 원소의 값을 바꾼 뒤, 그 원소가 속한 버킷의 최솟값을 갱신
      - 버킷의 크기는 B 이므로 O(B)에 가능
    - 2: 구간에 완전히 포함된 버킷은 버킷의 최솟값을 취하고, 일부만 겹친 버킷은 모든 원소를 확인
      - 구간에 완전히 포함된 버킷은 최대 N/B개
      - 일부만 겹친 버킷은 최대 2개이므로 최대 2B개의 버킷의 원소를 확인
      - O(N/B + 2B)
  - B = √N 이면 전체 시간 복잡도는 O(N + Q√N)

수열에 대한 버킷

- BOJ 10999 구간 합 구하기 2
  - 1: A_i, A_{i+1}, … , A_j에 v를 더한다.
  - 2: A_i, A_{i+1}, … , A_j의 합을 출력한다.

  - Lazy Propagation도 수행할 수 있을까?

수열에 대한 버킷

- BOJ 10999 구간 합 구하기 2
  - 똑같이 크기가 B인 버킷을 만들어 보자.
    - 구간에 v를 더하는 쿼리
      - 구간에 완전히 포함되는 버킷에는 v라는 태그를 달아 놓고, 버킷의 합을 vB 만큼 증가
      - 구간과 일부만 겹치는 버킷은 모든 원소를 확인하면서 구간에 포함되는 원소에 v를 더함
    - 구간의 합을 구하는 쿼리
      - 구간에 완전히 포함되는 버킷은 버킷의 합을 취함
      - 구간과 일부만 겹치는 버킷은
        - 버킷에 달려 있는 태그를 원소들에게 뿌려준 다음
        - 버킷의 모든 원소를 보면서 구간에 포함되는 원소의 값을 취함

수열에 대한 버킷

- BOJ 10999 구간 합 구하기 2
  - 똑같이 크기가 B인 버킷을 만들어 보자.
    - 구간에 v를 더하는 쿼리
      - 구간에 완전히 포함되는 버킷에는 v라는 태그를 달아 놓고, 버킷의 합을 vB 만큼 증가 O(N/B)
      - 구간과 일부만 겹치는 버킷은 모든 원소를 확인하면서 구간에 포함되는 원소에 v를 더함 O(B)
    - 구간의 합을 구하는 쿼리
      - 구간에 완전히 포함되는 버킷은 버킷의 합을 취함 O(N/B)
      - 구간과 일부만 겹치는 버킷은
        - 버킷에 달려 있는 태그를 원소들에게 뿌려준 다음 O(B)
        - 버킷의 모든 원소를 보면서 구간에 포함되는 원소의 값을 취함 O(B)
  - B = √N 이면 O(N + Q√N)

수열에 대한 버킷

```cpp
ll N, M, K, A[1'000'000], S[1000], L[1000];

void Update(int l, int r, ll v){
    for(int i=0; i<N; i+=B){
        int j = i + B - 1, buc = i / B;
        if(r < i || j < l) continue;
        else if(l <= i && j <= r) S[buc] += v * B, L[buc] += v;
        else for(int k=i; k<=j; k++) if(l <= k && k <= r) A[k] += v, S[buc] += v;
    }
}

ll Query(int l, int r){
    ll res = 0;
    for(int i=0; i<N; i+=B){
        int j = i + B - 1, buc = i / B;
        if(r < i || j < l) continue;
        else if(l <= i && j <= r) res += S[buc];
        else{
            for(int k=i; k<=j; k++) A[k] += L[buc];
            for(int k=i; k<=j; k++) if(l <= k && k <= r) res += A[k];
            L[buc] = 0;
        }
    }
    return res;
}
```

수열에 대한 버킷

```cpp
ll N, M, K, A[1'000'000], S[1000], L[1000];

void Update(int l, int r, ll v){
    while(l <= r && l % B != 0) S[l/B] += v, A[l++] += v;
    while(l <= r && r % B + 1 != B) S[r/B] += v, A[r--] += v;
    if(l <= r) for(int i=l/B; i<=r/B; i++) S[i] += v * B, L[i] += v;
}

void Push(int buc){
    for(int i=0; i<B; i++) A[buc*B+i] += L[buc];
    L[buc] = 0;
}

ll Query(int l, int r){
    ll res = 0;
    if(l % B != 0) Push(l/B);
    if(r % B + 1 != B) Push(r/B);
    while(l <= r && l % B != 0) res += A[l++];
    while(l <= r && r % B + 1 != B) res += A[r--];
    if(l <= r) for(int i=l/B; i<=r/B; i++) res += S[i];
    return res;
}
```

질문?

수열에 대한 버킷

- 연습 문제
  - 각종 세그먼트 트리 연습 문제
  - BOJ 23238 Best Student
  - BOJ 15055 Daunting device
  - Mo's Algorithm 공부하고 연습 문제 풀기
    - BOJ 13547 수열과 쿼리 5
    - BOJ 13548 수열과 쿼리 6
    - BOJ 13546 수열과 쿼리 4

쿼리 캐싱

쿼리 캐싱

- BOJ 12857 홍준이는 문자열을 좋아해
  - 문자열 S가 주어졌을 때 다음과 같은 쿼리를 처리해야 함
    - 쿼리: 문자열 A, B가 주어지면 A와 B를 모두 포함하는 S의 가장 짧은 부분 문자열의 길이 출력
  - $|S| \le 50'000$, $Q \le 100'000$
  - $|A|, |B| \le 4$, 문자열은 모두 알파벳 소문자로 구성

  - 길이가 4 이하인 S의 부분 문자열은 최대 $4N \le 200'000$개
  - 길이가 4 이하인 문자열은 $27^4 = 531'441$가지

  - 각 문자열이 S에서 등장하는 위치를 모두 전처리하자.
    - 길이가 4 이하인 문자열을 100만 이하의 정수로 해싱하는 함수를 사용
    - O(4N)에 전처리 가능

쿼리 캐싱

- BOJ 12857 홍준이는 문자열을 좋아해
  - P_h = 해시값이 h인 문자열이 S에서 등장하는 위치를 저장한 배열

  - 두 문자열 A, B가 주어졌을 때 정답을 구하는 건 $O(|P_A| \log |P_B|)$에 가능
    - P_A의 모든 원소 $[s_i, e_i]$를 차례대로 보면서
    - 시작점이 s_i 이후인 가장 빠른 P_B의 원소 $[s_j, e_j]$
    - 시작점이 s_i 이전인 가장 늦은 P_B의 원소 $[s_k, e_k]$
    - 이분 탐색으로 찾을 수 있음

  - 시간 복잡도는 어떻게 될까?

쿼리 캐싱

- BOJ 12857 홍준이는 문자열을 좋아해
  - 각 쿼리는 $O(\min(A_i,A_j) \log \max(A_i,A_j))$에 처리할 수 있고 $\text{sum}(A) = O(N)$, $A_i \geq 0$
  - 중복 쿼리가 없고 $\text{sum}(A) = N$이면 $\min(A_i,A_j)$의 합이 $O((N+Q)\sqrt{N})$임을 보일 수 있음
    - 일반성을 잃지 않고 $a = A_i \leq A_j = b$ 라고 하자.
    - case 1. $a \leq \sqrt{N}$
      - 각 쿼리마다 $\min(a, b) = O(\sqrt{N})$ 이므로 쿼리 Q개 처리하면 $O(Q\sqrt{N})$
    - case 2. $a,b > \sqrt{N}$
      - 크기가 $\sqrt{N}$보다 큰 원소는 최대 $N/\sqrt{N} = \sqrt{N}$개 존재하므로 b로 가능한 원소는 최대 $\sqrt{N}$가지
      - 쿼리로 (a, b)가 주어지는데, 이때 b마다 주어지는 a들은 서로 다르므로 a의 합은 최대 N
      - (b의 종류) * (a의 합) = $O(\sqrt{N}) * O(N) = O(N\sqrt{N})$
    - 따라서 $O((N+Q)\sqrt{N})$

쿼리 캐싱

- BOJ 12857 홍준이는 문자열을 좋아해
  - 각 쿼리는 $O(\min(A_i,A_j) \log \max(A_i,A_j))$에 처리할 수 있고 $\text{sum}(A) = O(N)$, $A_i \geq 0$
  - 중복 쿼리가 없으면
    - $\min(A_i,A_j)$의 합이 $O((N+Q)\sqrt{N})$
    - 전체 시간 복잡도는 $O((N+Q)\sqrt{N} \log N)$
  - 중복 쿼리가 있으면
    - 쿼리의 결과를 저장
    - std::map 쓰면 $O(Q \log \min(N^2,Q)) \leq O(2Q \log N)$이 추가로 붙음
    - 전체 시간 복잡도는 $O((N+Q)\sqrt{N} \log N)$
  - 꼭 쿼리 캐싱이 아니더라도 이런 분석 기법은 자주 쓰임

질문?

쿼리 캐싱

- BOJ 12857 홍준이는 문자열을 좋아해
  - 시간 복잡도에서 로그 뗄 수 있음

  - 중복 쿼리가 없는 경우
    - 일반성을 잃지 않고 $a = A_i \le A_j = b$ 라고 하자.
    - case 1. $b \le \sqrt{N}$
      - 투 포인터를 이용해 $O(a+b) \le O(\sqrt{N})$에 처리할 수 있으므로 총 $O(Q\sqrt{N})$
    - case 2. $b > \sqrt{N}$
      - $\sqrt{N}$보다 큰 원소는 최대 $N/\sqrt{N} = \sqrt{N}$가지
      - 각각의 b에 대해, 1..N에서의 lower bound 값을 전처리
        - 각 배열을 O(N)에 전처리할 수 있으므로 시간 복잡도와 공간 복잡도 모두 $O(N\sqrt{N})$
      - lower bound 대신 전처리 테이블 사용하면 각 쿼리를 O(a)에 처리할 수 있음
      - 중복 쿼리가 없으므로 각각의 b에 대해 a의 합은 O(N)이므로 총 $O(N\sqrt{N})$

쿼리 캐싱

- BOJ 1762 평면그래프와 삼각형
  - 그래프에서 크기가 3인 사이클의 개수를 구하는 문제
  - $N \le 100'000$, $M \le 300'000$

  - 두 정점 a, b를 고정하자.
    - 두 정점을 고정하는 방법은 총 M가지
    - G[a]와 G[b]의 교집합의 크기를 구하면 됨
    - 이분 탐색 사용하면 쿼리마다 $O(\min(S_a,S_b) \log \max(S_a,S_b))$

  - 전체 시간 복잡도는 $O(M \sqrt{M} \log N)$

쿼리 캐싱

```cpp
#include <bits/stdc++.h>
using namespace std;

int N, M, R;
vector<int> G[101010];

int main(){
    ios_base::sync_with_stdio(false); cin.tie(nullptr);
    cin >> N >> M;
    for(int i=1,u,v; i<=M; i++) cin >> u >> v, G[u].push_back(v), G[v].push_back(u);
    for(int i=1; i<=N; i++) sort(G[i].begin(), G[i].end());

    for(int i=1; i<=N; i++){
        for(auto j : G[i]){
            if(i < j) break;
            int u = i, v = j;
            if(G[u].size() > G[v].size()) swap(u, v);
            for(auto k : G[u]) R += binary_search(G[v].begin(), G[v].end(), k);
        }
    }
    cout << R / 3;
}
```

쿼리 캐싱

- BOJ 1762 평면그래프와 삼각형
  - 사실 O(M log N)에 풀 수도 있음
    - V ≥ 3인 평면 그래프는 E ≤ 3V – 6를 만족함
    - sum(degree) ≤ 6V-12
    - 따라서 min(degree) ≤ 5

    - 차수가 5 이하인 정점 v가 항상 존재하므로
    - v와 연결된 두 정점이 서로 연결되어 있는지 확인하면 됨
    - 우선순위 큐/BBST 등으로 차수가 최소인 정점 구하면 O(M log N)

질문?

쿼리 캐싱

- 연습 문제
  - BOJ 12857 홍준이는 문자열을 좋아해
  - BOJ 1762 평면 그래프와 삼각형
  - BOJ 25952 Rectangles
  - BOJ 22316 Regions
    - 방금 소개한 트릭은 regions trick으로 부르기도 함
  - BOJ 25504 최적 경로와 쿼리
  - BOJ 25505 공통 부분 문자열 쿼리

쿼리에 대한 버킷

쿼리에 대한 버킷

- BOJ 17635 다리
  - 무향 그래프에서 다음과 같은 쿼리를 처리해야 함
    - 간선의 무게 제한을 바꾸는 쿼리
    - 무게가 w인 자동차가 정점 s에서 출발해서 도달할 수 있는 정점의 개수
  - N ≤ 50‘000, M ≤ 100’000, Q ≤ 100’000

  - 갱신 쿼리가 없는 경우
    - 쿼리와 간선을 무게 내림차순으로 정렬한 뒤
    - Union-Find를 이용해 오프라인으로 처리

  - 갱신 쿼리가 있는 경우는 어떻게 풀어야 할까?

쿼리에 대한 버킷

```cpp
int N, M, Q, R[101010];
Edge E[101010];
Query Qry[101010];

int main(){
    ios_base::sync_with_stdio(false); cin.tie(nullptr);
    cin >> N >> M;
    for(int i=1; i<=M; i++) cin >> E[i].u >> E[i].v >> E[i].w;
    cin >> Q;
    for(int i=1; i<=Q; i++) cin >> Qry[i].op >> Qry[i].s >> Qry[i].w, Qry[i].idx = i;
    sort(E+1, E+M+1, [](auto e1, auto e2){ return e1.w > e2.w; });
    sort(Qry+1, Qry+Q+1, [](auto q1, auto q2){ return q1.w > q2.w; });

    UnionFind U(N);
    for(int i=1, j=1; i<=Q; i++){
        while(j <= M && Qry[i].w <= E[j].w) U.merge(E[j].u, E[j].v), j++;
        R[Qry[i].idx] = U.size(Qry[i].s);
    }
    for(int i=1; i<=Q; i++) cout << R[i] << "\n";
}
```

쿼리에 대한 버킷

- BOJ 17635 다리
  - 쿼리를 B개씩 묶어서 처리하자.
    - B개의 쿼리 동안 무게 제한이 바뀌는 간선은 최대 B개
      - 무게 제한이 바뀌는 간선을 d-간선, 바뀌지 않는 간선을 s-간선이라고 하자.
    - 1번 쿼리: 단순히 간선의 무게 제한을 바꾸면 됨. 각 버킷에서 최대 B번 수행
    - 2번 쿼리: 무게 내림차순 오프라인 처리
      - s-간선을 미리 구해서 무게 제한 내림차순으로 정렬하자.
      - 투 포인터를 사용해서 현재 자동차가 지나갈 수 있는 s-간선을 모두 Union
      - 자동차가 지나갈 수 있는 d-간선을 모두 구해서 Union
      - Union-Find에 저장된 정보를 이용해 정답을 구하고
      - d-간선 Union한 거 Undo
  - 시간 복잡도는?

쿼리에 대한 버킷

- BOJ 17635 다리
  - 쿼리를 B개씩 묶어서 처리하자.
    - B개의 쿼리 동안 무게 제한이 바뀌는 간선은 최대 B개
      - 무게 제한이 바뀌는 간선을 d-간선, 바뀌지 않는 간선을 s-간선이라고 하자.
    - 1번 쿼리: 단순히 간선의 무게 제한을 바꾸면 됨. 각 버킷에서 최대 B번 수행 $O(1) * B$
    - 2번 쿼리: 무게 내림차순 오프라인 처리
      - s-간선을 미리 구해서 무게 제한 내림차순으로 정렬하자. $O(M \log M)$
      - 투 포인터를 사용해서 현재 자동차가 지나갈 수 있는 s-간선을 모두 Union $O(B + M \log N)$
      - 자동차가 지나갈 수 있는 d-간선을 모두 구해서 Union $O(B \log N) * B$
      - Union-Find에 저장된 정보를 이용해 정답을 구하고 $O(\log N) * B$
      - d-간선 Union한 거 Undo $O(B \log N) * B$
  - 시간 복잡도는?
    - $(N + B^2 \log N + M \log M) * Q/B$
    - $B = \sqrt{Q}$으로 하면 $O(N \sqrt{Q} + Q \sqrt{Q} \log N + M \sqrt{Q} \log M)$

쿼리에 대한 버킷

- BOJ 17635 다리
  - https://oj.uz/submission/672725

질문?

쿼리에 대한 버킷

- 연습 문제
  - BOJ 17635 다리
  - BOJ 5823 코끼리
  - BOJ 18254 쿼리와 쿼리
  - BOJ 16793 Collapse