Two-Pointer & Sliding Window

2019-2020 Winter

20141574 임지환 (Sogang University)

도입

- 양의 정수 n개로 구성된 배열과 양의 정수 x

- 양의 정수 n개로 구성된 배열과 양의 정수 x

range sum이 x가 되는 구간의 경우의 수?

- range sum == x인 경우의 수

Solution 1

$i < j$ 인 구간 $[i, j]$에 대하여 각각의 sum을 $O(j - i)$에 구한다면?

- range sum == x인 경우의 수

Solution 1

$i < j$ 인 구간 $[i, j]$에 대하여 각각의 sum을 $O(j-i)$에 구한다면?

```
int a[mxn], N;
int sum(int i, int j) {
    int ret = 0;
    for (int it = i; it <= j; it++)
        ret += a[it];
    return ret;
}
```

```
int solve(int x) {
    int ret = 0;
    for (int i = 1; i <= N; i++)
        for (int j = i; j <= N; j++)
            if (sum(j, i) == x) ret++;

    return ret;
}
```

- range sum == x인 경우의 수

Solution 1

$i < j$ 인 구간 $[i, j]$에 대하여 각각의 sum을 $O(j - i)$에 구한다면?

```
int a[mxn], N;
int sum(int i, int j) {
    int ret = 0;
    for (int it = i; it <= j; it++)
        ret += a[it];
    return ret;
}
```

```
int solve(int x) {
    int ret = 0;
    for (int i = 1; i <= N; i++)
        for (int j = i; j <= N; j++)
            if (sum(j, i) == x) ret++;

    return ret;
}
```

Total time complexity : $O(N^3)$

- range sum == x인 경우의 수

Solution 2

구간 $[0, i]$에 대한 prefix sum을 전처리한다면?

- range sum == x인 경우의 수

Solution 2

구간 $[0, i]$에 대한 prefix sum을 전처리한다면?

전처리 시간 : $O(N)$

```
int N, a[mxn], psum[mxn];
int sum(int i, int j) {
    return psum[j] - psum[i - 1];
}
```

```
int solve(int x) {
    int ret = 0;
    for (int i = 1; i <= N; i++)
        for (int j = i; j <= N; j++)
            if (sum(j, i) == x) ret++;

    return ret;
}
```

- range sum == x인 경우의 수

Solution 2

구간 $[0, i]$에 대한 prefix sum을 전처리한다면?

전처리 시간 : $O(N)$

```
int N, a[mxn], psum[mxn];
int sum(int i, int j) {
    return psum[j] - psum[i - 1];
}
```

```
int solve(int x) {
    int ret = 0;
    for (int i = 1; i <= N; i++)
        for (int j = i; j <= N; j++)
            if (sum(j, i) == x) ret++;

    return ret;
}
```

Total time complexity : $O(N^2)$

- range sum == x인 경우의 수

N이 커진다면? $(N \leq 10^6)$

- range sum == x인 경우의 수

N이 커진다면? $(N \leq 10^6)$

Solution 3

target : find L, R $s.t$ $psum[R] - psum[L] = x$

- range sum == x인 경우의 수

N이 커진다면? ($N \leq 10^6$)

Solution 3

target : find $L, R\ s.t\ psum[R] - psum[L] = x$

find $psum[R] = x + psum[L]$

- range sum == x인 경우의 수

N이 커진다면? ($N \leq 10^6$)

Solution 3

```
int solve(int x) {
    int ret = 0;
    for (int i = 1; i <= N; i++) {
        int ub = upper_bound(psum + 1, psum + N, psum[i] + x) - (psum + 1);
        int lb = lower_bound(psum + 1, psum + N, psum[i] + x) - (psum + 1);
        if (psum[lb] != psum[i] + x) continue;
        ret += ub - lb;
    }

    return ret;
}
```

Total time complexity : $O(NlogN)$

- range sum == x인 경우의 수

N이 더 커진다면? $(N \leq 10^8)$

- range sum == x인 경우의 수

N이 더 커진다면? $(N \leq 10^8)$

Solution 4

Use Two-pointer method (inchworm)

find x = 8

1	3	2	5	1	1	2	3

- range sum == x인 경우의 수

N이 더 커진다면? ($N \leq 10^8$)

Solution 4

Use Two-pointer method (inchworm)

find x = 8

1	3	2	5	1	1	2	3

- range sum == x인 경우의 수

N이 더 커진다면? $(N \leq 10^8)$

Solution 4

Use Two-pointer method (inchworm)

find x = 8

1	3	2	5	1	1	2	3

- range sum == x인 경우의 수

N이 더 커진다면? ($N \le 10^8$)

Solution 4

Use Two-pointer method (inchworm)

find x = 8

1	3	2	5	1	1	2	3

- range sum == x인 경우의 수

N이 더 커진다면? $(N \leq 10^8)$

Solution 4

Use Two-pointer method (inchworm)

find x = 8

1	3	2	5	1	1	2	3

- range sum == x인 경우의 수

N이 더 커진다면? ($N \le 10^8$)

Solution 4

Use Two-pointer method (inchworm)

find x = 8

1	3	2	5	1	1	2	3

- range sum == x인 경우의 수

N이 더 커진다면? $(N \leq 10^8)$

Solution 4

Use Two-pointer method (inchworm)

find x = 8

1	3	2	5	1	1	2	3

- range sum == x인 경우의 수

N이 더 커진다면? $(N \leq 10^8)$

Solution 4

Use Two-pointer method (inchworm)

find x = 8

1	3	2	5	1	1	2	3

- range sum == x인 경우의 수

N이 더 커진다면? $(N \leq 10^8)$

Solution 4

Use Two-pointer method (inchworm)

find x = 8

1	3	2	5	1	1	2	3

http://boj.kr/72c93efdbfef4caebfefd2049dadb313

#1644 소수의 연속합

- 자연수 N ($1 \le N \le 4{,}000{,}000$)이 주어졌을 때 이를 연속된 소수의 합으로 나타낼 수 있는 경우의 수를 구하여라.

- **문제 해결 과정**

1. 400만 이하의 모든 소수 조합 구하기 (by Sieve of Eratosthenes, etc)
2. 아까 배운 것 적용하기

- Review for previous code :

```
int sum = 0, r = 1, ans = 0;
for (int l = 1; l <= N; l++) {
    if (r == N + 1) break;
    sum -= a[l - 1];
    while (r <= N && sum + a[r] <= x) sum += a[r++];
    if (sum == x) ans++;
}
```

- Review for previous code :

```
int sum = 0, r = 1, ans = 0;
for (int l = 1; l <= N; l++) {
    if (r == N + 1) break;
    sum -= a[l - 1];
    while (r <= N && sum + a[r] <= x) sum += a[r++];
    if (sum == x) ans++;
}
```

1~N for문에 while문까지 있고 r이 0부터 시작하니 $O(N^2)$

- Review for previous code :

```
int sum = 0, r = 1, ans = 0;
for (int l = 1; l <= N; l++) {
    if (r == N + 1) break;
    sum -= a[l - 1];
    while (r <= N && sum + a[r] <= x) sum += a[r++];
    if (sum == x) ans++;
}
```

1~N for문에 while문까지 있고 r이 0부터 시작하니 $O(N^2)$ $O(N)$

Amortized analysis

Asymptotic analysis

- N개의 수로 구성된 배열과 수 x

두 수의 합이 x가 되는 경우의 수?

- 문제 해결 과정

1. $i \neq j$인 $a[i], a[j]$에 대하여 $a[i] + a[j] = x$, 즉 $a[i] = x - a[j]$를 만족하는 j 찾기

- 문제 해결 과정

1. $i \neq j$인 $a[i], a[j]$에 대하여 $a[i] + a[j] = x$, 즉 $a[i] = x - a[j]$를 만족하는 j 찾기
2. 선형시간 search를 할 필요가 없으니 binary search를 위해 정렬하기 $\boldsymbol{O(NlogN)}$

- **문제 해결 과정**

1. $i \neq j$인 $a[i], a[j]$에 대하여 $a[i] + a[j] = x$, 즉 $a[i] = x - a[j]$를 만족하는 j 찾기
2. 선형시간 search를 할 필요가 없으니 binary search를 위해 정렬하기 $\boldsymbol{O(NlogN)}$
3. 모든 i에 대하여, $x - a[j]$를 만족하는 j의 개수 구하기 $\boldsymbol{O(NlogN)}$

- 문제 해결 과정

  by two-pointer method

- **문제 해결 과정**

  by two-pointer method

1. $i \neq j$인 $a[i], a[j]$에 대하여 $a[i] + a[j] = x$, 즉 $a[i] = x - a[j]$를 만족하는 j 찾기
2. 선형시간 search를 할 필요가 없으니 binary search를 위해 정렬하기 $\boldsymbol{O(NlogN)}$
3. Pointer L은 앞쪽부터, Pointer R은 뒤쪽부터 시작하여 탐색하기 $\boldsymbol{O(N)}$

2SUM Problem

- N=6, x=9

2	7	4	1	5	3

2SUM Problem

- N=6, x=9

1	2	3	4	5	7

2SUM Problem

- N=6, x=9

1	2	3	4	5	7

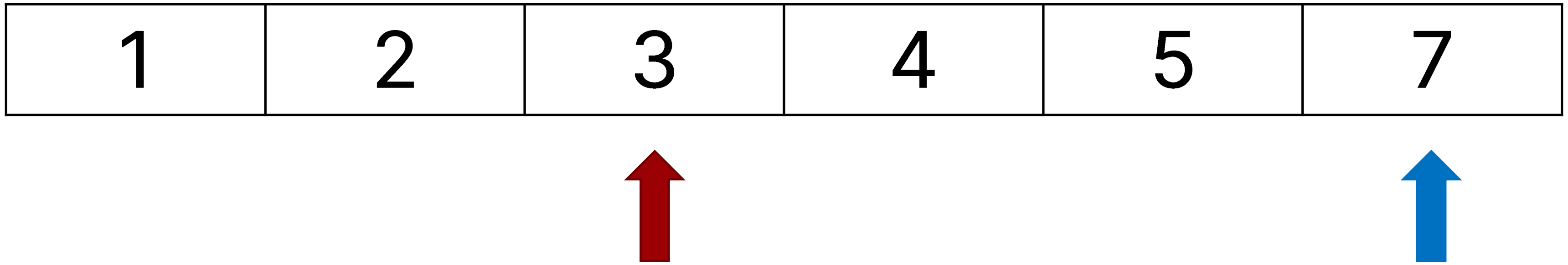
2SUM Problem

- N=6, x=9

1	2	3	4	5	7

2SUM Problem

- N=6, x=9



- N=6, x=9

1	2	3	4	5	7

- N=6, x=9

1	2	3	4	5	7

2SUM Problem

- N=6, x=9

1	2	3	4	5	7

2SUM Problem

- N=6, x=9

1	2	3	4	5	7

2SUM Problem

- N=6, x=9
- What if same element exists?

1	2	3	4	5	7

- 부 배열 : $a[i], a[i+1], \ldots, a[j-1], a[j] (1 \leq i \leq j \leq n)$

- 크기가 $n(1 \leq n \leq 1{,}000)$인 두 정수 배열 A와 B ($|A[i]|, |B[i]| < 1{,}000{,}000$)
- 부 배열의 합이 $T(-10^9 \leq T \leq 10^9)$가 되는 모든 부 배열 쌍의 개수?

- **문제 해결 과정**

1. 나올 수 있는 구간의 경우의 수 = $\binom{n}{2} \approx O(n^2) \leq 10^6$

- **문제 해결 과정**

1. 나올 수 있는 구간의 경우의 수 = $\binom{n}{2} \approx O(n^2) \leq 10^6$
2. 각 구간 합은 prefix sum으로 전처리 가능

- **문제 해결 과정**

1. 나올 수 있는 구간의 경우의 수 = $\binom{n}{2} \approx O(n^2) \le 10^6$
2. 각 구간 합은 prefix sum으로 전처리 가능
3. 아까 배운 것 적용하기

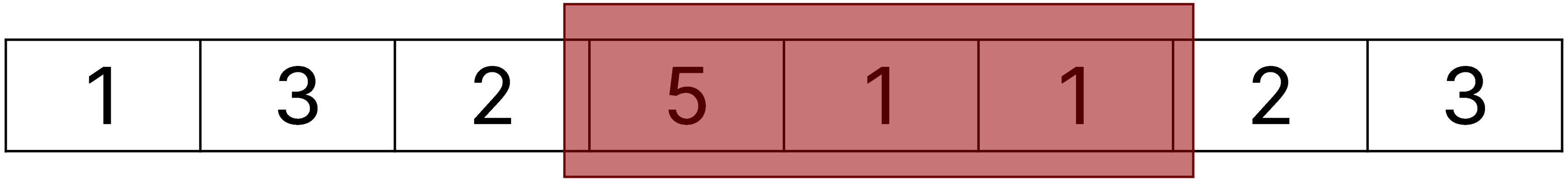
- **문제 해결 과정**

1. 나올 수 있는 구간의 경우의 수 = $\binom{n}{2} \approx O(n^2) \leq 10^6$
2. 각 구간 합은 prefix sum으로 전처리 가능
3. 아까 배운 것 적용하기
4. 구간 합이 같은 경우가 여러 개 나올 수 있는 경우

    Pointer L, R 각각에 대하여 크기가 같은 것의 개수 세기

Sliding Window

- 어떤 배열에 대해서 한 방향으로 움직이는 **고정된** 크기의 부분 배열
- 문제 해결 과정에서 전체 메모리가 아닌 **필요한 부분만**을 **저장**하는 기법



- $N(3 \le N \le 300{,}000)$명의 학생의 이름이 성적순으로 주어지고 자신과 등수 차이가 K $(1 \le K \le N)$이하인 사람 중 이름의 길이가 같은 경우 좋은 친구라 하자.
- $2 \le$이름의 길이≤ 20

- K=3

7	5	6	5	7	6

- K=3

7	5	6	5	7	6

- K=3

7	5	6	5	7	6

- **문제 해결 과정**

1. 이름을 받아 자연수 형태로 저장

- **문제 해결 과정**

1. 이름을 받아 자연수 형태로 저장
2. Window가 이동하면서 앞에는 들어가고 뒤에는 빠지는 자료구조?

- **문제 해결 과정**

1. 이름을 받아 자연수 형태로 저장
2. Window가 이동하면서 앞에는 들어가고 뒤에는 빠지는 자료구조?
   ⇒deque or queue

- **문제 해결 과정**

1. 이름을 받아 자연수 형태로 저장
2. Window가 이동하면서 앞에는 들어가고 뒤에는 빠지는 자료구조?
   ⇒deque or queue
3. Window의 가장 우측 원소 값과 같은 길이를 window 내에서 count

- **문제 해결 과정**

1. 이름을 받아 자연수 형태로 저장
2. Window가 이동하면서 앞에는 들어가고 뒤에는 빠지는 자료구조?
   ⇒deque or queue
3. Window의 가장 우측 원소 값과 같은 길이를 window 내에서 count
4. 당연히 매번 count하면 TLE, Window가 이동할 때마다
   우측 원소 값은↑ , 좌측 원소 값은↓

#3078 좋은 친구

```cpp
7   int N, K, len[300030], lcnt[22];
8   int main() {
9       ios_base::sync_with_stdio(false); cin.tie(0);
10      cin >> N >> K;
11      for (int i = 0; i < N; i++) {
12          string str;
13          cin >> str;
14          len[i] = (int)str.length();
15      }
16      deque<int> dq;
17      long long ans = 0;
18      for (int i = 0; i < N; i++) {
19          while (!dq.empty() && i - dq.front() > K)
20              lcnt[len[dq.front()]]--, dq.pop_front();
21          ans += lcnt[len[i]];
22          dq.push_back(i), lcnt[len[i]]++;
23      }
24      cout << ans;
25  }
```

19 : in case of diff(grade) is bigger than K

22 : add len[i] after query done

#Problem set

#2003 수들의 합 2
#1806 부분합
#1644 소수의 연속합
#7453 합이 0인 네 정수
#2143 두 배열의 합
#4373 수집합
#10256 돌연변이
#14572 스터디 그룹

#2230 수 고르기
#3078 좋은 친구
#2492 보석
#11003 최솟값 찾기