Interval Comparison

> 출제자 : 이승준
> 정해 : 투포인터, 이분탐색

풀이

함수 f의 정의로부터 $f(s,e) \leq f(s,e+1),\ f(s+1,e+1) \leq f(s,e+1)$임을 관찰할 수 있습니다. 따라서 문제에서 주어진 구간의 대소비교의 정의에 의해 $(s,e) \leq (s,e+1),\ (s+1,e+1) \leq (s,e+1)$임을 알 수 있습니다.

그러므로 모든 구간의 집합에 대해 투포인터를 사용해 $f(s,e) < T$인 동안 e를 증가시키고 $f(s,e) \geq T$인 동안 s를 증가시키면서 $f(s,e) \geq T$인 (s,e)들을 전부 찾으면 그 중 가장 작은 것이 정답이 됩니다.

이 때, 어떤 구간 (s,e)에 대해 $\sum_{i=s}^{e} a[i]/(T-1) \leq M$ 이면 $f(s,e)$가 $T-1$이하이고, $\sum_{i=s}^{e} a[i]/(T-1) > M$ 이면 $f(s,e)$가 T 이상이라는 점을 이용하면 $a[i]/(T-1)$의 누적합 배열을 사용해 $f(s,e)$와 T를 $O(1)$에 비교할 수 있습니다.

이제 $f(s,e) \geq T$ 인 구간들을 모두 찾았으니, 이 중 $f(s,e)$값이 가장 작은 구간을 찾아 봅시다. ($f(s,e) \geq T$ 인 구간이 존재하지 않을 경우 -1을 출력하면 됩니다)

$f(s,e)$값에 대해 이분탐색을 사용해 $f(s,e) \geq T$인 구간들 중 $f(s,e)$값이 가장 작은 구간의 $f(s,e)$값을 구합니다. 마찬가지로 이분탐색의 mid값으로 $a[i]$를 모두 나눈 수열의 누적합을 저장하면 각 구간에 대해 $\sum_{i=s}^{e} a[i]/mid$ 값을 $O(1)$에 구할 수 있습니다. $\sum_{i=s}^{e} a[i]/mid$ 값이 M보다 큰 구간이 하나도 없으면 left=mid+1, 하나라도 있으면 right=mid로 이분탐색을 구현하면 됩니다.

$\min_{f(s,e) \geq T} f(s,e)$를 구했으므로 $f(s,e)$값이 이 최솟값과 같은 모든 구간들에 대해 구간의 크기와 시작 위치를 비교하면 정답을 구할 수 있습니다. 이분탐색으로 최솟값을 찾기 전에 구간들을 미리 구간의 크기와 시작 위치에 대해 정렬해도 됩니다.

투포인터로 $f(s,e) \geq T$ 인 구간들을 찾는 데 $O(n)$, $\min_{f(s,e) \geq T} f(s,e)$를 찾는 데 $O(n \cdot log(10^9))$이 걸리므로 최종 시간복잡도는 $O(n \cdot log(10^9))$입니다.

```
#include <bits/stdc++.h>
#define mp make_pair
using namespace std;
typedef long long ll;
typedef pair<ll, ll> pii;
typedef pair<ll, pii> piii;
ll n, m, t, arr[3000001], a[3000001], l = 1, r = 1, sum, mid, sen;
int psum[3000001];
pii ans;
vector<piii> ranges;
pii itvcmp(pii a, pii b){
  if(a.second-a.first < b.second-b.first) return a;
  if(a.second-a.first > b.second-b.first) return b;
  if(a.first < b.first) return a;
```

```
  return b;
}
int main(){
  scanf("%lld %lld %lld", &n, &m, &t);
  for(int i = 1; i <= n; i++){ scanf("%lld", &a[i]); arr[i] = (t > 1 ? a[i]/(t-
1) : m+1);}
  while(1){
    while(sum <= m && r <= n) sum += arr[r++];
    if(sum <= m) break;
    while(sum > m) sum -= arr[l++];
    ranges.push_back(mp(r-1-(l-1), mp(l-1, r-1)));
  }
  if(ranges.size() == 0){printf("-1"); return 0;}

  sort(ranges.begin(), ranges.end());
  l = t; r = 1000000001;
  while(l < r){
    sen = 0; mid = (l+r)/2;
    for(int i = 1; i <= n; i++) psum[i] = psum[i-1] + a[i]/mid;
    for(int i = 0; i < ranges.size(); i++){
      ll s = ranges[i].second.first, e = ranges[i].second.second;
      if(psum[e]-psum[s-1] <= m){
        sen = 1;
        ans = mp(s, e);
        break;
      }
    }
    if(sen == 0) l = mid+1;
    else r = mid;
  }
  printf("%lld %lld", ans.first, ans.second);
}
```