신성한 요리

> 출제자 : 이승준
> 정해 : 구현, 세그먼트 트리

풀이

기본적인 구현은 매일 들어오는 재료들의 종류를 저장한 뒤 모든 재료들이 각각 하나 이상 모였을 경우 모든 재료들을 하나씩 제거하는 것입니다. 이 때, 시간이 지나면 재료가 상하기 때문에 매일 어떤 재료가 상하는지를 확인해야 합니다.

우선 매일 들어오는 각 재료들과 재료가 상하는 시간을 저장하는 큐(storage) 하나와 각 재료들에 대해 현재 모아 놓은 재료들의 상하는 시간을 저장하는 큐들을(ings[]) 재료의 갯수만큼 준비합니다.

또한, 현재 재료가 모두 모였는지를 확인하기 위해 전체 재료의 갯수 크기의 배열에 대한 세그먼트 트리를 준비합니다. 이 배열은 현재 i번째 재료가 하나라도 있으면 그 자리에 1, 하나도 없으면 0을 가집니다. 따라서 이 배열에 대해 0부터 n-1까지의 구간합이 n일 경우 재료가 모두 모여 있는 것입니다.

이제 하루 단위로 k개의 재료들을 입력 받으며 매일 들어오는 k개의 재료들을 전부 현재 시간+t와 함께 storage에 push하고, 각 재료들에 대한 ings[재료]에 현재 시간+t를 push 합니다. 이 때, ings[재료]가 push전 비어 있었다면 세그먼트 트리의 해당 재료를 1로 업데이트 합니다.

k개의 재료들을 전부 저장한 뒤, 세그먼트 트리의 [0, n-1]구간의 합이 n일 경우 요리를 만들 수 있습니다. 그러면 모든 ings[]에서 한 번씩 pop을 진행합니다. (재료는 항상 현재 가진 것 중에서 가장 오래 된 것을 사용하는게 이득이기 때문입니다) 구간합이 n보다 작을 경우 저녁을 굶는 날을 1 증가시킵니다.

매일 첫 번째 재료가 입력되기 전 storage에서 오늘 상하기로 되어 있는 재료들을 전부 pop하고 이 재료들(tmp)에 대해 ings[tmp]를 확인합니다. 만약 ings[tmp]의 top이 오늘 상하기로 되어 있는 재료라면, 이 재료는 요리에 사용되지 않은 것이므로 그대로 pop합니다. 이 때, 해당 재료의 갯수가 0이 되면 세그먼트 트리의 해당 재료를 0으로 업데이트 합니다.

시간복잡도를 분석해 보면, 매일 들어온 모든 재료들과 오늘 상하는 모든 재료들에 대한 처리를 하는 데 $O(k \cdot log(n))$, 모든 재료가 다 모였을 경우 ings[]에서 한 번씩 pop하고 재료들이 0개가 된 경우 $O(n \cdot log(n))$이 걸립니다. 이 때, 모든 재료가 다 모이기 위해서는 적어도 n/k일이 필요하므로, 실제로 요리를 만들 수 있는 날의 수는 최대 $m/(n/k) = (m \cdot k)/n$이 됩니다. 따라서 모든 m번의 날에 대해 요리를 만드는 일이 일어남으로 인해 걸리는 시간의 합은 $O(n \cdot log(n) \cdot (m \cdot k)/n) = O(m \cdot k \cdot log(n))$입니다.

결국 전체 시간복잡도는 $O(m \cdot k \cdot log(n))$가 됩니다. 이 풀이는 굳이 세그먼트 트리를 사용하지 않고도 구현할 수 있는데, 이 경우 전체 시간복잡도는 $O(m \cdot k)$가 됩니다.

```
#include <bits/stdc++.h>
#define mp make_pair
using namespace std;
typedef pair<int, int> pii;
int n, m, k, t, seg[4000001], ing, segp, ans;
queue<int> ings[1000001];
queue<pii> storage;
```

```
void update(int i, int diff){
  i += (1<<segp); seg[i] += diff;
  for(i /= 2; i >= 1; i /= 2) seg[i] = seg[i*2]+seg[i*2+1];
}
int segq(int l, int r, int i, int ql, int qr){
  if(qr < l || r < ql) return 0;
  if(ql <= l && r <= qr) return seg[i];
  return segq(l, (l+r)/2, i*2, ql, qr) + segq((l+r)/2+1, r, i*2+1, ql, qr);
}
int main(){
  scanf("%d %d %d %d", &n, &m, &k, &t);
  while((1<<segp) < n) segp++;
  for(int l = 1; l <= m; l++){
    while(storage.size() > 0 && storage.front().second == l){
      int tmp = storage.front().first; storage.pop();
      if(ings[tmp].size() > 0 && ings[tmp].front() == l){
        ings[tmp].pop();
        if(ings[tmp].size() == 0) update(tmp, -1);
      }
    }
    for(int i = 0; i < k; i++){
      scanf("%d", &ing);
      ing--;
      if(ings[ing].size() == 0) update(ing, 1);
      ings[ing].push(l+t);
      storage.push(mp(ing, l+t));
    }
    if(segq(0, (1<<segp)-1, 1, 0, n-1) == n)
      for(int i = 0; i < n; i++){
        ings[i].pop();
        if(ings[i].size() == 0) update(i, -1);
      }
    else ans++;
  }
  printf("%d", ans);
}
```