CS202 Topic #5 Report

전산학부
20170038 권기훈

[purifier]

단순히 이 문제를 dp로 풀게 되면 n개의 공기청정기가 주어졌을 때, 업데이트는 O(1)이면 충분하나 그 최대 값을 찾는 과정은

```
dp[i] = max(dp[i-1], dp[i-2] + pur[i]);
```

위를 거쳐 1<i<=N에 대하여 반복해야 하므로 O(n)이 필요하다. 총 M번의 업데이트가 있으므로 총 시간복잡도는 O(N*M)이 되고, N, M 이 각각 2만 정도까지만 주어진 시간 내에 통과한다. 따라서 이 알고리즘으로 문제를 해결하면 23점 밖에 받지 못한다.

따라서 더 효율적으로 문제를 해결하기 위해서 문제를 분할하여 접근하였다. 반으로 쪼개어 왼쪽 부분과 오른쪽 부분에서 각각 최대가 될 수 있는 경우를 구하여 합하는 방식이다. 그러나 이 때의 문제는 왼쪽과 오른쪽의 최대를 그냥 합칠 경우에 아래와 같이 인접한 공기청정기를 고를 수 있게 된다.

(5, 6) & (9, 7) → (5, 6, 9, 7)

그래서 가장 왼쪽, 오른쪽 공기청정기를 골랐는지 여부에 따라서 모든 경우를 다 고려해줄 필요가 있다. 이를 해결하기 위해서 총 4개의 벡터를 사용했고, 이를 세그먼트 트리의 형태로 저장하였다. 변수명의 의미는 왼쪽과 오른쪽을 골랐는지 여부이다.

```
vector<long long> oo(530000), ox(530000), xo(530000), xx(530000);
```

초기화 혹은 변경 시에는 가급적 구간의 길이가 2인 노드까지 내려가서 ox와 xo에 각각 넣어주었고, 길이가 3인 구간을 쪼개느라 1인 구간이 나왔을 때는 xo에만 넣어주었다. 이제 말단에 대입을 하였으므로, 이제 재귀를 탈출하면서 업데이트를 해주는데, 왼쪽 부분과 오른쪽 부분이 각 트리의 특징을 만족하며, 합치는 부분에 공기청정기가 이웃하지 않도록 처리해준다. 단, 구간의 길이가 3일 때는 위에서 xo에만 대입했던 문제 때문에 이를 해결하기 위해 따로 처리를 해주었다. 이를 update 함수에 구현하였다.

그 후에는 새 공기청정기 값을 업데이트하면서 네 트리의 루트의 값 중의 최대값을 계속 더해주었다. 이제 업데이트를 하는 시간과 최대값을 구하는 시간은 O(logN)으로 줄었고, M번 업데이트를 하므로 시간복잡도는 O(MlogN) 이여서 문제 조건 하에 충분히 해결할 수 있다.

[errand]

이 문제는 I부터 J까지 모든 건물에 심부름을 한다고 가정한 후 가장 이득이 되는 경로를 찾아서 그 값을 빼주는 방식이다.

```
vector<pair<int, int> > buildings(110000);
vector<long long> stree(530000), mtree(530000);
```

이를 구현하기 위해 buildings에는 모든 건물의 위치를 기록하였고, stree에는 거리의 합을 세그먼트 트리의 형태로 저장한다. mtree에는 i+1번째 건물을 건너뛰고, i번째 건물과 i+2번째 건물을 바로 갔을 때 거리의 변화를 저장하고, 음수이므로 그 최솟값을 찾으면 가장 이득이 되는 경로가 된다.

실제 어떤 경로로 움직이는지는 알 필요가 없기 때문에, 단순히 그 값들만 가져와서 계산하였다. 업데이트나 구간 합, 구간 최소값을 구하는 함수는 segtest 문제에서 사용했던 것을 약간만 변형하여 사용하였다. 이 방법으로 해결할 경우, 각 연산마다 업데이트를 두번 하거나. 구간합과 구간 최소를 찾으므로 O(2logN)이고, 이를 총 M번 하므로, O(MlogN)으로 해결할 수 있다.