CS202 Topic #4 Report

전산학부
20170038 권기훈

[chaltteok]

떡을 하나씩 증가시켜가면서 규칙성을 찾아본 결과, k개의 떡의 pack 값과 k+1개의 떡의 pack 값의 차이는 +1, 0, -1 중 하나의 값을 갖는 것을 알 수 있습니다. 따라서 떡을 하나씩 추가하면서 이전 값과의 비교를 통하여 배열하였습니다. 따라서 떡을 배열하기 위한 이차원 벡터를 추가적으로 사용하였습니다.

① +1일 때는 기존에 있는 떡과 인접하지 않은 새로운 떡이 추가되는 상황입니다. 이 떡을 포함하는 벡터를 만들어 이차원 벡터에 넣습니다.

② 0일 때는 기존에 있는 떡들 중 하나의 무리(?)와만 인접하는 경우입니다. 정의한 함수(findTeok)를 통하여 이 떡과 겹치는 것을 떡을 찾고 그 벡터를 업데이트 해줍니다. findTeok 함수는 분할 정복의 원리로 이차원 벡터를 반씩 쪼개가며 합쳐지는 떡의 index를 반환합니다. 따라서 전체 벡터의 크기만큼 순회하는 것이 아니라, logN번으로 순회하는 횟수를 줄일 수 있습니다.

③ -1일 때는 기존에 있는 떡들 중 두 무리와 인접하는 경우입니다. 정의한 다른 함수(findTwoTeok)을 통하여 양 떡의 index를 pair로 반환합니다. 원리는 findTeok의 탐색과 유사하게 작동합니다. 이 pair를 이용하여 구한 떡 벡터들을 하나로 합쳐주고 이차원 벡터를 업데이트 합니다.

1부터 N까지 모든 수를 부르면 항상 1이기 때문에 최종적으로 이차원 벡터에는 정답 순서대로 배열된 벡터 하나만 남습니다. 이 값을 report 하였습니다.

[bridges]

위 아래 합쳐서 k개의 지점이 존재할 때, 모든 경우를 찾으려면 k-1번을 탐색해야 하는 규칙성을 발견할 수 있었습니다. 즉, 끝 지점이 제외되면 k-1개의 상황을 보면 따지는 것과 같아집니다. 끝 지점을 제외하는 방식은 윗줄에서는 뒤에서 두번째, 아랫줄에서는 제일 뒤의 지점을 sail 했을 때의 결과에 따라 달라집니다.

① -1일 때는 두 지점을 잇고 있는 다리가 있다는 의미이므로, 정답 벡터에 넣어주었고, 이제 윗줄의 제일 끝 점은 어느 점과도 연결될 수 없기 때문에 제거해주었습니다.

② 0일 때는, 두 지점을 연결하고 있는 다리가 없는 것이고, 또한 마찬가지로 윗줄의 제일 끝 점은 어느 점과도 연결될 수 없기 때문에 제거해주었습니다.

③ 이 값이 아닌 양수일 때는 윗줄의 제일 끝점에 도달하는 다리들이 있다는 의미이고, 아랫줄의 제일 끝점에는 도달하는 다리가 더 이상 없다는 뜻이기 때문에 아랫줄의 가장 마지막 점을 제거해 주었습니다. 다만 이제 아랫줄의 어떤 지점에서 다리가 시작되는 것인지는 모르기 때문에, 그 값을 last 변수에 저장해주었고, 다음 sail에서 이 값과 1차이가 발생할 경우, 그 다리를 통과했다는 의미가 되어서 그 때 정답 벡터에 넣어주었습니다.

최종적으로 위나 아래의 줄 중에서 하나의 지점만 남게 되면, 반대편의 모든 지점들과 연결 여부를 확인해야 하고, 이까지 마치게 된다면 답을 찾을 수 있습니다.