Day 1 problem solve

- for Lee

  1부터 1000까지의 숫자 중 어떤 숫자 a를 골라도 $n \equiv a - 1 (\mathrm{mod}\, a)$인 n를 구하는 문제였습니다. n이 음수도 될 수 있다는 생각을 해 보면 n이 -1이면 a가 꼭 1부터 1000까지가 아니라 어떤 정수가 되어도 위 식이 성립함을 알 수 있습니다.

- symmetric difference set

  대칭차집합에 속하는 원소의 개수를 묻는 문제였습니다. 각각의 집합에서 원소가 다른 집합에 있는지 naive하게 찾으면 $O(nm)$에 풀리며 set 혹은 vector로 값을 관리하며 stl에 있는 함수인 symmetric_difference_set 등을 사용하면 $O(n\log n + m\log m)$정도의 시간복잡도에 통과할 수 있습니다. 이때 set은 매우 느리므로 다른 방법을 생각해봐야 하는데 전체 원소개수에서 중복된 원소 개수를 2번 빼면 된다는 생각을 가지고 stl의 unique라는 함수를 이용하며 중복을 제외한 원소의 개수를 구해 2*(중복이 없는 경우의 전체 원소 개수) – (전체 원소의 개수)를 하면 대칭차집합에 속하는 원소 개수를 구할 수 있습니다. 이때 한 배열에 n+m개의 원소를 모두 담으면 여유롭게 110점을 받을 수 있으며 vector를 n+m의 크기로 한 번에 늘린 경우 역시 아슬아슬하게 통과가 가능하지만 vector에 한 원소씩 n+m번 원소를 push_back할 경우 시간복잡도에 상수배가 되어 110점을 받기 힘들 수 있습니다.

- parenthesis string

  이 문제는 stack을 이용하여 푸는 대표적인 문제입니다. '('인 경우에는 stack에 push, ')'인 경우에는 stack에서 pop을 하여 stack이 비어있을 때 pop을 하거나 모든 string을 봤을 때 stack이 비어 있지 않으면 NO, 그렇지 않은 경우 YES를 출력하면 됩니다.

- deck

  이 문제는 deque를 쓰라고 준 문제입니다. 문자열의 맨 앞부터 보면서 deque가 비어있으면 deque에 넣어주고 deque의 맨 앞의 문자보다 사전순으로 뒤인 문자면 deque의 뒤에, 그렇지 않다면 deque의 앞에 문자를 넣은 뒤에 가장 마지막에 deque의 맨 앞부터 출력하며 pop을 하면 됩니다.

- top

  이 문제도 stack을 쓰는 대표적인 문제 중 하나입니다. 풀이 방식은 다음과 같습니다.

  오른쪽에 위치한 탑부터 왼쪽에 위치한 탑까지 순회합니다.

  각 탑을 볼 때 다음의 작업을 수행합니다.

  1) stack이 비어있지 않고 stack의 top에 있는 탑이 현재 보고 있는 탑보다 높이가 낮다면 stack의 top에 위치한 탑의 신호는 지금 보고 있는 탑에 의해 막히게 됩니다. 따라서 그 표시를 해주고 stack에서 pop을 해 줍니다.

  2) stack이 비었거나 stack의 top에 있는 탑이 현재 탑의 높이와 같거나 더 높다면 현재 탑을 stack에 넣어줍니다.

  모든 순회가 끝났을 때 stack에 남아있는 탑들은 신호가 막히지 않는 탑들입니다.

  이 과정이 왜 되는지는 간단한 예제를 만들어 손으로 직접 따라가다 보면 이해가 되실 겁니다. 만약 잘 되지 않는다면 comment 주시면 감사하겠습니다.

- range

  이 문제는 deque를 사용하여 푸는 문제였습니다. 최솟값과 최댓값을 구하는 문제이지만 일반화하여 최솟값만 구하는 경우로 생각하겠습니다.

  우선 가장 앞의 k개는 정방향으로 모두 deque에 넣습니다. 이때 주의할 점은 deque의 back에 push하는데 만약에 deque의 back에 있는 값이 현재 넣으려는 값보다 크다면 그 값들은 모두 pop하고 현재 값을 push합니다. 예를 들겠습니다.

  dq : 2 5 7, 현재 넣으려는 값 : 4

  dq의 back의 값(7)이 4보다 큼 → 7 pop

  dq의 back의 값(5)이 4보다 큼 → 5 pop

  dq의 back의 값(2)이 4보다 작거나 같음 → 4 push

  dq : 2 4

  이 과정을 1~k번째 값들에 대해 했다면 이 구간의 최솟값은 dq의 front에 해당하는 값입니다.

  이제 구간을 옮겨야 합니다. i~i+k-1의 구간에서 i+1~i+k의 구간으로 넘어갈 때 해야 할 것은 dq에서 i번째 값을 빼고 i+k번째 값을 넣는 것 입니다. dq에서 i번째 값을 뺄 때는 다음과 같은 작업을 합니다.

1) 만약 dq의 front 값이 i번째 값과 같다면 아직 빠지지 않은 것. 따라서 front pop

2) 만약 dq의 front 값이 i번째 값과 같지 않다면 아직 값이 빠지지 않은 것. 따라서 pop할 필요 없음

i+k번째 값을 넣는 것은 위에서 back에 push할 때처럼 하면 됩니다. 이 과정을 한 뒤 dq의 front 값이 최솟값입니다.

이 과정을 구간을 옮길 때 마다 해 주면 각 구간의 최솟값을 구할 수 있습니다. 각 구간의 최대값은 이 과정에서 대소관계를 뒤집어 주어 다른 dq를 동시에 관리하여 풀 수 있습니다.

위의 문제들의 구현 소스는 (문제이름).cpp파일로 이 파일과 같은 경로에 존재합니다.