1. A[j]-A[i]+A[l]-A[k]

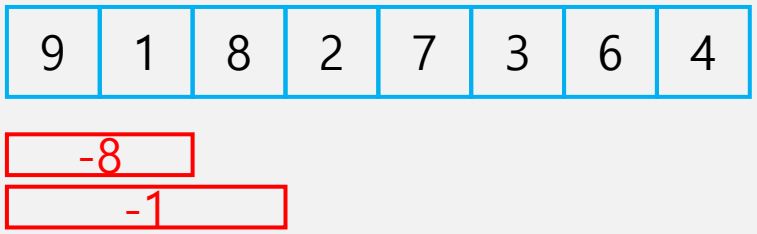
문제 파악

1. 배열 A에서 (A[j] + A[i] + A[l] – A[k])의 최댓값 구하기 (i<j<k<l)

1. A[j]-A[i]+A[l]-A[k]

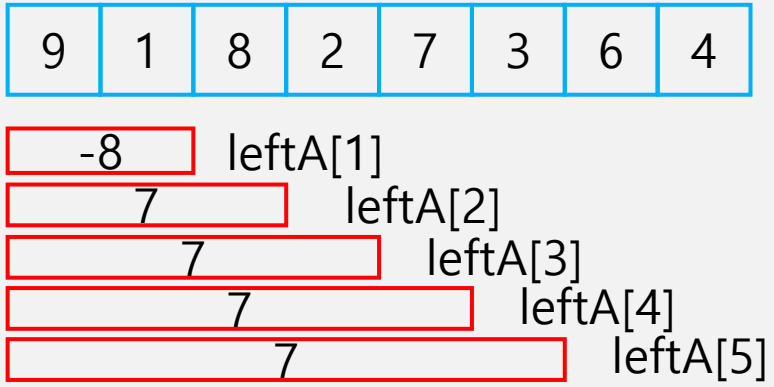
문제 풀이

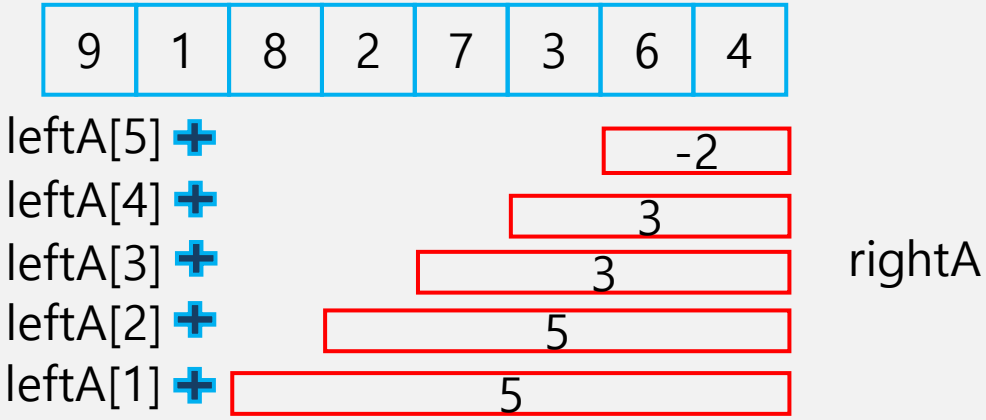
1. 1에서부터 j를 늘려가며 각 구간에서 (A[j] – A[i])의 최댓값을 계산 (leftA[j]는 index 0에서 j까지 (A[j] – A[i])의 최댓값)
2. n – 2부터 k를 줄여가며 각 구간에서 A[l] – A[k]의 최댓값을 계산 (rightA는 index k에서 (n – 1)까지 (A[l] – A[k])의 최댓값), 그 후 (1)에서 구한 leftA[j](j < k)와 합산하며 최대 합 갱신



1. A[j]-A[i]+A[l]-A[k]

문제 풀이





1. A[j]-A[i]+A[l]-A[k]

핵심 코드

```
// [0, j] 범위에서 A[j] - A[i]의 최댓값
int minA = A[0];
leftA[0] = -1e9;
for (int j = 1; j < n - 2; j++) {  // A[j] 뒤에 최소 숫자 2개 더 존재
    leftA[j] = max(leftA[j - 1], A[j] - minA);  // minA는 [0, j) 범위에서 최솟값
    minA = min(minA, A[j]);
}

// [k, n) 범위에서 A[l] - A[k]의 최댓값 구하고 최종 결과 합산
int maxA = A[n - 1], rightA = -1e9;
for (int k = n - 2; k > 1; k--) {  // A[k] 앞에 최소 숫자 2개 더 존재
    rightA = max(rightA, maxA - A[k]);  // 마찬가지로 maxA는 (k, n) 범위에서 최솟값
    max_sum = max(max_sum, leftA[k - 1] + rightA);  // (A[j] - A[i])의 최댓값 + (A[l] - A[k])의 최댓값
    maxA = max(maxA, A[k]);
}
```

2. 행렬

문제 파악

1. 0과 1로만 이루어진 두 행렬 A, B
2. A를 B와 같게 만드는 최소한의 행렬 연산 횟수, 불가능하면 -1 출력
3. 행렬 연산: 3*3 범위를 동시에 바꾸는 것 (0->1, 1->0)

2. 행렬

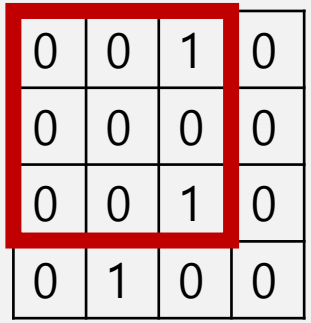
문제 풀이

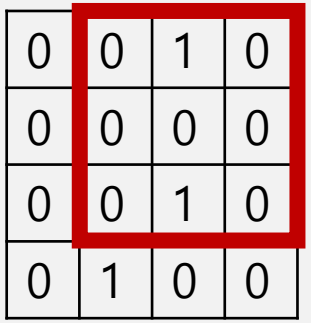
1. 예외 처리: 행이나 열 둘 중 하나가 3보다 작고 A와 B가 다르면 행렬 연산을 해야 하지만 행렬 연산 불가하므로 무조건 -1 출력
2. 왼쪽 위에서부터 행렬 연산을 실행할 경우 위쪽 행은 더 이상 변화 불가

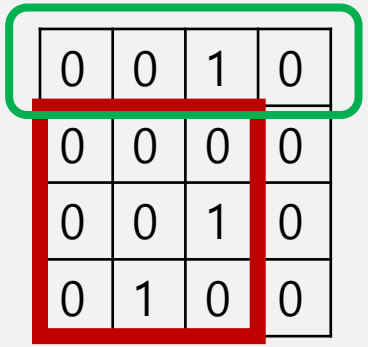
2. 행렬

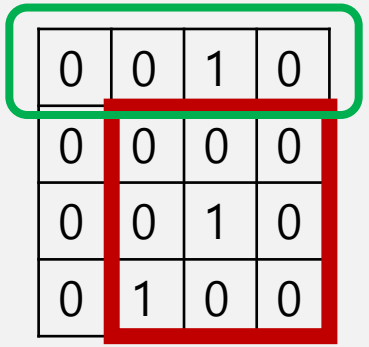
문제 풀이

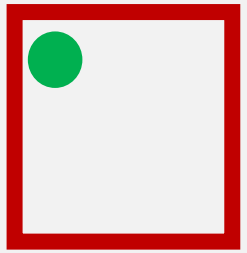
한 번 거쳐간 행은 다시 계산 불가:
행이 넘어갈 때마다 A와 B의 행 비교











왼쪽 위의 원을 기준으로 반복문을 통해서 행렬 연산

이 부분은 한번 행렬 연산하면 다시는 변경할 수 없으므로 무조건
이 위치에서 A와 B의 값은 같아야 함

2. 행렬

핵심 코드

```
// 행렬 변환 연산은 A[i][j]가 왼쪽 위가 되로록 3*3 범위를 변화시킴
for (int i = 0; i < n - 2; i++) {
    for (int j = 0; j < m - 2; j++) {
        if ((A[i][j] + d[i][j]) % 2 != B[i][j]) {
            for (int k = 0; k < 3; k++)
                for (int l = 0; l < 3; l++)
                    d[i + k][j + l]++;
            sum++;
        }
    }
    // 더 이상 i번째 행에서 행렬 변환 연산 불가
    // 이때 i번째 행 전체는 A와 B가 동일해야 함, 동일하지 않으면 -1 출력 후 종료
    for (int j = 0; j < m; j++) {
        if ((A[i][j] + d[i][j]) % 2 != B[i][j]) {
            cout << -1;
            return 0;
        }
    }
}
```