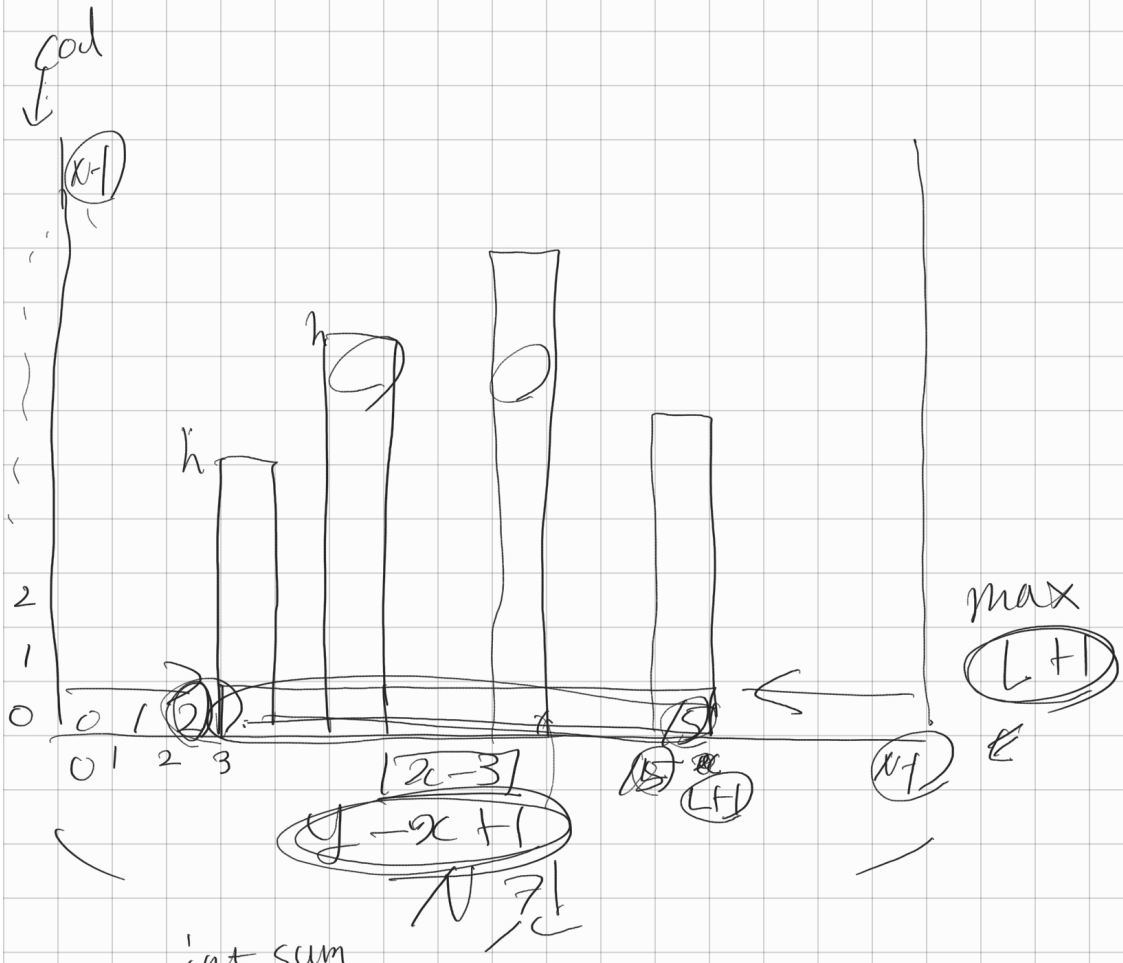
Baekjoon 2304

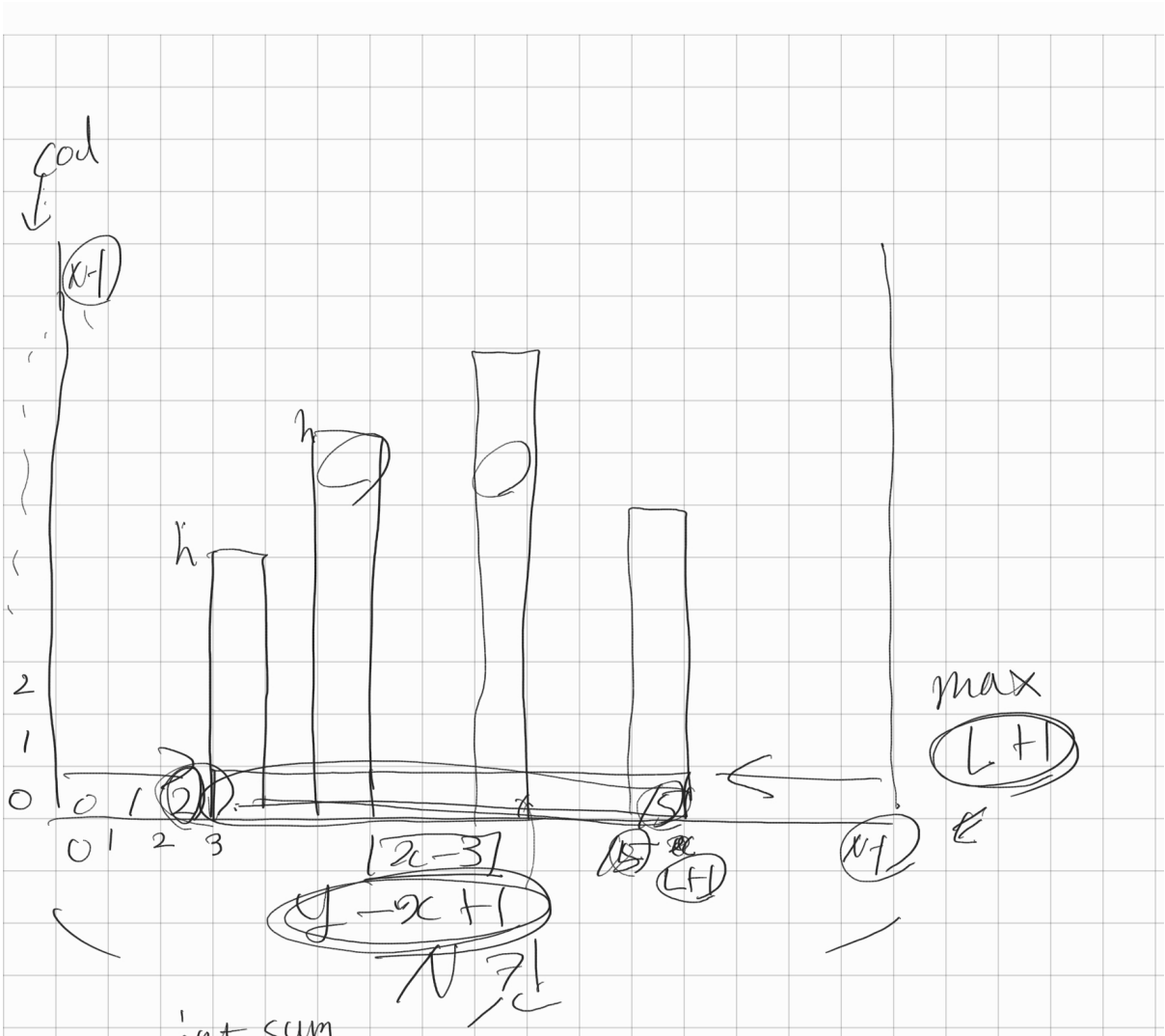
날짜	@2021년 7월 28일 → 2021년 8월 1일
링크	https://www.acmicpc.net/problem/2304
열	
태그	Baekjoon Silver2 브루트 포스 알고리즘 스택 자료 구조




```
int sum
for ( int col = 0 ; col < maxH ; col++)
    int x = 0
    while( ( x <= L ) && ( map[cnt][x] == false )
        x++

    int y = L+1
    while ( ( y >= 0 ) && ( map[cnt][y] == false )
        y -= 2

    sum += y-x
```




```
int sum
for ( int col = 0 ; col < maxH ; col++)
    int x = 0
    while( ( x<=L ) && ( map[cnt][c] == false )
        x++

    int y = L+1
    while ( ( y >= 0 ) && ( map[cnt][y] == false )
        y -= 2

    sum += y-x
```

실패

```
package silver2;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.StringTokenizer;

public class Test2304_01 {
  public static void main(String[] args) throws NumberFormatException, IOException {
    BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
    StringTokenizer st;
    int num = Integer.valueOf(br.readLine());
    int maxL = 0;
    int maxH = 0;
    boolean[][] map = new boolean[1005][1005];
    for (int i = 0; i < num; i++) {
      st = new StringTokenizer(br.readLine());
      int l = Integer.valueOf(st.nextToken());
      int h = Integer.valueOf(st.nextToken());
      if (l > maxL)
        maxL = l;
      if (h > maxH)
        maxH = h;
      for (int j = 0; j < h; j++) {
        map[l][j] = true;
      }
    }

    int sum = 0;
    for (int row = 0; row <= maxL + 2; row++) {
      int x = 0;
      while ((x <= maxL + 1) && (map[x][row] != true))
        x++;
      if (x >= maxL + 1)
        break;
      int y = maxL + 2;
      while ((y >= 0) && (map[y][row] != true))
        y--;
      System.out.println("y : "+y+" x : " +x + "\ty - x "+ (y - x+1));
      sum += (++y - x );

    }
    System.out.println(sum);
  }
}
```

내 계획

1. 입력 받은 정보들을 2차원 배열(논리형)에 저장한다. {입력 for}
    a. 이때 건물을 최대 높이(maxH), 마지막 위치(maxL)를 저장한다.
2. 0층부터 가장 높은 층까지 반복한다. {계산 for row}

a. 행렬의 시작(x)부터 한 칸씩 증가 하면서 첫번째 건물을 만날 때 까지 칸을 센다.

b. 행렬의 최대 위치 + 2 부터 한 칸씩 감소 하면서 가장 마지막의 건물을 만날 때 까지 칸을 센다.

c. 각 층 {for row}에서의 y - x +1 값을 sum에 저장한다.

   ex) x = 1, y = 3 이면 1,2,3에 건물이 있고 면적이 3이다. ⇒ y - x + 1

▼ 채점 81%에서 틀림....

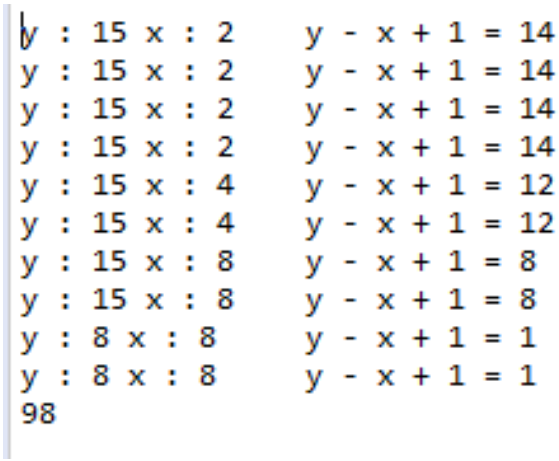
기본 입력 뒤집은 결과는 다음과 같다.

```
7
13 4
8 4
5 8
4 6
15 3
11 10
2 6
```

```
y : 15 x : 2    y - x + 1 = 14
y : 15 x : 2    y - x + 1 = 14
y : 15 x : 2    y - x + 1 = 14
y : 13 x : 2    y - x + 1 = 12
y : 11 x : 2    y - x + 1 = 10
y : 11 x : 2    y - x + 1 = 10
y : 11 x : 5    y - x + 1 = 7
y : 11 x : 5    y - x + 1 = 7
y : 11 x : 11   y - x + 1 = 1
y : 11 x : 11   y - x + 1 = 1
90
```

기본 입력 결과는 다음과 같다.

```
7
2 4
11 4
15 8
4 6
5 3
8 10
13 6
```



단순히 뒤집은 거라 같은 결과 나와야 하는거 아닌가...?

마무리는 @2021년 7월 29일 의 나에게.... 행렬의 인덱스부터 정확하게 식으로 표현해보자....

그리고 문제 유형에 스택이 있는데 스택을 사용할 곳이 있나?

브루스 포스 알고리즘은 전체 탐색이란 의미인데 내가 한 방법인듯?

요즘 신용카드 돌려 막기처럼 내일의 나로 돌려 막기 하고 있는데 문제 난이도 선택을 잘 해야 할 듯....

```
package silver2;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.StringTokenizer;

//public class Main {
public class Test2304_02 {
  public static void main(String[] args) throws NumberFormatException, IOException {
    BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
    StringTokenizer st;
    int N = Integer.valueOf(br.readLine());
    int maxL = 0;
    int maxH = 0;
    boolean[][] map = new boolean[1001][1001];
    for (int n = 1; n <= N; n++) {
      st = new StringTokenizer(br.readLine());
      int Ln = Integer.valueOf(st.nextToken());
      int Hn = Integer.valueOf(st.nextToken());
      if (Ln > maxL)
        maxL = Ln;
      if (Hn > maxH)
        maxH = Hn;
      for (int h = 1; h <= Hn; h++) {
        map[Ln][h] = true;
      }
```

```
    }
    int sum = 0;
    for (int row = 1; row < 1001; row++) {
      int x = 0;
      while ((x <= 1000) && (map[x][row] != true))
        x++;
      if (x > 1000)
        break;
      int y = maxL + 2;
      while ((y >= 0) && (map[y][row] != true))
        y--;
      System.out.println("y : " + y + " x : " + x + "\ty - x + 1 = " + (y - x + 1));
      sum += (y - x + 1);
    }
    System.out.println(sum);

  }
}
```

변수설정

map[col][row]

⇒ $M_{col,row}$

int sum = 0 (면적 저장)

$M = 1001 \times 1001$

l: 1~1000, h: 1~1000

N = 기둥개수

L_n = n번째 기둥위치, L_{max} = 가장 끝 위치

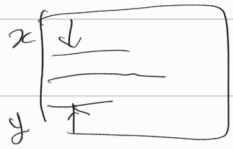
H_n = n번째 기둥높이, H_{max} = 가장높은 높이

입력

```
for(int n=1 ; n≤N ; n++)
    Ln 입력, Hn입력
    Lmax, Hmax 파악
    for(int h=1 ; h≤Hn ; h++)
        M[Ln,h] = true;
```

⇒ L→ [~~~~] Ln위치에 Hn 만큼의 칸이 true 셋팅

```
    // end h
// end n
```



면적 구하기 n번째 기둥 높이 1≤ ≤hn
이 없다.

```
for (int row = 1 ; r < 1001 ; r++)          멈춤 조건
{
  int x = 1;
  while(!((x > 1000) || (M[x][row] == true)))
    x++;                                     건물의 시작위치
  int y = 1000; 마지막 위치
  while(!(y < 0) || (M[y][row] == true))
    y++;                                     건물의 마지막위치
  if((x > 1000))
    break;
  sum += y - x + 1                           건물의 마지막 위치 포함
}
sys out (sum)
```

▼ 여전히 같은 결과

```
7
13 4
8 4
5 8
4 6
15 3
11 10
2 6
```

```
y : 15 x : 2    y - x + 1 = 14
y : 15 x : 2    y - x + 1 = 14
y : 15 x : 2    y - x + 1 = 14
y : 13 x : 2    y - x + 1 = 12
y : 11 x : 2    y - x + 1 = 10
y : 11 x : 2    y - x + 1 = 10
y : 11 x : 5    y - x + 1 = 7
y : 11 x : 5    y - x + 1 = 7
y : 11 x : 11   y - x + 1 = 1
y : 11 x : 11   y - x + 1 = 1
90
```

기본 입력 결과는 다음과 같다.

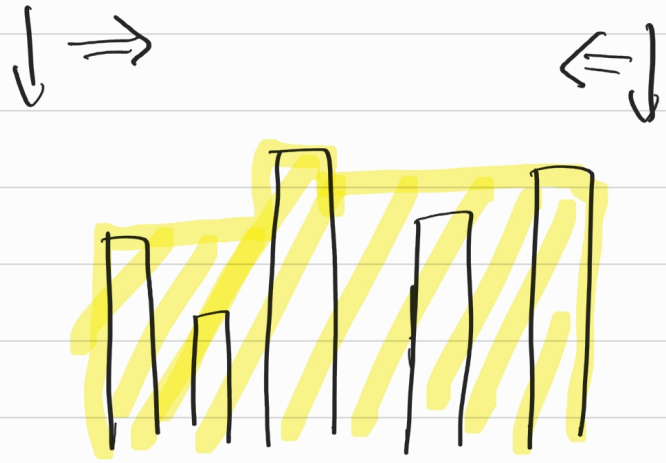
```
7
2 4
11 4
15 8
4 6
5 3
8 10
13 6
```

```
y : 15 x : 2    y - x + 1 = 14
y : 15 x : 2    y - x + 1 = 14
y : 15 x : 2    y - x + 1 = 14
y : 15 x : 2    y - x + 1 = 14
y : 15 x : 4    y - x + 1 = 12
y : 15 x : 4    y - x + 1 = 12
y : 15 x : 8    y - x + 1 = 8
y : 15 x : 8    y - x + 1 = 8
y : 8 x : 8     y - x + 1 = 1
y : 8 x : 8     y - x + 1 = 1
98
```

@2021년 8월 1일 뭐지 문제 푸는 개념은 똑같은데... 풀이 방법만 조금 바꿨는데 왜 성공한 거지???

성공

기존 행렬 방법도 틀리는데
왜 틀리는지 모르겠다.....
다른 방식으로 접근해보자



① 왼쪽에서 가장 높은 기둥까지 탐색
② 오른쪽에서 〃

둘다 가장 높은 기둥 높이만큼 더해주면서
이동한다.

```
package silver2;
```

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.StringTokenizer;

//public class Main { // 접근을 바꿔보자
public class Test2304_3 {
  public static void main(String[] args) throws NumberFormatException, IOException {
    BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
    StringTokenizer st;
    int N = Integer.valueOf(br.readLine());
    int maxL = 0;
    int maxH = 0;
    int maxHIndexL = 0;
    int[] pillar = new int [1001];

    for (int n = 1; n <= N; n++) {
      st = new StringTokenizer(br.readLine());
      int Ln = Integer.valueOf(st.nextToken());
      int Hn = Integer.valueOf(st.nextToken());

      if (Ln > maxL)
        maxL = Ln;
      if (Hn > maxH) {
        maxH = Hn;
        maxHIndexL = Ln;
      }
      pillar[Ln] = Hn;

    }
    int sum = 0;
    int leftIdx = 0;
    int leftMaxPillar = 0;
    while(leftIdx != maxHIndexL) {
      if(leftMaxPillar < pillar[leftIdx])
        leftMaxPillar = pillar[leftIdx];
      sum += leftMaxPillar;
      leftIdx++;
    }

    int rightIdx = 1000;
    int rightMaxPillar = 0;
    while(rightIdx != maxHIndexL) {
      if(rightMaxPillar < pillar[rightIdx])
        rightMaxPillar = pillar[rightIdx];
      sum += rightMaxPillar;
      rightIdx--;
    }
    sum += maxH;
    System.out.println(sum);
  }
}
```

뒤집은 결과는 다른게 맞나보다....

코드 통과했는데 애도 똑같이 98,90 나온다