```
Given n non-negative integers representing an elevation map where the
width of each bar is 1, compute how much water it can trap after raining.
```

分析：

1. 位置i可以放置多少水和左侧最大值l_max和右侧最大值r_max有关。
2. 对位置i的储水量为std::min(l_max, r_max) - height[i].

有暴力搜索解法：

```
class Solution {
public:
  int trap(std::vector<int>& height) {
    int len = height.size();
    int ans = 0;
    for (int i = 0; i < len; i++) {
      int l_max = 0, r_max = 0;
      for (int j = i; j >= 0; j--) {
        l_max = (l_max > height[j] ? l_max : height[j]);
      }

      for (int j = i; j < len; j++) {
        r_max = (r_max > height[j] ? r_max : height[j]);
      }

      ans += std::min(l_max, r_max) - height[i];
    }

    return ans;
  }
};
```

大数据量超时。

备忘录优化

1. 备忘录优化方法可以先计算出i位置出的leftmax[i], rightmax[i],后续计算直接使用该值。

```
class Solution {
public:
  int trap(std::vector<int>& height) {
    int              len = height.size();
    int              ans = 0;
    std::vector<int> left(len + 1, 0);
    std::vector<int> right(len + 1, 0);
    left[0]        = height[0];
    right[len - 1] = height[len - 1];
```

```
    for (int i = 1; i < len; i++) {
      left[i] = (left[i - 1] > height[i] ? left[i - 1] : height[i]);
    }

    for (int i = len - 2; i >= 0; i--) {
      right[i] = (right[i + 1] > height[i] ? right[i + 1] : height[i]);
    }

    for (int i = 0; i < len; i++) {
      ans += std::min(left[i], right[i]) - height[i];
    }

    return ans;
  }
};
```

双指针法

```
class Solution {
public:
  int trap(std::vector<int>& height) {
    int len  = height.size();
    int ans  = 0;
    int left = 0, right = len - 1;
    int left_max = height[0], right_max = height[len - 1];

    while (left <= right) {
      left_max  = std::max(left_max, height[left]);
      right_max = std::max(right_max, height[right]);

      if (left_max < right_max) {
        ans += left_max - height[left];
        left++;
      } else {
        ans += right_max - height[right];
        right--;
      }
    }

    return ans;
  }
};
```