Lecture 30 Stacks

1. Reverse a stack
2. #include <iostream>
3. #include <stack>
4. using namespace std;
5. void transfer(stack<int> &s1, stack<int> &s2, int n)
6. {
7. for(int i=0; i<n; i++)
8. {
9. s2.push(s1.top());
10. s1.pop();
11. }
12. }
13. void reverseStack(stack<int> &s1)
14. {
15. stack<int> s2;
16. int n = s1.size();
17. for(int i=0; i<n; i++)
18. {
19. int x = s1.top();
20. s1.pop();
21. transfer(s1, s2, n-i-1);
22. s1.push(x);
23. transfer(s2, s1, n-i-1);
24. }
25. }
26. int main() {
27. stack<int> s;
28. s.push(1);
29. s.push(2);
30. s.push(3);
31. s.push(4);
32. reverseStack(s);
33. while(!s.empty())
34. {
35. cout<<s.top()<<" ";
36. s.pop();
37. }
38. return 0;
39. }

2. <https://leetcode.com/problems/largest-rectangle-in-histogram/>

class Solution {

public:

int largestRectangleArea(vector<int>& heights) {

stack<int> s;

int n = heights.size();

int tp;

int currArea = 0;

int maxArea = 0;

int i = 0;

while(i < n)

{

while(!s.empty() && heights[i] < heights[s.top()])

{

tp = s.top();

s.pop();

currArea = heights[tp] \* (s.empty() ? i : i-s.top()-1);

maxArea = max(currArea, maxArea);

}

s.push(i);

i++;

}

while(!s.empty())

{

tp = s.top();

s.pop();

currArea = heights[tp] \* (s.empty() ? i : i-s.top()-1);

maxArea = max(currArea, maxArea);

}

return maxArea;

}

};