Max frequency Stack in C++

```
#include <iostream>
#include <unordered_map>
#include <stack>
using namespace std;

class MaxFrequencyStack {
private:
    unordered_map<int, stack<int>> st;
    unordered_map<int, int> fmap;
    int maxfreq;

public:
    MaxFrequencyStack() {
        maxfreq = 0;
    }

    void push(int val) {
        int f = ++fmap[val];
        st[f].push(val);
        maxfreq = max(maxfreq, f);
    }

    int pop() {
        int val = st[maxfreq].top();
        st[maxfreq].pop();
        if (st[maxfreq].empty()) {
            st.erase(maxfreq);
            maxfreq--;
        }
        fmap[val]--;
        return val;
    }
};

int main() {
    MaxFrequencyStack freqStack;

    freqStack.push(5);
    freqStack.push(7);
    freqStack.push(5);
    freqStack.push(7);
    freqStack.push(4);
    freqStack.push(5);

    cout << freqStack.pop() << endl; // Should print 5
    cout << freqStack.pop() << endl; // Should print 7
    cout << freqStack.pop() << endl; // Should print 5
    cout << freqStack.pop() << endl; // Should print 4

    return 0;
}
```

Dry Run: Input Sequence
push(5)
push(7)
push(5)
push(7)
push(4)
push(5)

pop() → ?
pop() → ?
pop() → ?
pop() → ?

📋 Dry Run Table (Tracking fmap, st, and maxfreq):

Operation	fmap	st (per freq)	maxfreq	Top Element Popped
push(5)	{5: 1}	{1: [5]}	1	—
push(7)	{5: 1, 7: 1}	{1: [5, 7]}	1	—
push(5)	{5: 2, 7: 1}	{1: [5, 7], 2: [5]}	2	—
push(7)	{5: 2, 7: 2}	{1: [5, 7], 2: [5, 7]}	2	—
push(4)	{5: 2, 7: 2, 4: 1}	{1: [5, 7, 4], 2: [5, 7]}	2	—
push(5)	{5: 3, 7: 2, 4: 1}	{1: [5, 7, 4], 2: [5, 7], 3: [5]}	3	—
pop()	{5: 2, 7: 2, 4: 1}	3 is [5] → pop 5, delete 3	2	**5**
pop()	{5: 2, 7: 1, 4: 1}	2 is [5, 7] → pop 7	2	**7**
pop()	{5: 1, 7: 1, 4: 1}	2 is [5] → pop 5, delete 2	1	**5**
pop()	{5: 1, 7: 1, 4: 0}	1 is [5, 7, 4] → pop 4	1	**4**

	✔ Output: 5 7 5 4 **💡 Notes:**
5 7 5 4	