難度 2 / 5

題目敘述

You have an infinite number of stacks arranged in a row and numbered (left to right) from `0`, each of the stacks has the same maximum capacity.

Implement the `DinnerPlates` class:

- `DinnerPlates(int capacity)` Initializes the object with the maximum capacity of the stacks `capacity`.
- `void push(int val)` Pushes the given integer `val` into the leftmost stack with a size less than `capacity`.
- `int pop()` Returns the value at the top of the rightmost non-empty stack and removes it from that stack, and returns `-1` if all the stacks are empty.
- `int popAtStack(int index)` Returns the value at the top of the stack with the given index `index` and removes it from that stack or returns `-1` if the stack with that given index is empty.

Example 1:

```
Input
["DinnerPlates", "push", "push", "push", "push", "push", "popAtStack", "push", "push",
"popAtStack", "popAtStack", "pop", "pop", "pop", "pop", "pop"]
[[2], [1], [2], [3], [4], [5], [0], [20], [21], [0], [2], [], [], [], [], []]
Output
[null, null, null, null, null, null, 2, null, null, 20, 21, 5, 4, 3, 1, -1]
```

參考答案

```
class DinnerPlates {
private:
    int n;
    vector<vector<int>> plates;
    set<int> st;
    int top;
public:
    DinnerPlates(int capacity): n(capacity), top(-1) {

    }

    void push(int val) {
        if (st.empty()) {
            st.insert(++top);
            plates.push_back({});
        }

        int i = *st.begin();
        vector<int> &v = plates[i];
        top = max(top, i);
        v.emplace_back(val);
        if (v.size() == n) st.erase(i);
    }

    int pop() {
```

```cpp
        return popAtStack(top);
    }

    int popAtStack(int index) {
        if (index == -1 or index >= plates.size()) return -1;
        vector<int> &v = plates[index];
        if (v.empty()) return -1;

        int ret = v.back(); v.pop_back();
        if (v.size() == n - 1) st.insert(index);
        while (top >= 0 and plates[top].empty()) --top;

        return ret;
    }
};
```