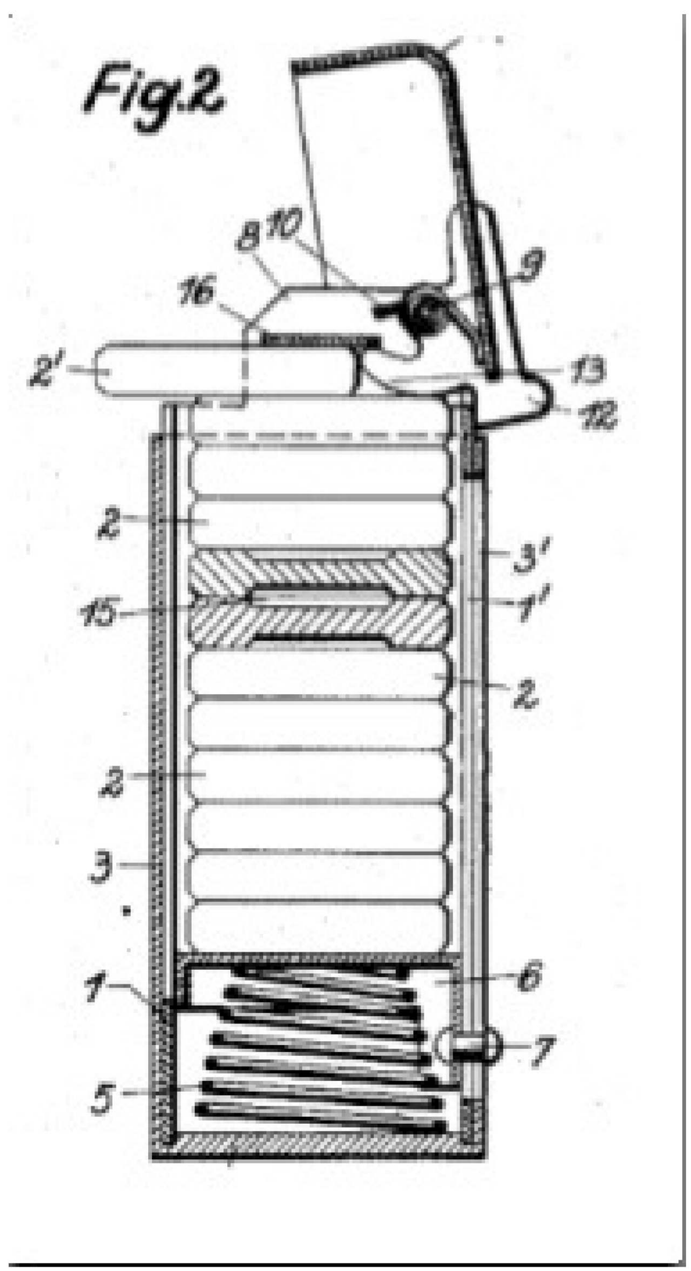
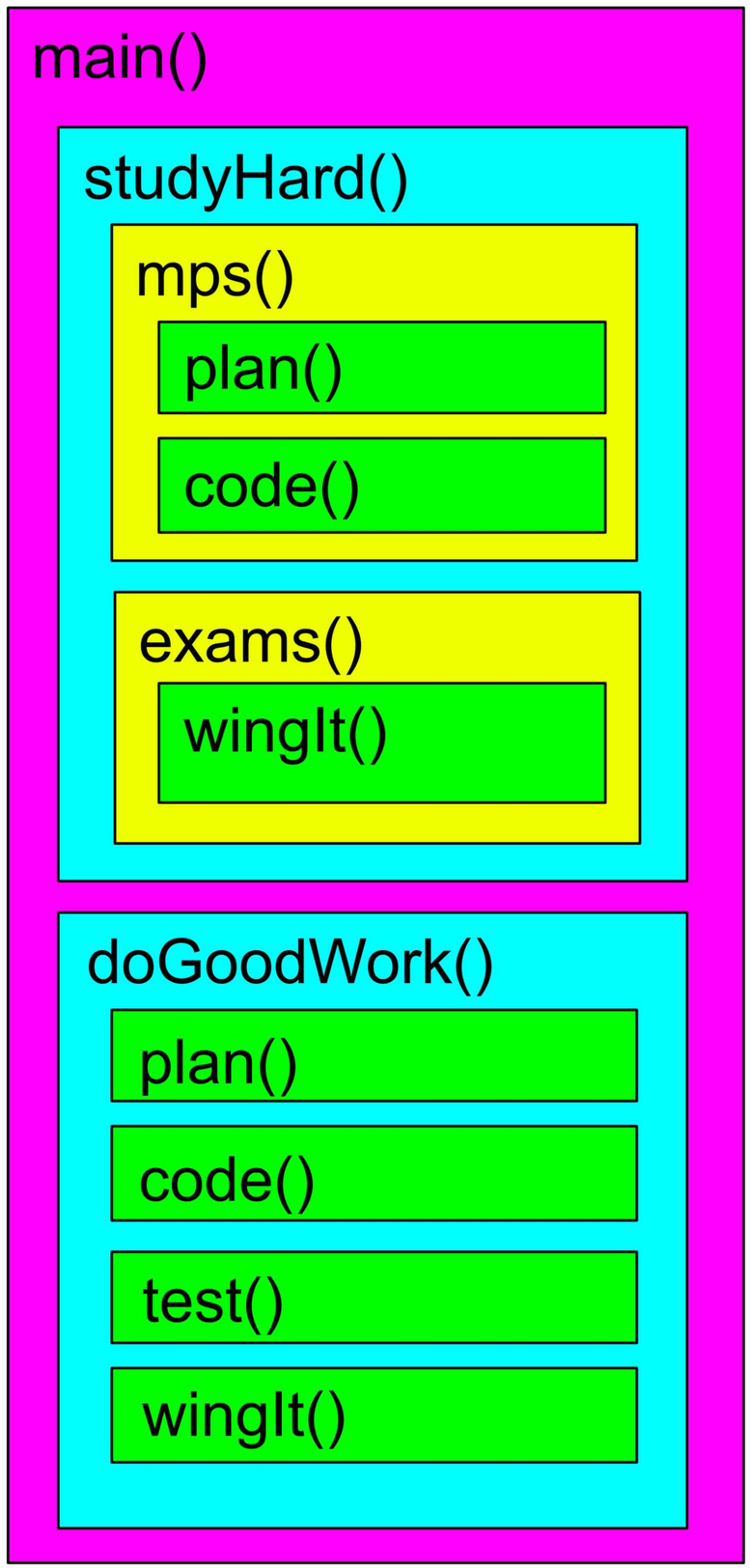
Stacks:



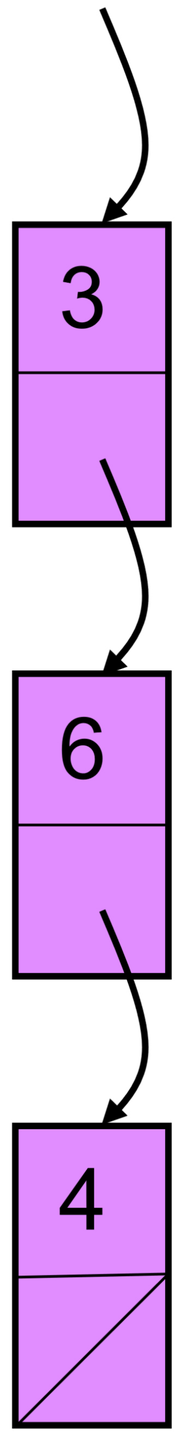


({ } ( ) [(( )) { (( )) ( ) }] ( ))

4 5 + 7 2 - * 3 - 6 /

Stack linked memory implementation:

```
template<class SIT>
class Stack {
public:
    Stack();
    ~Stack(); // etc.
    bool empty() const;
    void push(const SIT & e);
    SIT pop();
private:
    struct stackNode {
        SIT data;
        stackNode * next;
    };
    stackNode * top;
    int size;
};
```




```
template<class SIT>
SIT Stack<SIT>::pop(){




}
```

```
template<class SIT>
void Stack<SIT>::push(const SIT & d){
stackNode * newNode = new stackNode(d);
newNode->next = top;
top = newNode;
}
```

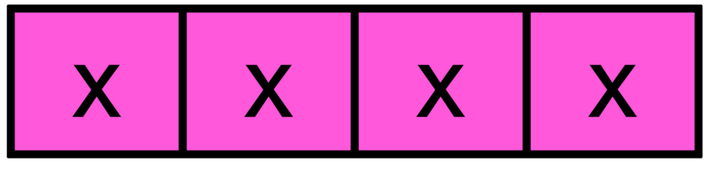
Stack array based implementation:

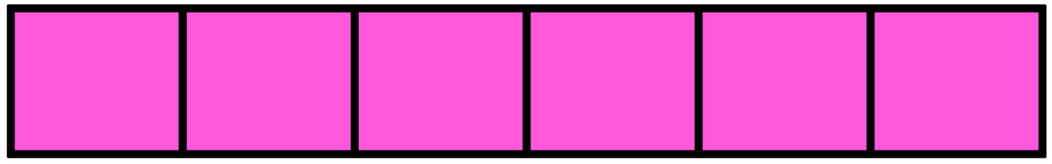
```
template<class SIT>
class Stack {
public:
    Stack();
    ~Stack(); // etc.
    bool empty() const;
    void push(const SIT & e);
    SIT pop();
private:
    int capacity;
    int size;
    SIT * items;
};
```

```
template<class SIT>
void Stack<SIT>::push(const SIT & e){
    if (size >= capacity) {
      // grow array somehow
    }
    items[size] = e;
    size ++;
}
```

Stack array based implementation: (what if array fills?)

Analysis holds for array based implementations of Lists, Stacks, Queues, Heaps…

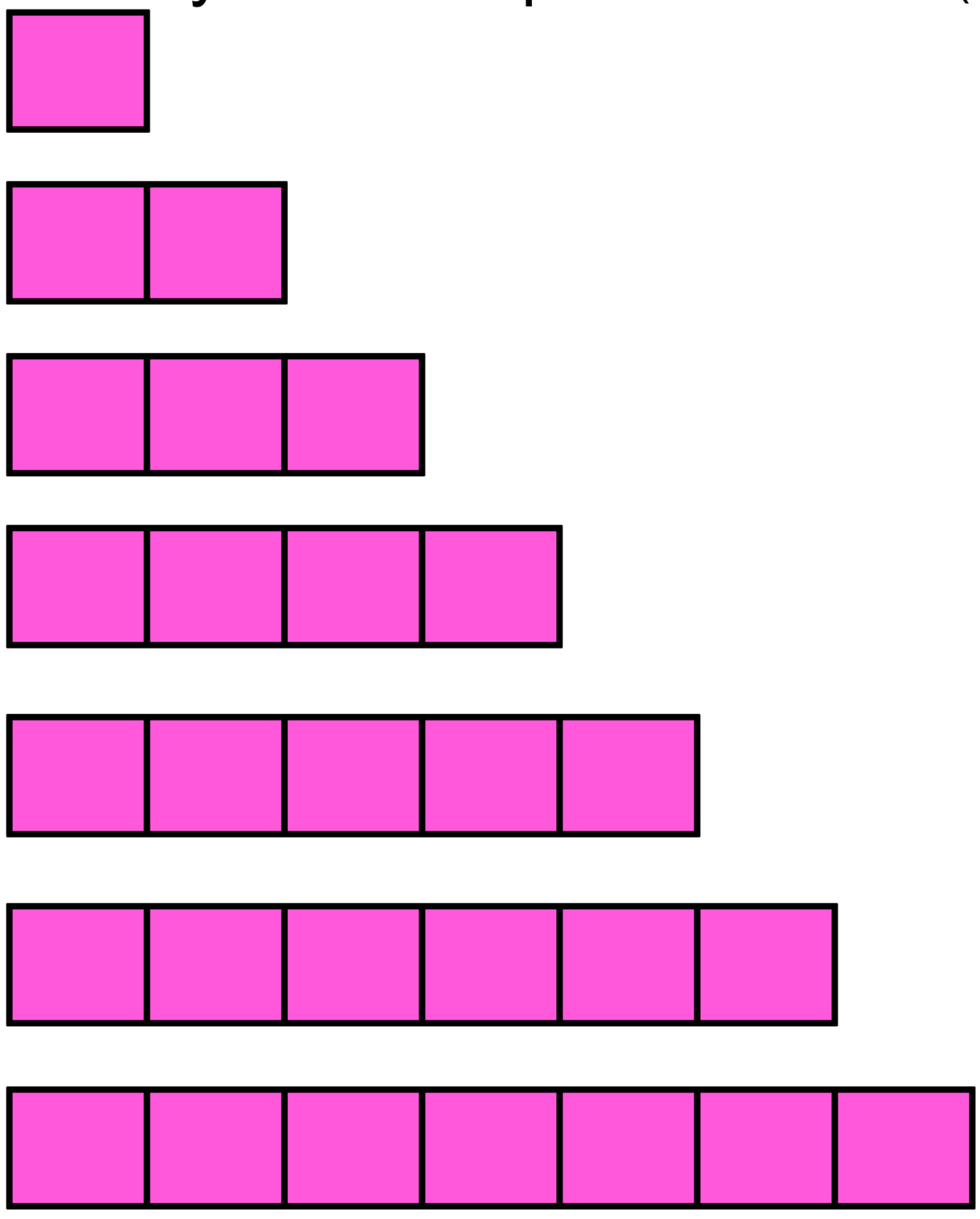




General Idea: upon an insert (push), if the array is full, create a larger space and copy the data into it.

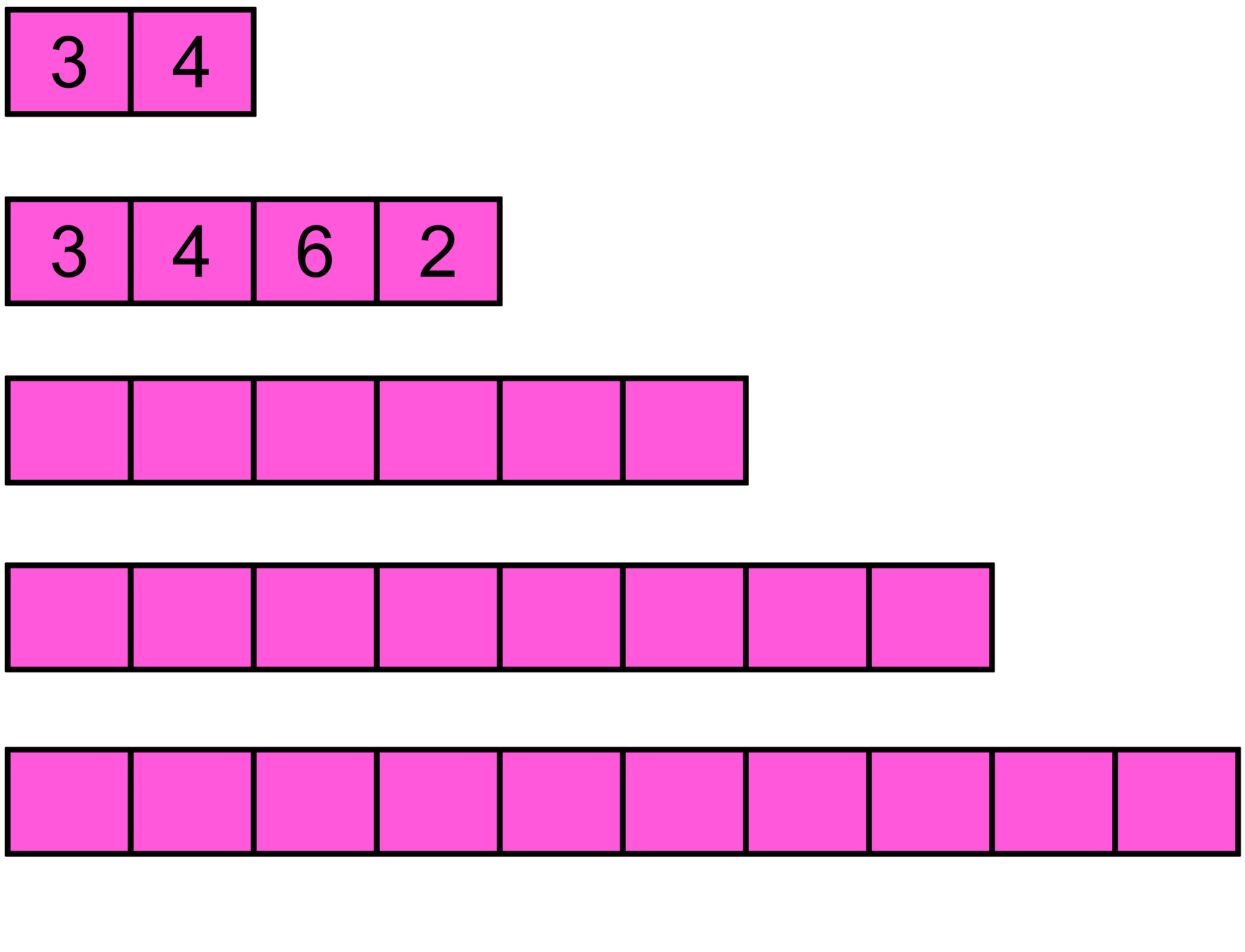
Main question: What's the resizing scheme? We examine 3.

Stack array based implementation: (what if array fills?)



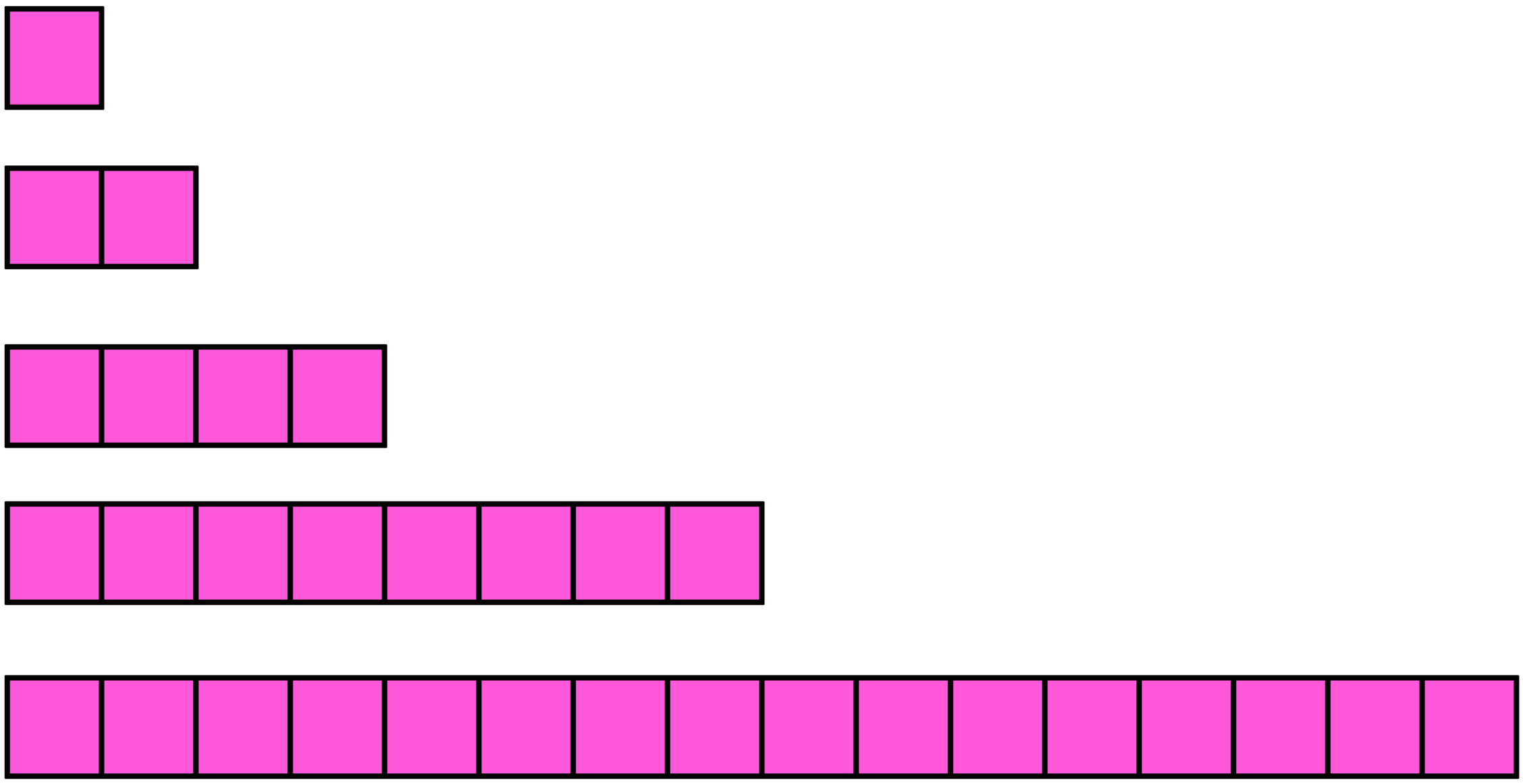
How does this scheme do on a sequence of n pushes?

Stack array based implementation: (what if array fills?)



How does this scheme do on a sequence of n pushes?

Stack array based implementation: (what if array fills?)



How does this scheme do on a sequence of n pushes?

Summary

Linked list based implementation of a stack:

- Constant time push and pop

Array based implementation of a stack:

- _____ time pop.
- _____ time push if capacity exists

Cost over O(n) pushed is _____ for an average of _____ per push.

Why consider an array?