



Stack

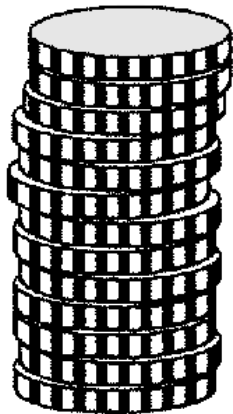
Data structure & Algorithms



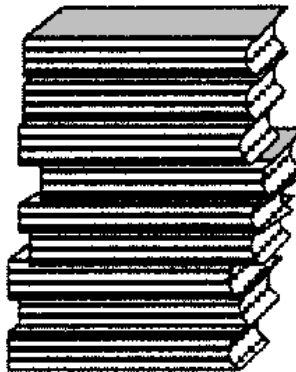
Stack

- ❖ Stack เป็นโครงสร้างข้อมูลแบบ LIFO (Last-In, First-Out)
- ❖ Operations พื้นฐานของ Stack ได้แก่
 - การนำข้อมูลเข้าสู่ Stack เรียกว่า Push
 - การนำข้อมูลออกจาก Stack เรียกว่า Pop
 - การเรียกใช้ข้อมูลจาก Stack เรียกว่า Top
- ❖ การสร้าง Stack
 - ใช้ Array แทน Stack
 - ใช้ Linked list แทน Stack

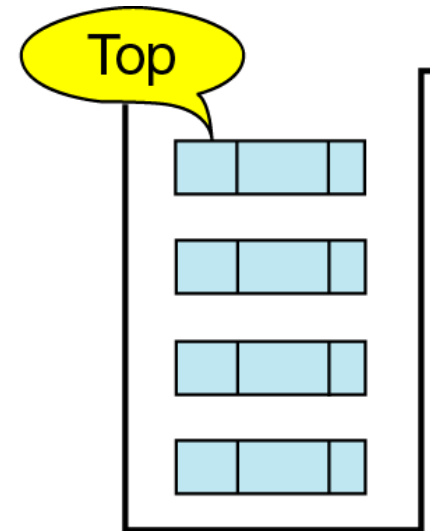
Stack



Stack of coins

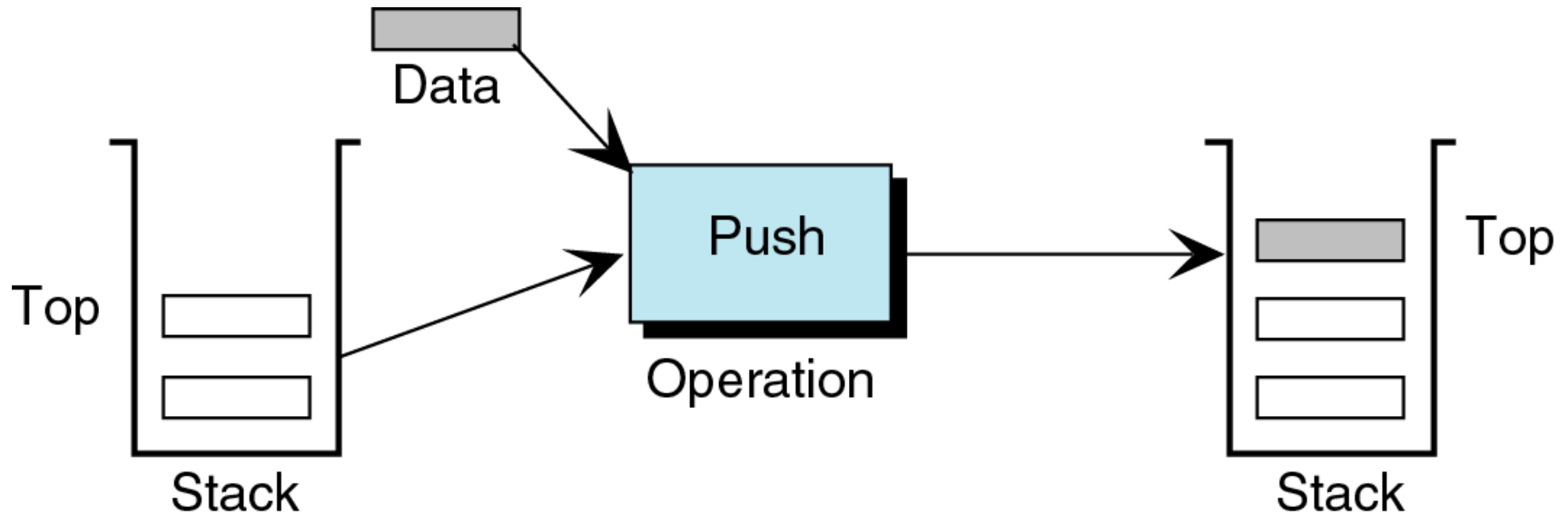


Stack of books

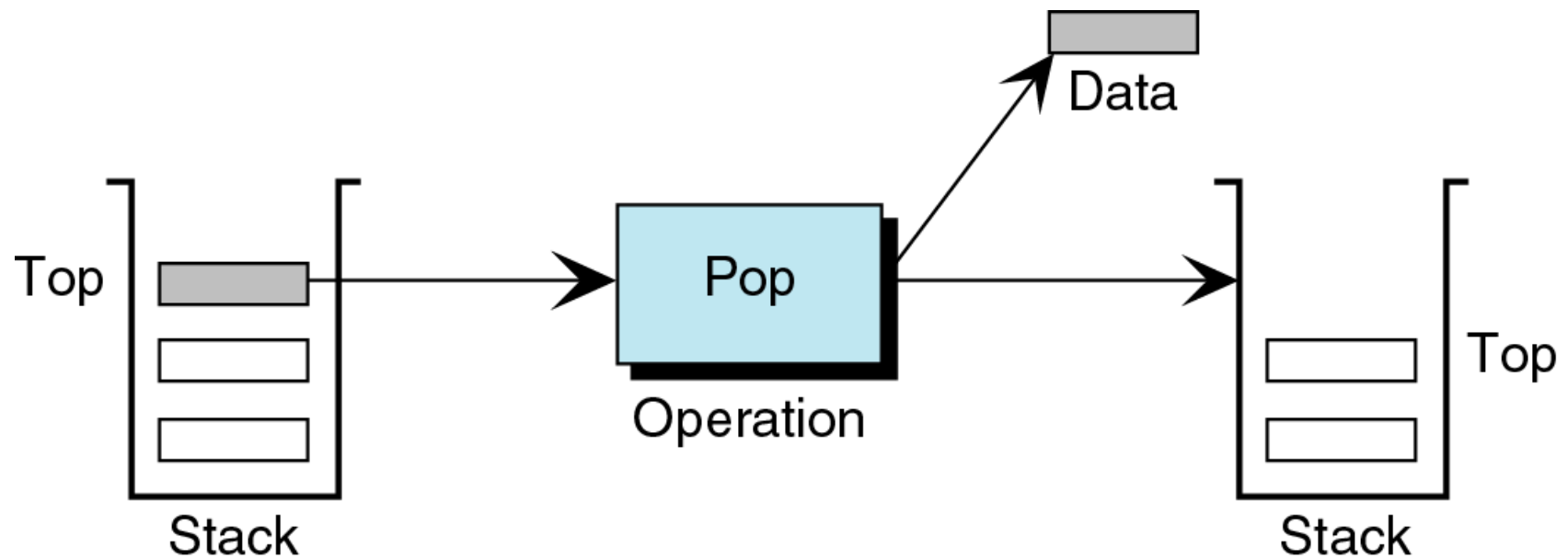


Computer stack

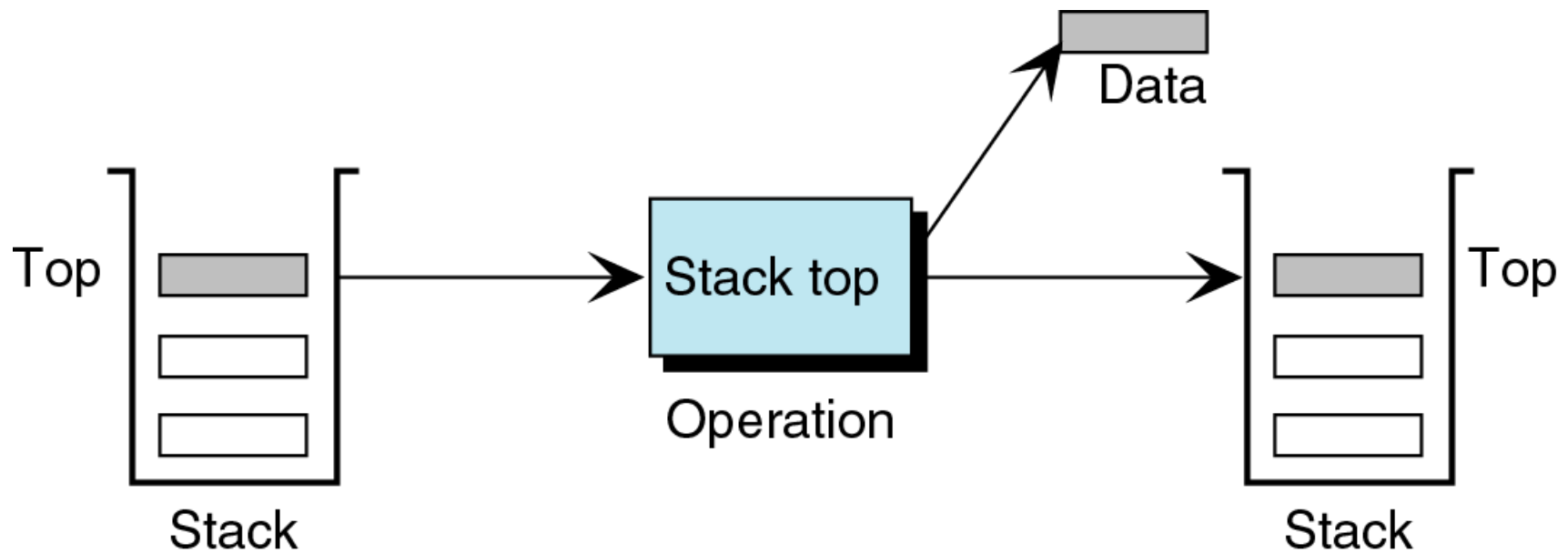
เพิ่มข้อมูลใน *Stack*: *Push*



นำข้อมูลออกจาก Stack : Pop

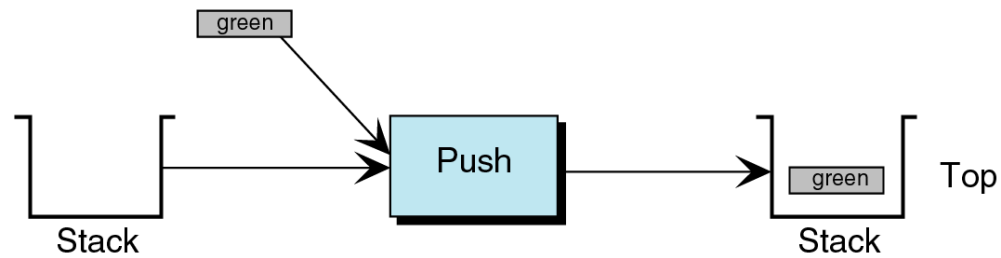


เรียกใช้ข้อมูลใน *Stack: Top*

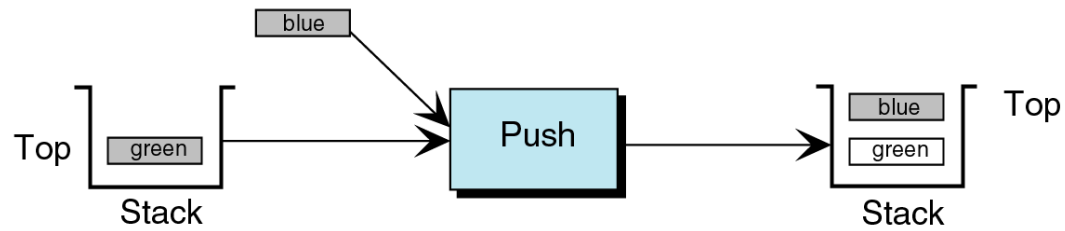




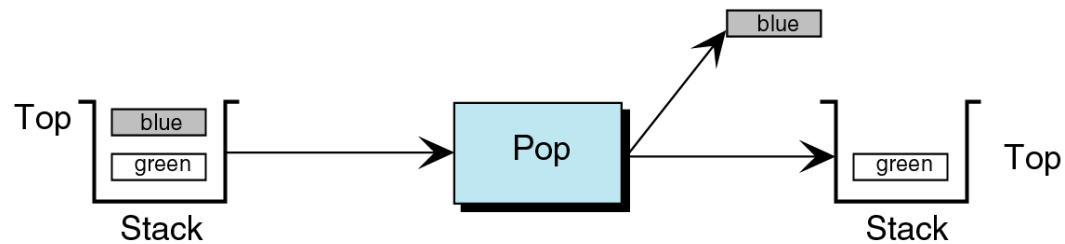
Step 1



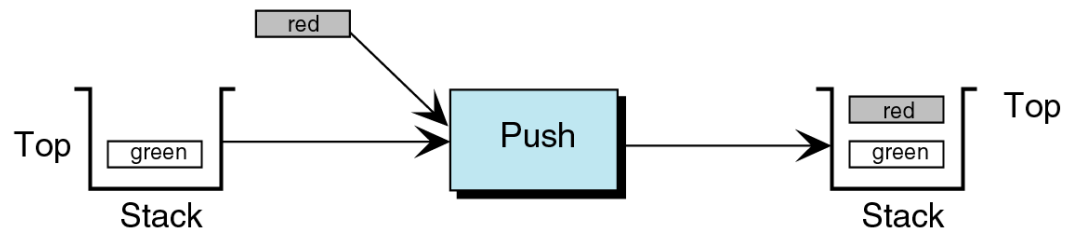
Step 2

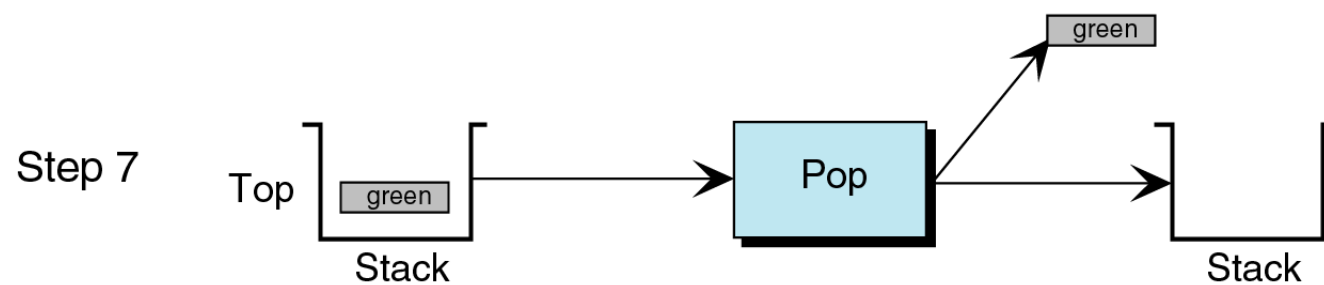
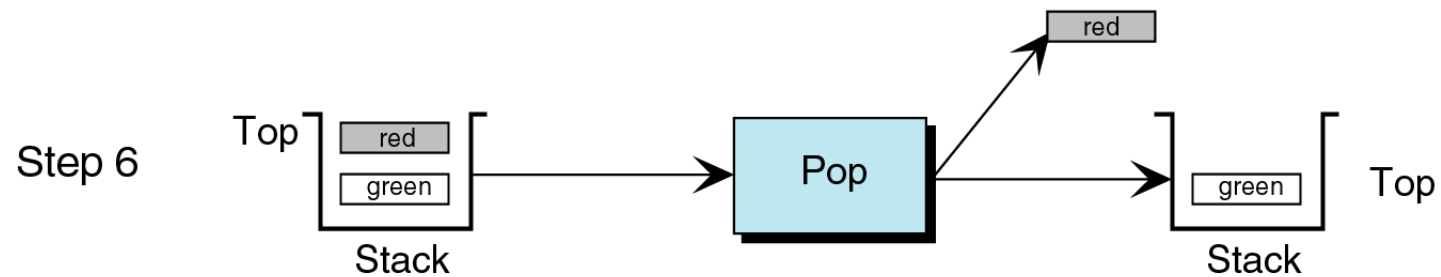
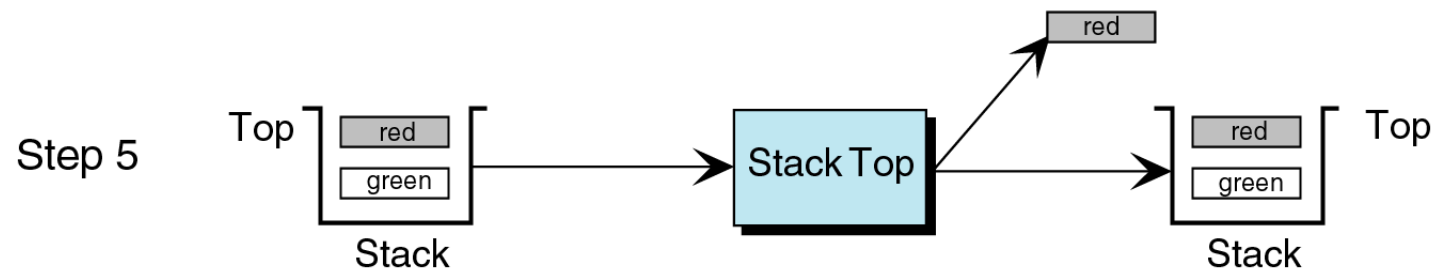


Step 3

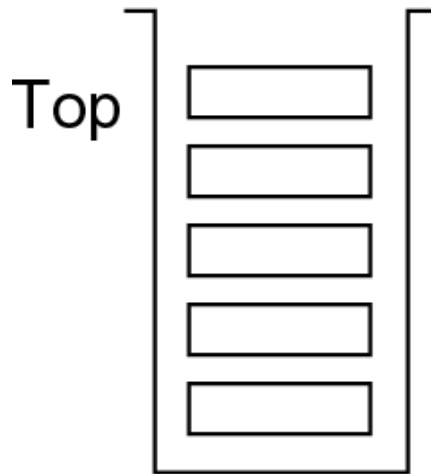


Step 4

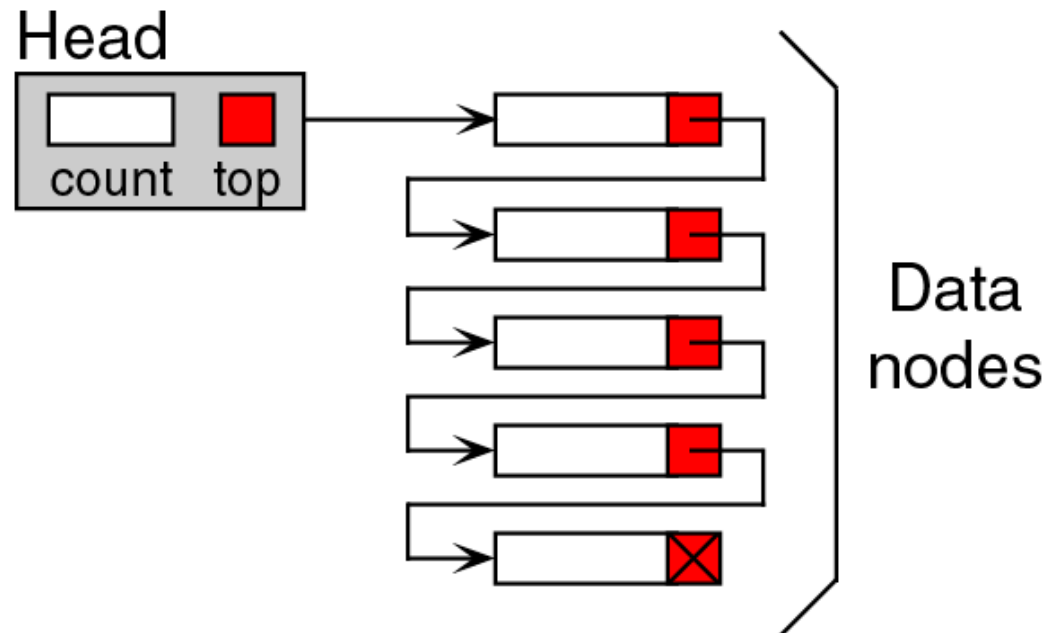




Linked list မှတ်စု *Stack*

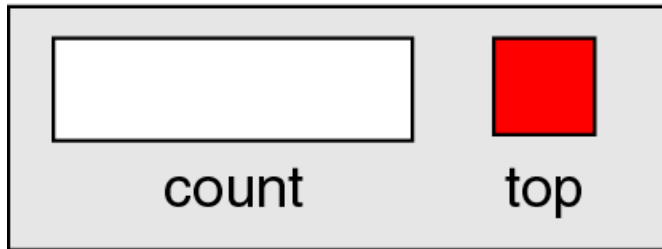


(a) Conceptual

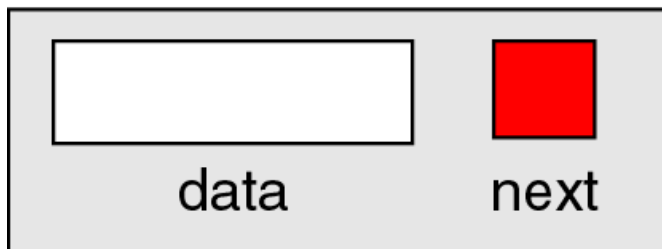


(b) Physical

Linked list and Stack



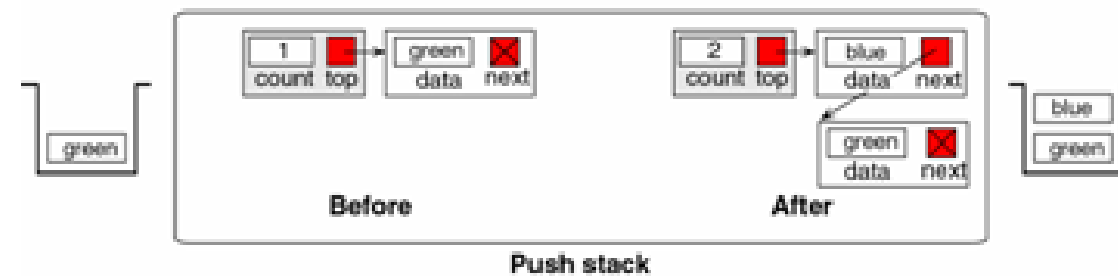
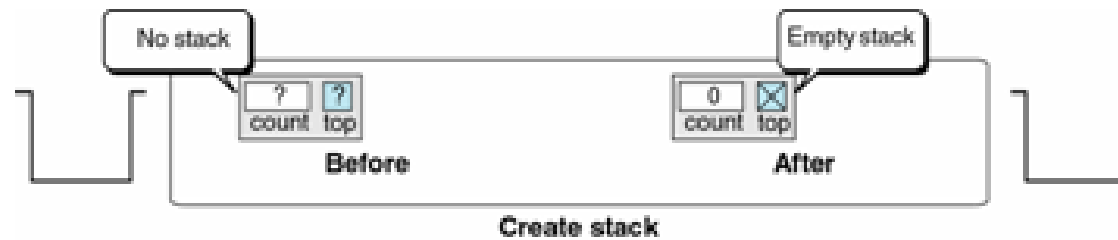
Stack head structure

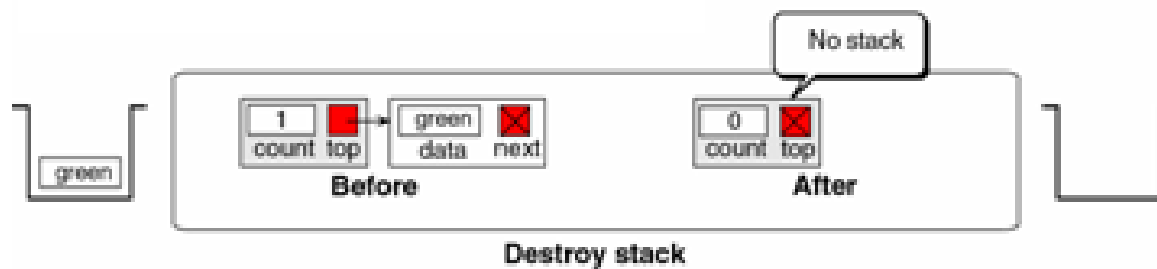
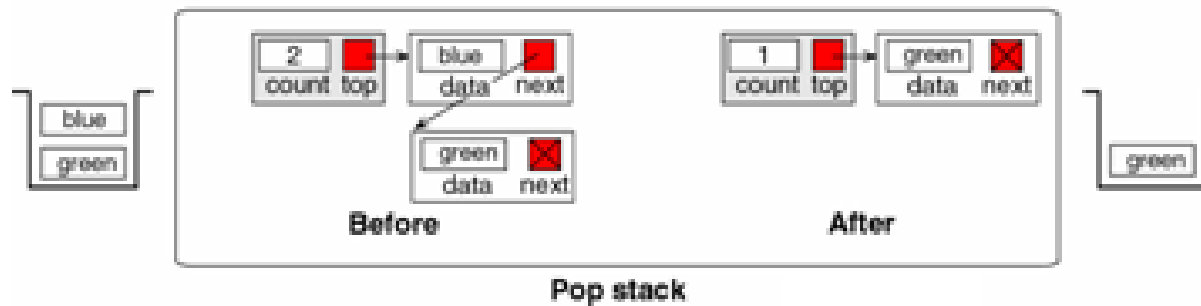


Stack node structure

```
stack  
  count <integer>  
  top   <node pointer>  
end stack
```

```
node  
  data <dataType>  
  next <node pointer>  
end node
```



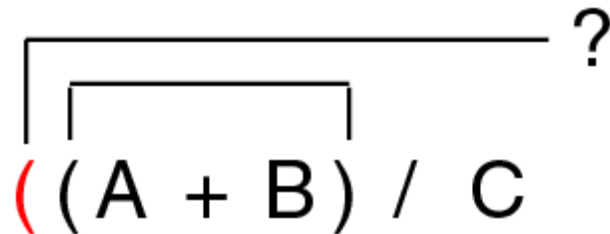




Operations พื้นฐานของ Stack ที่สร้างด้วย Linked list

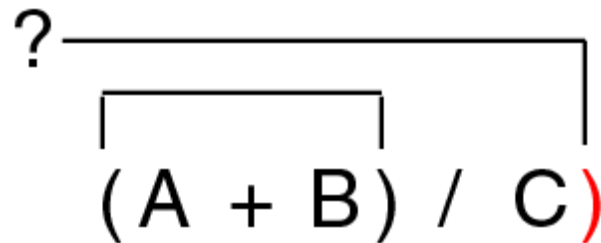
- | | |
|-------------------|--|
| 1. Create stack: | สร้าง stack head node |
| 2. Push stack: | เพิ่มรายการใน stack |
| 3. Pop stack: | ลบรายการใน stack |
| 4. Stack top: | เรียกใช้รายการข้อมูลที่อยู่บนสุดของ stack |
| 5. Empty stack: | ตรวจสอบว่า stack ว่างเปล่าหรือไม่ |
| 6. Full stack: | ตรวจสอบว่า stack เต็มหรือไม่ |
| 7. Stack count: | ส่งค่าจำนวนรายการใน stack |
| 8. Destroy stack: | คืนหน่วยความจำของทุก node ใน stack ให้ระบบ |

Stack Applications: Balancing Symbols



A diagram illustrating an unbalanced expression. The expression is $((A + B) / C$. A bracket connects the opening parenthesis at the start to a question mark above the closing parenthesis, indicating that the opening parenthesis is not matched.

(a) Opening parenthesis not matched



A diagram illustrating an unbalanced expression. The expression is $(A + B) / C)$. A bracket connects the closing parenthesis at the end to a question mark above the opening parenthesis, indicating that the closing parenthesis is not matched.

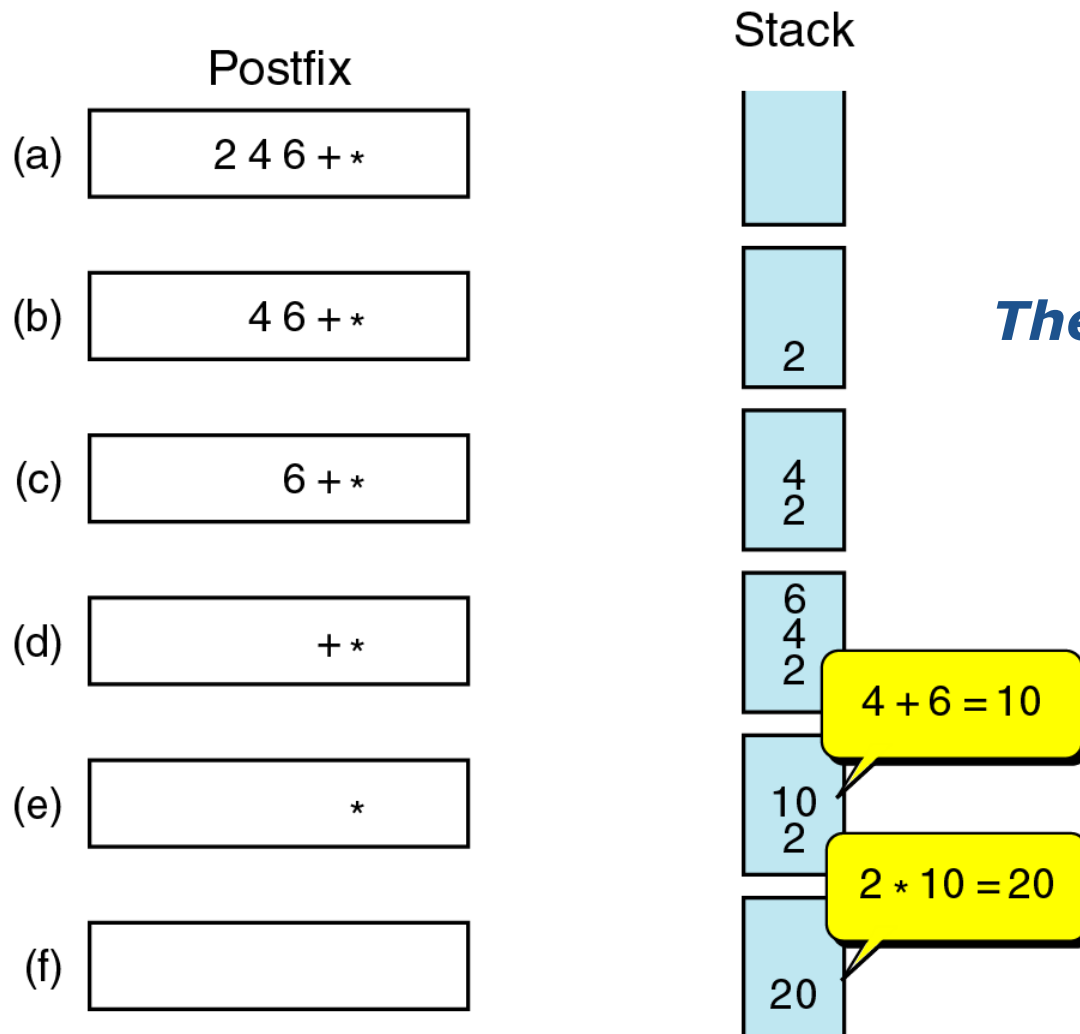
(b) Closing parenthesis not matched

Stack Applications: Infix to Postfix conversion

	Infix	Stack	Postfix
(a)	A+B*C-D/E		
(b)	+B*C-D/E		A
(c)	B*C-D/E	+	A
(d)	*C-D/E	+	AB
(e)	C-D/E	* +	AB
(f)	-D/E	* +	ABC
(g)	D/E	-	ABC++
(h)	/E	-	ABC++D
(i)	E	/ -	ABC++D
(j)		/ -	ABC++DE
(k)			ABC++DE/-

*The conversion time
is $O(n)$*

Postfix expression evaluation



The evaluation time is $O(n)$

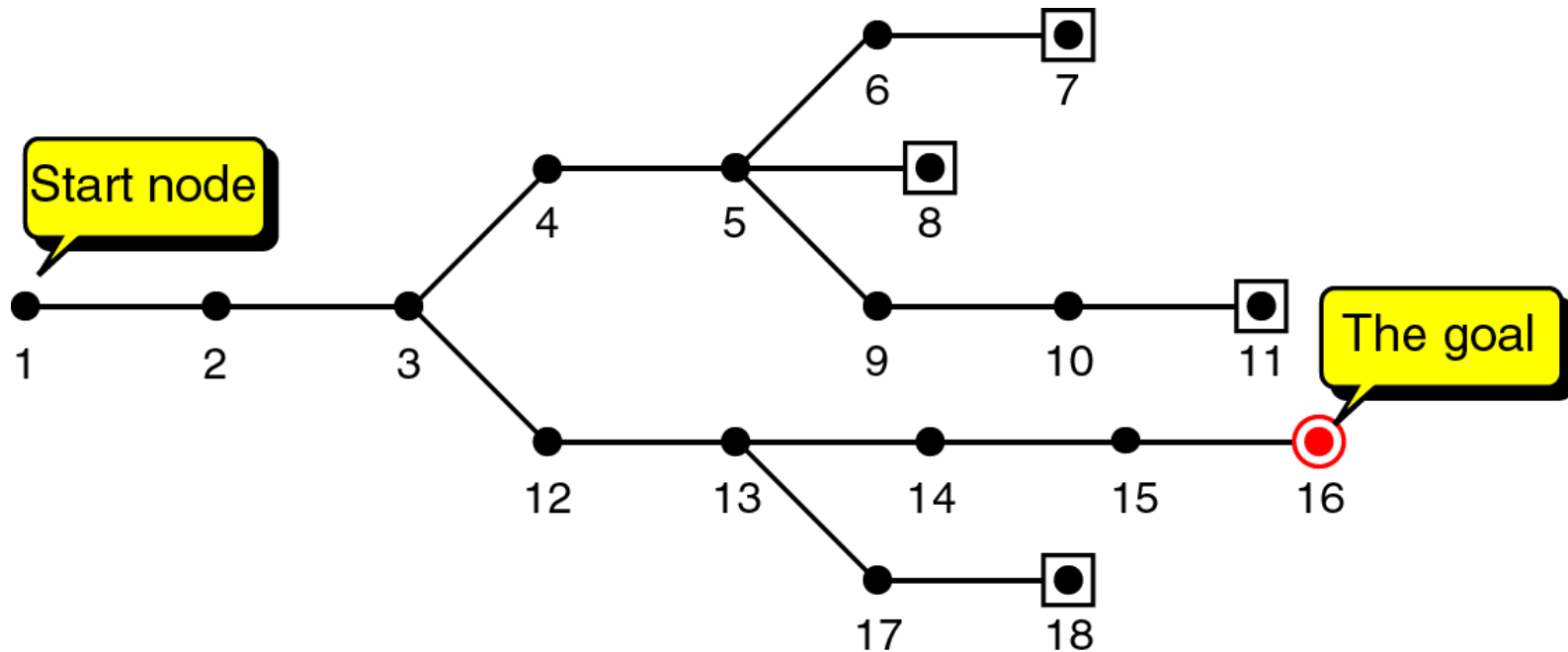


Backtracking

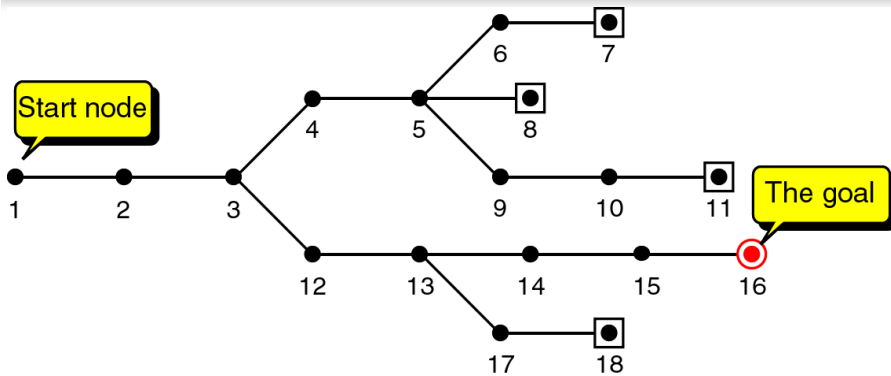
Backtracking คือวิธีการหาคำตอบโดยเดินหน้าไปยังเป้าหมาย เมื่อถึงทางแยกก็จะต้องตัดสินใจเลือกเส้นทางใดเส้นทางหนึ่งเดินหน้าต่อไปเพื่อหาเป้าหมาย หากเดินไปจนสุดเส้นทางแล้วยังไม่พบเป้า ก็จะเดินย้อนกลับมายังจุดแยกครั้งสุดท้ายแล้วเลือกเส้นทางใหม่ที่ยังไม่เคยไปทำเช่นนี้ไปเรื่อย ๆ จนกว่าจะพบเป้าหมาย หรือจนครบทุกเส้นทาง

Backtracking เป็นการประยุกต์ใช้โครงสร้างข้อมูลแบบ **Stack** สำหรับการเขียนโปรแกรมประเภทเกมส์คอมพิวเตอร์ (computer gaming) การวิเคราะห์การตัดสินใจ(decision analysis) และระบบผู้เชี่ยวชาญ(expert system) ตัวอย่างปัญหาที่ใช้วิธี **Backtracking** เช่นปัญหาการค้นหเป้าหมาย (goal seeking) และ ปัญหา 8 ราชีนี (eight queens problem)

Stack Applications: Backtracking



Stack Applications: Backtracking



B12
3
2
1

(a)

B8
B9
5
4
B12
3
2
1

(b)

end
7
6
B8
B9
5
4
B12
3
2
1

(c)

end
8
B9
5
4
B12
3
2
1

(d)

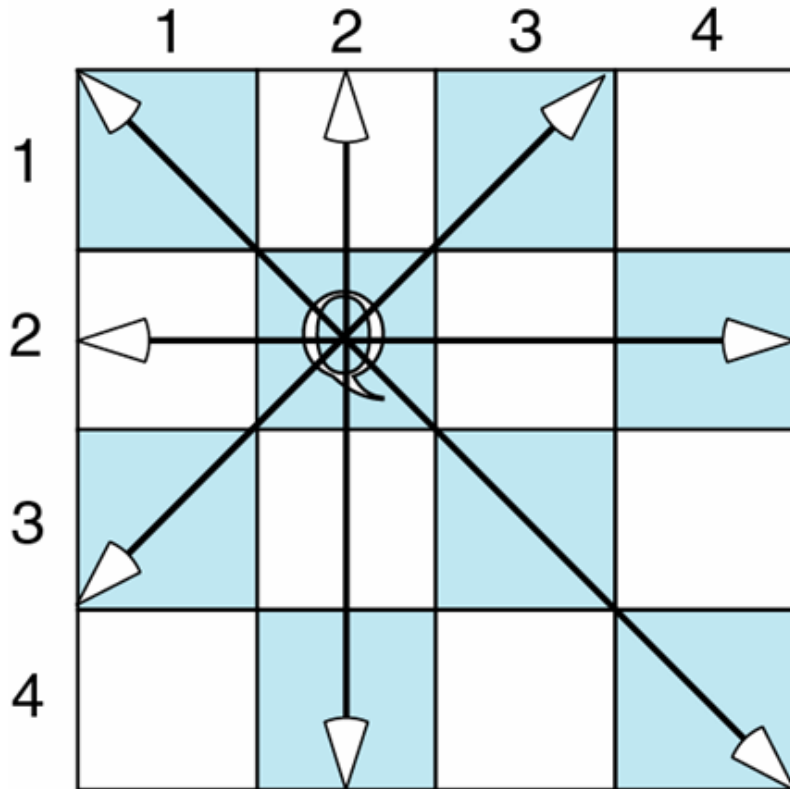
end
11
10
9
5
4
B12
3
2
1

(e)

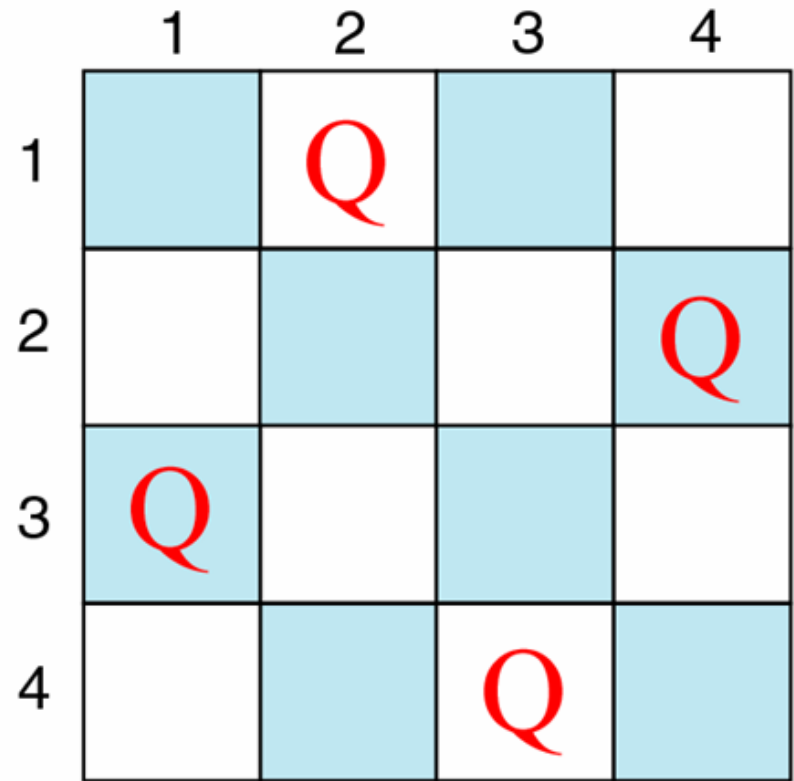
goal
16
15
14
B17
13
12
3
2
1

(f)

Stack Applications: Backtracking



(a) Queen capture rules



(b) First four queens solution

Stack Applications: Backtracking

