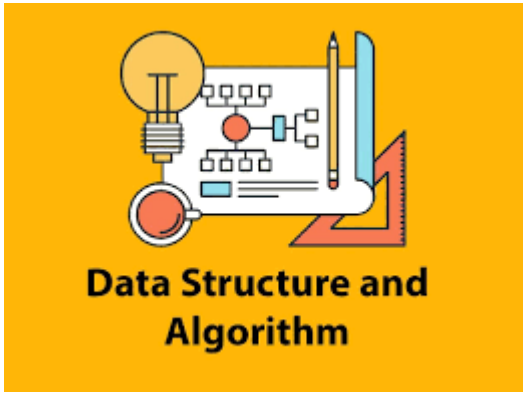
CẤU TRÚC DỮ LIỆU & GIẢI THUẬT

NGÀY 3.1: NGĂN XẾP - STACK



Giảng viên: Th.S Bùi Văn Kiên

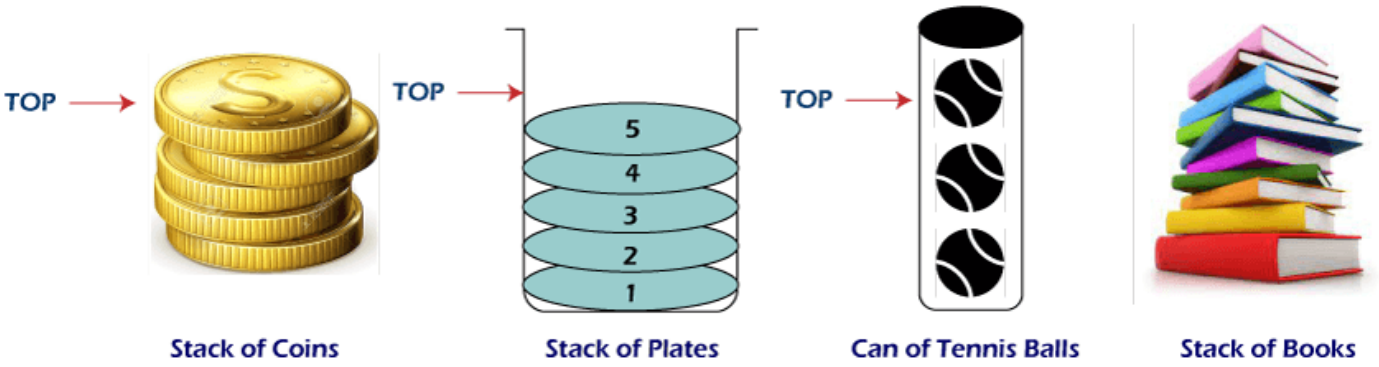
NỘI DUNG

- Khái niệm
  - Ví dụ với thư viện chuẩn
  - Triển khai với mảng
  - Triển khai với Linked list
- Các bài toán ứng dụng:
  - Dãy ngoặc đúng
  - Dùng Stack để khử đệ quy
  - Chuyển đổi biểu thức trung tố - hậu tố
  - Tính toán giá trị biểu thức hậu tố
  - Tìm phần tử đầu tiên bên phải lớn hơn A[i]

1.1 Khái niệm

- Ngăn xếp là một danh sách đặc biệt, trong đó các phép toán chỉ được thực hiện ở một đầu của danh sách.

- Tính chất: vào trước ra sau (FILO – First in Last Out)



1.1 Khái niệm

Trừu tượng hóa cấu trúc ngăn xếp

- Đặc tả dữ liệu

A = (a_0, a_1, …, a_{n-1}) trong đó a_{n-1} là đỉnh ngăn xếp

- Các thao tác:
  - Kiểm tra ngăn xếp có rỗng hay không: isEmpty()
  - Kiểm tra ngăn xếp có đầy hay không: isFull()
  - Trả về phần tử ở đỉnh ngăn xếp: top()
  - Thêm phần tử x vào đỉnh ngăn xếp: push(x)
  - Loại phần tử ở đỉnh ngăn xếp: pop()
  - Đếm số phần tử của ngăn xếp: size()

1.1 Stack của thư viện chuẩn STL

- #include <stack>

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

int main() {
    stack<int> S;
    S.push(1);
    S.push(2);
    S.push(3);

    while(!S.empty()) {
        cout << S.top() << endl;
        S.pop();
    }
    return 0;
}
```

Stack
Head = Empty
1 (Head)
1 2 (Head)
1 2 3 (Head)

→ 1 2 (Head)
→ 1 (Head)

Output = 3 2 1

1.2 Biểu diễn stack bằng mảng

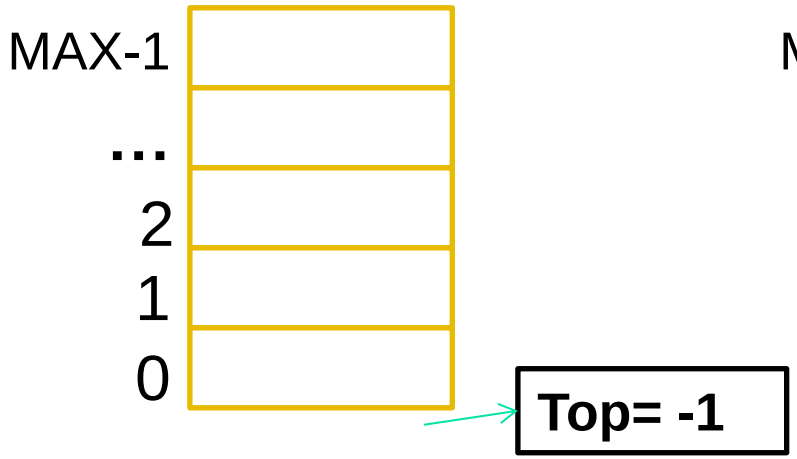
- Khai báo struct:

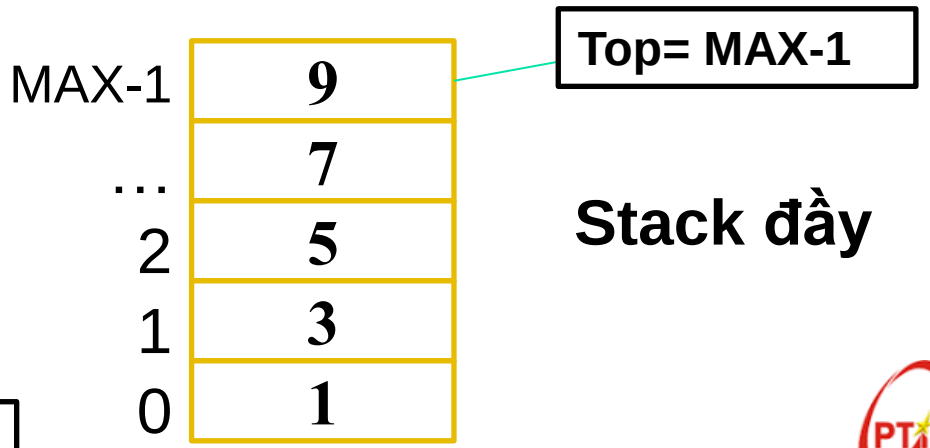
```
#define MAX 1000

struct Stack {
    int top;
    int a[MAX];

    Stack() { top = -1; }
    bool push(int x);
    int pop();
    int topElement();
    bool isEmpty();
};
```

Stack rỗng





Stack đầy

1.2 Biểu diễn stack bằng mảng

- Thao tác 1: Kiểm tra tính rỗng của stack

```
bool Stack::isEmpty() {
    if (top < 0) return 1;
    return 0;
}
```

- Thao tác 2: Kiểm tra tính đầy của stack

```
bool Stack::isFull() {
    if (top == MAXX-1) return 1;
    return 0;
}
```

1.2 Biểu diễn stack bằng mảng

- Thao tác 3: Trả về phần tử đầu tiên của danh sách

```
int Stack::topElement() {
    if (top < 0) {
        cout << "Stack is Empty\n";
        return 0;
    }
    else {
        int x = a[top];
        return x;
    }
}
```

1.2 Biểu diễn stack bằng mảng

- Thao tác 4: Đưa dữ liệu vào danh sách

Chỉ được thực hiện khi và chỉ khi ngăn xếp chưa tràn.

```
bool Stack::push(int x) {
    if (top >= (MAX - 1)) {
        cout << "Stack Overflow\n";
        return false;
    }
    else {
        a[++top] = x;
        cout << x << " pushed into stack\n";
        return true;
    }
}
```

a[++top] tương đương với
top = top+1
a[top] = x

1.2 Biểu diễn stack bằng mảng

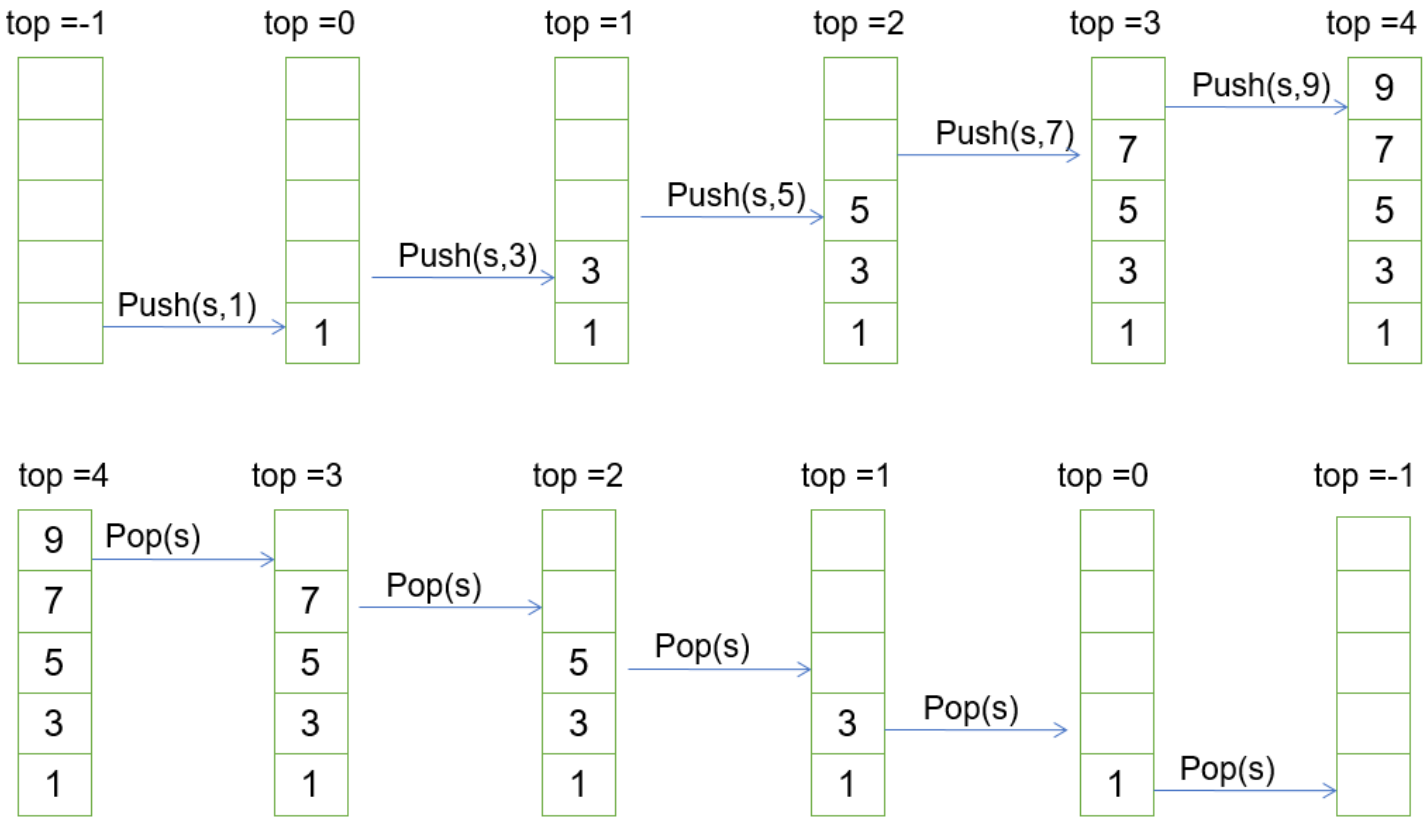
- Thao tác 5: Lấy dữ liệu ra khỏi danh sách

Chỉ được thực hiện khi và chỉ khi ngăn xếp chưa tràn.

```
int Stack::pop() {
    if (top < 0) {
        cout << "Stack Underflow\n";
        return 0;
    }
    else {
        int x = a[top--];
        return x;
    }
}
```

1.2 Biểu diễn stack bằng mảng

- Ví dụ: stack có giới hạn 5 phần tử



1.3 Biểu diễn stack bằng DSLK

- Khai báo struct:

```
struct Stack {
    struct Node {
        int data;
        Node* next;

        Node(int n) {
            this->data = n;
            this->next = nullptr;
        }
    };

    Node* top;
    Stack() { top = nullptr; }
```

1.3 Biểu diễn stack bằng DSLK

- Thao tác 1: Kiểm tra tính rỗng của stack

```
bool isEmpty() {
    return top == nullptr;
}
```

- Thao tác 2: Đưa dữ liệu vào danh sách

```
void push(int data) {
    Node* temp = new Node(data);
    temp->data = data;
    temp->next = top;
    top = temp;
}
```

1.3 Biểu diễn stack bằng DSLK

- Thao tác 3: Lấy phần tử đầu tiên của danh sách

→ Return top->data

- Thao tác 4: Đẩy phần tử ra khỏi danh sách

```
void pop() {
    Node* temp;
    if (top == nullptr) {
        cout << "Stack Underflow" << endl;
        return;
    }
    else {
        temp = top;
        top = top->next;
        free(temp);
    }
}
```

2. Các bài toán áp dụng

- Dãy ngoặc đúng
- Dùng Stack để khử đệ quy
- Chuyển đổi biểu thức trung tố - hậu tố
- Tính toán giá trị biểu thức hậu tố
- Tìm phần tử đầu tiên bên phải lớn hơn A[i]

2-1. Kiểm tra dãy ngoặc đúng

- Biểu thức ngoặc là xâu chỉ gồm các ký tự '(' hoặc ')'. Biểu thức ngoặc đúng và bậc của biểu thức ngoặc được định nghĩa một cách đệ qui như sau:
  - Biểu thức rỗng là biểu thức ngoặc đúng và có bậc bằng 0,
  - Nếu A là biểu thức ngoặc đúng có bậc bằng **k** thì (A) cũng là một biểu thức ngoặc đúng có bậc bằng **k+1**,
  - Nếu A và B là hai biểu thức ngoặc đúng và có bậc tương ứng là k1 và k2 thì AB cũng là một biểu thức ngoặc đúng có bậc bằng max(k1, k2).
- Ví dụ, '**()(())**' là một biểu thức ngoặc đúng có bậc bằng 2, còn '**((()(())))**' là một biểu thức ngoặc đúng và có bậc bằng 3.

2-1. Kiểm tra dãy ngoặc đúng

- Bài toán 1:

Cho xâu S chỉ gồm ký tự (và). Kiểm tra xem S có phải là dãy ngoặc đúng không?

- Bài toán 2:

Nếu S là dãy ngoặc đúng, với mỗi vị trí trong S, ban cần in ra vi trí của dấu ngoặc tương ứng.

In ra cặp (mở, đóng) theo thứ tự tăng dần

(1, 6), (2, 3), (4, 5), (7, 10), (8, 9)

S = ((()())(()))

- Bài toán 3:

Cho số nguyên **k** và xâu S là một xâu chỉ gồm các ký tự '(', ')' và '?', hãy đếm số cách cách thay các ký tự '?' trong xâu S thành ký tự '(' hoặc ')' để nhận được xâu T là biểu thức ngoặc đúng có bậc bằng k.

2-1. DSA07021

DÃY NGOẶC ĐÚNG DÀI NHẤT

Cho một xâu chỉ gồm các kí tự '(' và ')'. Một dãy ngoặc đúng được định nghĩa như sau:

- Xâu rỗng là 1 dãy ngoặc đúng.
- Nếu A là 1 dãy ngoặc đúng thì (A) là 1 dãy ngoặc đúng.
- Nếu A và B là 2 dãy ngoặc đúng thì AB là 1 dãy ngoặc đúng.

Cho một xâu S. Nhiệm vụ của bạn là hãy tìm dãy ngoặc đúng dài nhất xuất hiện trong xâu đã cho.

Input: Dòng đầu tiên là số lượng bộ test T ($T \leq 20$).

Mỗi test gồm một xâu S có độ dài không vượt quá 10^5 kí tự.

Output: Với mỗi test in ra một số nguyên là độ dài dãy ngoặc đúng dài nhất tìm được.

Ví dụ:

Input:	Output
3 ((( )()()) ()((()))))	2 4 6

2-1. Kiểm tra dãy ngoặc đúng

- Bài toán con:

Cho xâu S là một xâu chỉ gồm các ký tự ‘(‘, ‘)’ và ‘?’.

Hãy viết chương trình sinh ra các cấu hình bằng cách thay dấu ‘?’ bằng kí tự ‘(‘ và ‘)’.

- **Ví dụ:**
- Input:

()??

- Output:
  - ()((
  - ()()
  - ())(
  - ()))

2.6 Biểu thức dãy ngoặc đúng

- **Input:**
  - Dòng đầu tiên chứa số nguyên k
  - Dòng thứ hai chứa xâu S, gồm các kí tự (,) và ?
- **Output:**

In ra một số nguyên là số cách thay kí tự ? để được dãy ngoặc đúng có bậc bằng k.

- **Giới hạn:**
  - 30% số test có độ dài xâu S <= 20
  - **30% số test có độ dài xâu S <= 64**
  - 40% số test có độ dài xâu S <= 200

(())() → bậc = 2

- **Ví dụ:**

Input	Output
????(? k = 2	1
?????? k = 2	

2-1. Kiểm tra dãy ngoặc đúng

- Nếu S là dãy ngoặc đúng, với mỗi vị trí trong S, bạn cần in ra vị trí của dấu ngoặc tương ứng.
- Định nghĩa:
  - Xâu rỗng là dãy ngoặc đúng
  - Nếu xâu A là dãy ngoặc đúng thì (A) cũng là dãy ngoặc đúng. Khi đó, cặp dấu ngoặc quanh xâu A này là cặp dấu ngoặc tương ứng.
  - Nếu xâu A và B đều là dãy ngoặc đúng thì xâu A + B cũng là dãy ngoặc đúng

S = (()())(())

2-1. Kiểm tra dãy ngoặc đúng

- **Tính chất:**
  - Tính chất 1: Một dãy ngoặc đúng độ dài 2*n có n cặp dấu ngoặc tương ứng, tương ứng với n dấu ngoặc mở và n dấu ngoặc đóng.
  - Tính chất 2: Trong một dãy ngoặc đúng, mỗi dấu ngoặc tương ứng với một dấu ngoặc khác duy nhất.

→ Dấu ngoặc tương ứng của một dấu ngoặc mở phải nằm ở sau nó trong dãy, và dấu ngoặc tương ứng của một dấu ngoặc đóng phải nằm trước nó.

→ Do đó, dãy ngoặc đúng không thể bắt đầu bằng dấu đóng ngoặc, hay kết thúc bằng dầu mở ngoặc.

2-1. Kiểm tra dãy ngoặc đúng

- **Tính chất:**
  - Tính chất 3: Dãy ngoặc đúng khi và chỉ khi số dấu ngoặc mở bằng số dấu ngoặc đóng, và trong mọi tiền tố của dãy, số dấu ngoặc mở >= số dấu ngoặc đóng.
- A[0], A[1], …, A[N-1]
- Các dãy số A[0], A[1], …, A[i] được gọi là một dãy tiền tố

- (()())(())
- (()())(() mở = 5, đóng = 4
- (()())((mở = 5, đóng = 3
- (()())(mở = 4, đóng = 3
- (()()) mở = 3, đóng = 3

(()() mở = 3, đóng = 2
(()(mở = 3, đóng = 1
(() mở = 2, đóng = 1
((mở = 2, đóng = 0
(mở = 1

2-1. Kiểm tra dãy ngoặc đúng

- **Tính chất dãy ngoặc bị sai:**
  - Số lượng mở khác số lượng đóng
  - Tồn tại một tiền tố của S mà số lượng mở ngoặc < đóng ngoặc.

-)(((((, tiền tố) có mở = 0, đóng = 1 (mở < đóng)
- (())((có mở = 4, đóng = 2 (vi phạm tính chất n mở, n đóng)
- ((()))(, tiền tố ((())) có mở = 2, đóng = 3 (mở < đóng)

2-1. Kiểm tra dãy ngoặc đúng

- **Lời giải:**

  Cho một Stack chứa các phần tử kiểu char, đang ở trạng thái rỗng. Xét quá trình sau:

  - Duyệt xâu S từ trái qua phải
  - (1) Nếu ký tự đang xét là dấu (, thêm vào Stack
  - (2) Nếu ký tự đang xét là dấu), xét phần tử đang ở đỉnh Stack.

    (a) Nếu Stack rỗng, xâu S không phải dãy ngoặc đúng.

    (b) Nếu đó là dấu (, ta tìm được một cặp dấu ngoặc tương ứng, và loại bỏ dấu ngoặc mở khỏi Stack.
  - Sau khi thực hiện quá trình, nếu Stack rỗng thì S là một dãy ngoặc đúng.

2-1. Kiểm tra dãy ngoặc đúng

- **Lời giải:**

  Cho một Stack chứa các phần tử kiểu char, đang ở trạng thái rỗng. Xét quá trình sau:

  - Duyệt xâu S từ trái qua phải
  - Nếu ký tự đang xét là dấu (, thêm vào Stack
  - Nếu ký tự đang xét là dấu), xét phần tử đang ở đỉnh Stack.

  (1) Nếu Stack rỗng, xâu S không phải dãy ngoặc đúng.

  (2) Nếu đó là dấu (, ta tìm được một cặp dấu ngoặc tương ứng, và loại bỏ dấu ngoặc mở khỏi Stack.

  - Sau khi thực hiện quá trình, nếu Stack rỗng thì S là một dãy ngoặc đúng.

S = (())()	Stack
i = 0, S[0] = (	(
i = 1, S[1] = (	((
i = 2, S[2] =)	Stack.pop() → (
i = 3, S[3] =)	Stack.pop() → empty
i = 4, S[4] = (	Stack.push() → (
i = 5, S[5] =)	Stack.pop() → empty

2-1. Kiểm tra dãy ngoặc đúng – ver 2

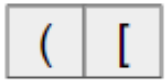
- **DSA07110**
- Mỗi ngoặc mở “(“, “[“, “{“ phải được cặp với một ngoặc đóng “)“, “]“, “}“ tương ứng.

- Ví dụ
  - Dãy ngoặc đúng: ( )(( )){([( )])}
  - không cân xứng: ((( )(( )){([( )])}
  - không cân xứng:)(( )){([( )])}
  - không cân xứng: ({[ ])}
  - không cân xứng: (

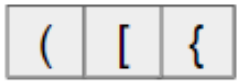
2-1. Kiểm tra dãy ngoặc đúng – ver 2

- Ví dụ: S = ([{}])() **DSA07110**

S[0] = (, đẩy vào stack → stack = (



S[1] = [, đẩy vào stack → stack = ([



S[2] = {, đẩy vào stack → stack = ([{

S[3] = }, do dấu ngoặc ở đỉnh trùng kiểu với nó, thực hiện pop stack → stack = ([

S[4] =], do dấu ngoặc ở đỉnh trùng kiểu với nó, thực hiện pop stack → stack = (

S[5] =), do dấu ngoặc ở đỉnh trùng kiểu với nó, thực hiện pop stack

S[6] = (, đẩy vào stack

S[7] =), do dấu ngoặc ở đỉnh trùng kiểu với nó, thực hiện pop stack

2-2. Dùng stack để khử đệ quy

- Vì tính chất LIFO (vào sau, ra trước) của Stack, nó có thể được sử dụng để khử các hàm đệ quy. Trên thực tế, khi ta gọi đệ quy, hệ thống sẽ tự động tạo một cấu trúc LIFO như Stack để chứa và thực hiện các lời gọi hàm.

- Ví dụ, xét thuật toán DFS sử dụng đệ quy:

```
void dfs(int start){
    visited[start] = true;
    for (int i = 0; i < adj[start].size(); i++) {
        int v = adj[start][i];
        if (visited[v])
            continue;
        dfs(v);              // đệ quy: thăm u
    }
}
```

2-3. Chuyển đổi trung tố - hậu tố

- Biểu thức trung tố ⇔ hậu tố (toán tử đặt ở phía sau toán hạng)
  - a + b ⇔ a b +
  - a - b ⇔ a b -
  - a * b ⇔ a b *
  - a / b ⇔ a b /
  - (P) ⇔ P

$$(a+b*c)-(a/b+c)=$$

$$=(a+bc*)-(ab/+c)$$

$$=(abc*+)-(ab/c+)$$

$$=abc*+-ab/c+$$

$$=abc*+ab/c+-$$

2-3. Chuyển đổi trung tố - hậu tố

Thuật toán:

- Bước 1: Khởi tạo Stack rỗng.
- Bước 2: Lặp:

Duyệt vòng lặp for từ i = 1 cho đến cuối chuỗi P:

  - Nếu P[i] là (, đẩy vào trong ngăn xếp S.
  - Nếu P[i] là toán hạng thì đưa vào Output.
  - Nếu P[i] là toán tử thì so sánh độ ưu tiên rồi push vào ngăn xếp S.
  - Nếu P[i] là “)” thì Pop vào ngăn xếp S (lấy giá trị trên đỉnh của S) sau đó đưa vào Output.

- Bước 3: Hoàn chỉnh biểu thức hậu tố

In ra nốt những phần tử còn lại ở trong Stack sang Output

2-3. Chuyển đổi trung tố - hậu tố

Thuật toán:

Bước 1 (Khởi tạo): stack = ∅; Out = ∅;
Bước 2 (Lặp): For each x∈P do
2.1. Nếu x =‘ (‘ : Push(stack, x);
2.2. Nếu x là toán hạng: x⇒Out;
*2.3. Nếu x ∈ { +, -, *, / }*
y = get(stack);
** Nếu priority(x)≥priority(y): Push(stack, x);*
** Nếu priority(x)<priority(y):*
y = Pop(stack); y⇒Out; Push(stack, x);
** Nếu stack = ∅: Push(stack, x);*
2.4. Nếu x =‘)’:
While (true) do
y = stack.top(); stack.pop();
if (y == ‘(‘) break;
y⇒Out;
EndWhile;
Bước 3(Hoàn chỉnh biểu thức hậu tố):
While (stack ≠ ∅) do
y = stack.top(); stack.pop(); y⇒Out;
EndWhile;
Bước 4(Trả lại kết quả): Return(Out).

```
int getPriority(char x) {
    if (x == '(')
        return 0;
    if (x == '+' || x == '-')
        return 1 ;
    if (x == '*' || x == '/')
        return 2;

    return 3;
}
```

Kiểm nghiệm thuật toán: P = $(a+b*c)-(a/b+c)$

x∈P	Bước	Stack	Out
x =‘(‘	2.1	(	∅
x =a	2.2	(	a
x =+	2.3.a	(+	a
x =b	2.2	(+	a b
x =*	2.3.a	(+ *	a b
x = c	2.2	(+ *	a b c
x =‘)‘	2.3	∅	a b c * +
x =-	2.2.c	-	a b c * +
x =‘(‘	2.1	- (	a b c * +
x =a	2.2	- (	a b c * + a
x =/	2.2.a	- (/	a b c * + a
x =b	2.2	- (/	a b c * + a b
x =+	2.3.b	- (+	a b c * + a b/
x =c	2.2	- (+	a b c * + a b/c
x =‘)‘	2.4	∅	a b c * + a b/c + -
P = a b c * + a b/c + -			

2-4. Tính toán biểu thức hậu tố

Thuật toán tính toán giá trị biểu thức hậu tố

- Bước 1 (Khởi tạo): stack = $\varnothing$;
- Bước 2 (Lặp) :

  For each x $\in$ P do

  - 2.1. Nếu x là toán hạng:

    Push(stack, x);
  - 2.2. Nếu x $\in$ { +, -, *, / }

    a) TH2 = Pop(stack, x);

    b) TH1 = Pop(stack, x);

    c) KQ = TH1 $\otimes$ TH2;

    d) Push (stack, KQ);

  EndFor;
- Bước 4(Trả lại kết quả):

  Return(Pop(stack)).

2-4. Tính toán biểu thức hậu tố

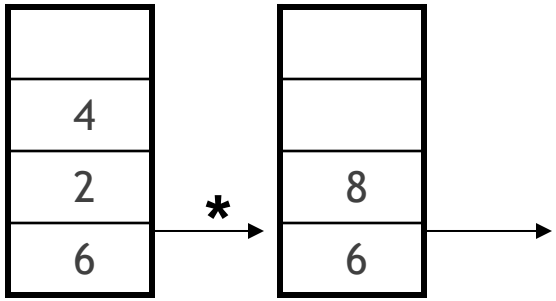
Thuật toán tính toán giá trị biểu thức hậu tố

- Bài toán 1: Cho biểu thức ví dụ: 2*9-7
  - Bước 1: Chuyển biểu thức trung tố → hậu tố
  - Bước 2: Tính giá trị biểu thức hậu tố
  - Stack sử dụng dạng stack<int> or stack<char>

- Bài toán 2: Cho biểu thức ví dụ: 20*10 - 100
  - Bước 1: vector<string> output
  - Bước 2: stack<string> st
  - Chuyển đổi string sang số nguyên để tính toán

2-4. Tính toán biểu thức hậu tố

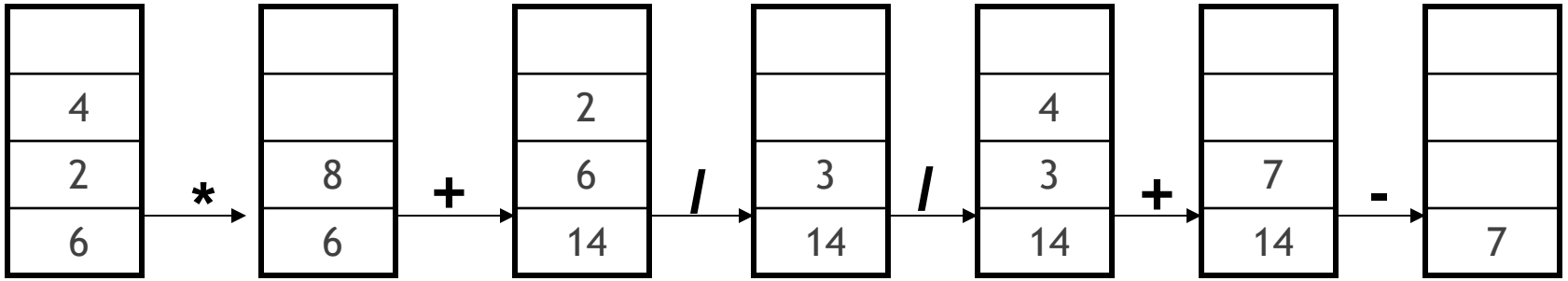
- **Ví dụ:** P = 6 2 4 * + 6 2 / 4 + -



6 2 4 *

2-4. Tính toán biểu thức hậu tố

- **Ví dụ:** P = 6 2 4 * + 6 2 / 4 + -



2-5. Phần tử bên phải đầu tiên lớn hơn A[i]

- DSA07027
- Cho dãy số A[] gồm N phần tử. Với mỗi A[i], bạn cần tìm phần tử bên phải đầu tiên lớn hơn nó. Nếu không tồn tại, in ra -1.

- VÍ DỤ:
  - Dãy số: A[] = {4, 5, 2, 25}
  - Kết quả: R[] = {5, 25, 25, -1}

- Giải thuật:
  - Cách làm thông thường: độ phức tạp O(n^2)
  - Sử dụng Stack: độ phức tạp O(n)

2-5. Phần tử bên phải đầu tiên lớn hơn A[i]

- Cho dãy số A[] gồm N phần tử. Với mỗi A[i], bạn cần tìm phần tử bên phải đầu tiên lớn hơn nó. Nếu không tồn tại, in ra -1.

i	0	1	2	3
A[i]	4	5	2	25
R[i]				
Stack				

```
void process(int a[], int n){
    stack<int> st;
    int R[n];
    for (int i = n-1; i >= 0; i--){
        while(!st.empty() && a[i] >= st.top())
            st.pop();
        if(st.empty())
            R[i] = -1;
        else
            R[i] = st.top();
        st.push(a[i]);
    }
    for(int i = 0; i < n; i++) cout << R[i] << " ";
    cout << endl;
}
```

2-5. Phần tử bên phải đầu tiên lớn hơn A[i]

- Cho dãy số A[] gồm N phần tử. Với mỗi A[i], bạn cần tìm phần tử bên phải đầu tiên lớn hơn nó. Nếu không tồn tại, in ra -1.

i	0	1	2	3
A[i]	4	5	2	25
R[i]	5	25	25	-1
Stack	25 5	25 5	25 2	25

i = 3, R[3] = -1
i = 2, A[2] < st.top = 25
→ R[2] = 25
i = 1, A[1] >= st.top = 2
→ st.pop
→ R[1] = st.top = 25
i = 0, A[0] < st.top
→ R[1] = st.top = 5

```
void process(int a[], int n){
    stack<int> st;
    int R[n];
    for (int i = n-1; i >= 0; i--){
        while(!st.empty() && a[i] >= st.top())
            st.pop();
        if(st.empty())
            R[i] = -1;
        else
            R[i] = st.top();
        st.push(a[i]);
    }
    for(int i = 0; i < n; i++) cout << R[i] << " ";
    cout << endl;
}
```

Một số bài toán khác

- Chuyển đổi giữa các dạng biểu thức:
  - Trung tố
  - Hậu tố
  - Tiền tố
- Kiểm tra dư thừa dấu ngoặc
- Tính độ dài dãy ngoặc đúng dài nhất
- Đếm số dấu ngoặc cần đổi chiều
- Tính diện tích hình chữ nhật lớn nhất

QUESTIONS & ANSWERS