Welcome to my class

Presented by…..
Shajeda Yeasmin
Instructor(Computer)
Bangladesh Sweden Polytechnic Institute.
Kaptai. Rangmati.

Name of the Subject & code

Data Structure & Algorithm

Subject Code : 28542

অধ্যায়-৪

(STACK)

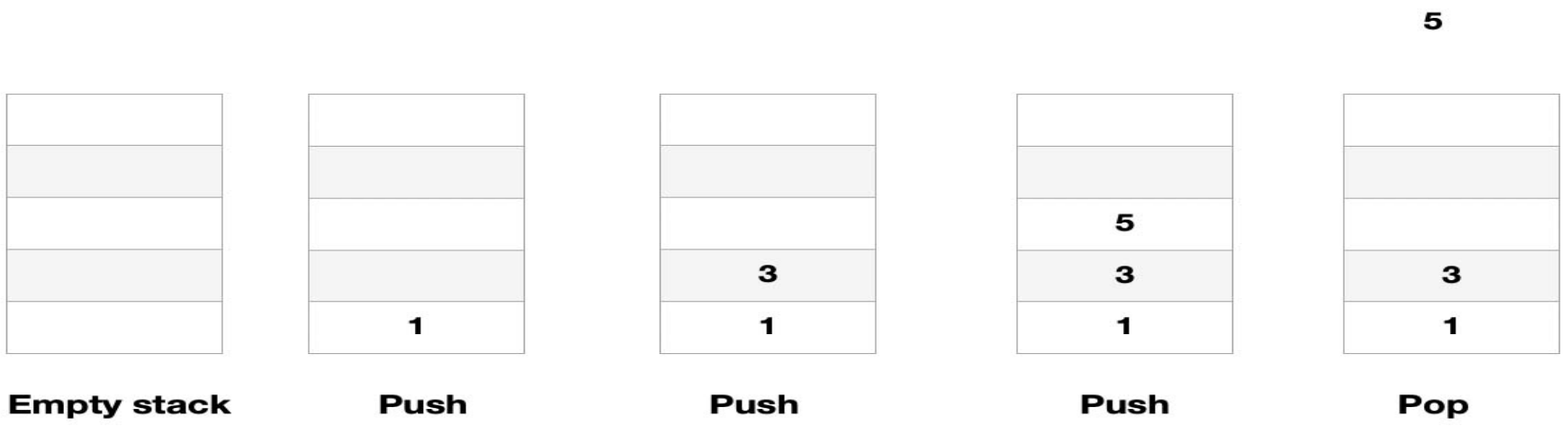
স্ট্যাক কিঃ

স্ট্যাক / Stack হচ্ছে লিনিয়ার ডেটা স্ট্র্যাকচার। স্ট্যাক কিছুটা অ্যারের মতই। তবে এখানে LIFO স্ট্র্যাকচারে ডেটা গুলো রাখা হয়। LIFO এর পূর্ণরুপ হচ্ছে Last In First Out। যেমন ধরি প্লেটের স্ট্যাক। রান্না ঘরে সাধারণত প্লেট গুলো পরিষ্কার করার পর একটার উপর আরেকটা রাখা হয়। প্লেটের স্ট্যাকে যে প্লেটটা সবার শেষে রাখে, তাই তো সবার আগে নেওয়া হয়, তাই না? যেটা সবার আগে রাখা হয়, ঐটা সবার শেষে নিতে হয়। আরেকটা উদাহরণ হচ্ছে প্রিংগেলস বা লেইস এর স্ট্যাক চিপস গুলো। এগুলোর মধ্যে যে চিপটা সবার শেষে ঢুকানো হয়, সেই চিপচটাই আমরা সবার আগে বের করি।

স্ট্যাকের ব্যাসিক অপারেশন গুলোঃ

Push: স্ট্যাকে নতুন আইটেম যুক্ত করা (সর্বশেষ আইটেমের উপরে)। – Time Complexity: O(1)
Pop: স্ট্যাকের সবার উপরের আইটেমটি বের করা। – Time Complexity: O(1)
IsEmpty: স্ট্যাকটি খালি কিনা, তা চেক করা। – Time Complexity: O(1)
Size: স্ট্যাকের সাইজ বের করা। – Time Complexity: O(1)
Peek: স্ট্যাকের সর্বশেষ আইটেম বা উপরের আইটেমটি দেখা। – Time Complexity: O(1)

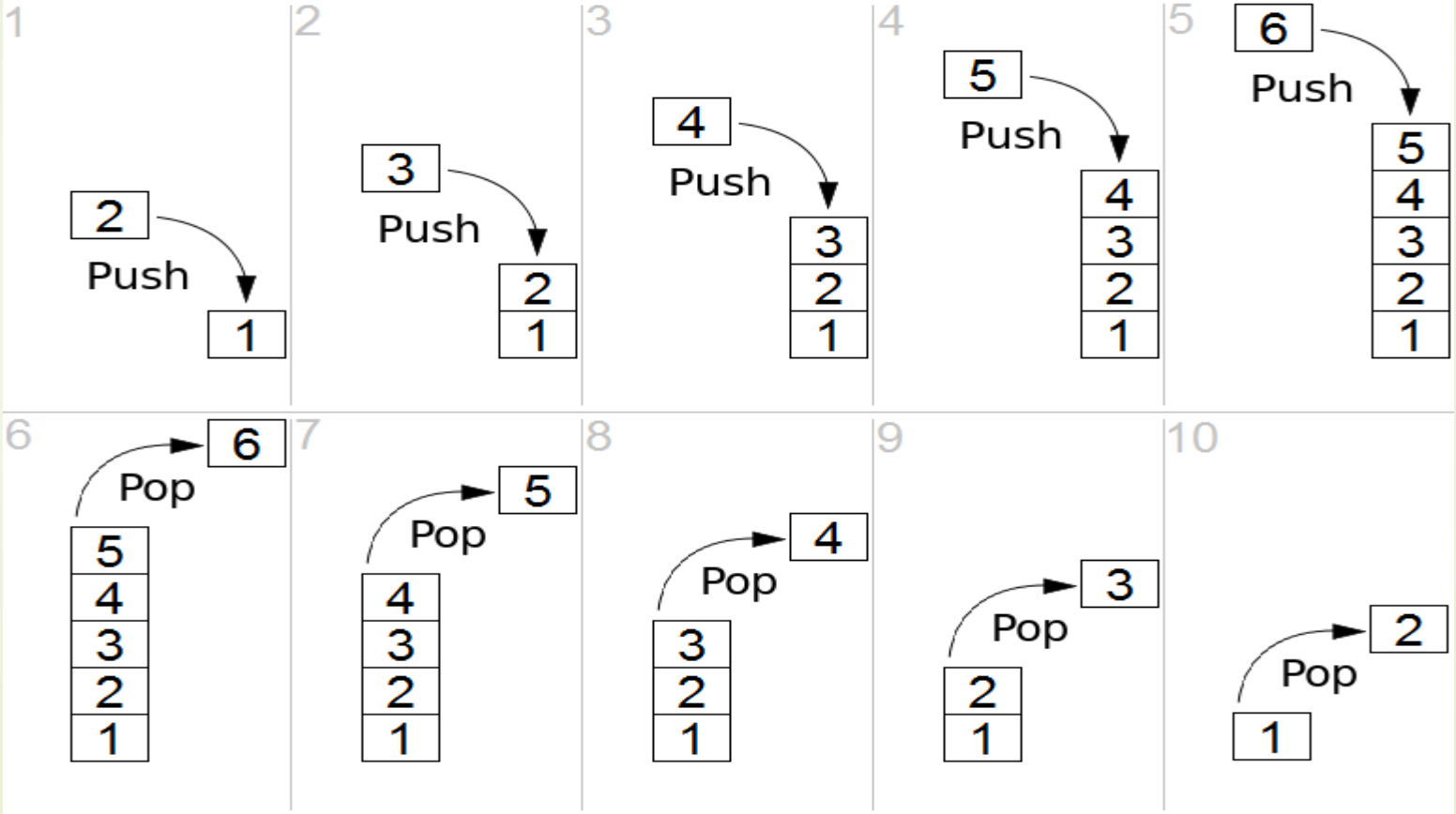
স্ট্যাকের বিভিন্ন অপারেশনের টাইম কমপ্লেক্সিটি হচ্ছে O(1)।



State PUSH and POP

Push(new_element): স্ট্যাকের উপরে নতুন বস্তুটা রাখো। যদি শুরুতে স্ট্যাকটা খালি হয়, তাহলে প্রথম জায়গায় বস্তুটা রাখতে হবে।

Pop(): স্ট্যাকের উপরের বস্তুটা সরিয়ে ফেলো। যদি শুরুতে স্ট্যাকটা খালি হয় তাহলে এই অপারেশনটা করা সম্ভব না।



Insertion: push()

push() is an operation that inserts elements into the stack. The following is an algorithm that describes the push() operation in a simpler way.

Algorithm

1 – Checks if the stack is full.
2 – If the stack is full, produces an error and exit.
3 – If the stack is not full, increments top to point next empty space.
4 – Adds data element to the stack location, where top is pointing.
5 – Returns success.

Deletion: pop()

pop() is a data manipulation operation which removes elements from the stack. The following pseudo code describes the pop() operation in a simpler way.

Algorithm

1 – Checks if the stack is empty.
2 – If the stack is empty, produces an error and exit.
3 – If the stack is not empty, accesses the data element at which top is pointing.
4 – Decreases the value of top by 1.
5 – Returns success.

Explain the concept of Infix, Postfix & Prefix expression.

Infix notation: আমরা সাধারণত যেভাবে কোন একটা equation লিখি অর্থাৎ একটা arithmetic operator এর দুই পাশে দুটি operand থাকে। অপারেটরটা ঐ অপারেন্ড দুটির উপর কাজ করে। এই প্রকাশ পদ্ধতিই হচ্ছে infix expression. যেমনঃ A + B. A ও B হচ্ছে operand এবং + হচ্ছে অপারেটর। অপারেটরটি অপারেন্ড দুটির মাঝে বসেছে।

Polish notation: এই পদ্ধতিতে অপারেটর বসে অপারেন্ড দুটির শুরুতে। যেমনঃ +AB বলতে বুঝাচ্ছে A আর B-কে যোগ করতে হবে।

Postfix notation: এক্ষেত্রে অপারেটর বসে অপারেন্ড দুটির পরে। একে Reverse Polish Notation-ও বলা হয়। যেমনঃ AB+ বলতে Aএবং B এর মাঝের যোগের সম্পর্কই বুঝাচ্ছে।

Infix to Postfix conversion

ধরো, ইনফিক্সে একটা এক্সপ্রেশন দেয়া আছে **(A+(B^C)*D)/C**. এটাকে পোস্টফিক্স নোটেশনে কনভার্ট করতে হবে। এতে সুবিধা হচ্ছে কোন ব্র্যাকেট থাকবে না এক্সপ্রেশনটিতে। আর অপারেটরগুলো এমন ভাবে সাজানো থাকবে যেন তাদের precedence অনুযায়ীই অপারেশনগুলো হয়। চলো কনভার্ট করিঃ

(A+(B^C)*D)/C

= (A+[BC^]*D)/C {exponent এর precedence সবচেয়ে বেশি তাই এটার কাজ আগে হয়েছে}

= (A+[BC^D*])/C {BC^ এর সাথে D এর গুণের সম্পর্ক। তাই এটার priority বেশি}

= ([ABC^D*+])/C {[BC^D*] এর সাথে A এর যোগের কাজের প্রাধান্য বেশি কারণ এরা () এর ভিতরে}

= ABC^D*+C/ {[ABC^D*+] কে ভাগ করা হয়েছে C দিয়ে}

অবশেষে ইনফিক্স নোটেশন **(A+(B^C)*D)/C** এর পোস্টফিক্স ফরমেট পাওয়া গেল **ABC^D*+C/**. একই ভাবে তুমি Infix to Polish notation এ রূপান্তর করতে পারবে। পার্থক্য শুধু "অপারেটরগুলো পরে না লিখে আগে লিখবে"।

Evaluation of Postfix notation

1 2 3 ^ 4 * + 3 / এর বাম দিক থেকে পড়া শুরু করি। 1, 2, 3 এর পরে পাওয়া গেল ^. যেহেতু পোস্টফিক্স তার মানে এই অপারেটর কাজ করবে তার আগে থাকা দুটি অপারেন্ডের উপর। অর্থাৎ 2^3 = 8 এর কাজটা প্রথমে হবে। এই রেজাল্টটা বসিয়ে দিব 2 3 ^ এর স্থলে। তাহলে নতুন এক্সপ্রেশনটা দাঁড়ায় 1 8 4 * + 3 /. এরপর যথারীতি ডান দিকে এগিয়ে যেতে থাকি। পাওয়া গেল 4. আর এক ঘর সামনে গেলে পাওয়া গেল * চিহ্ন। তাহলে এই গুণ চিহ্ন কোন দুইটা সংখ্যার উপর কাজ করবে? ঠিক ধরেছ! 8 * 4 = 32 এর হিসাব হবে এখন। 32 কে আগের নিয়মে বসিয়ে দেই 8 4 * এর স্থলে। নতুন এক্সপ্রেশন দাঁড়ায়ঃ 1 32 + 3 /. আরেক ঘর ডানে গিয়ে পাওয়া গেল + চিহ্ন। 1 + 32 = 33. আপডেটেড এক্সপ্রেশনটা এখনঃ 33 3 /. এবং ফাইনালি কাজ হবে 33 / 3 = 11

অর্থাৎ 1 2 3 ^ 4 * + 3 / = 11

Line by line execution:

Given, 1 2 3 ^ 4 * + 3 / as a postfix notation.

1 2 3 ^ 4 * + 3 /

= 1 **8** 4 * + 3 /

= 1 **32** + 3 /

= **33** 3 /

= **11**

ধন্যবাদ