ডাটা স্ট্রাকচার - Data structures   লিনিয়ার ডাটা স্ট্রাকচার

ডাটা স্ট্রাকচার: স্ট্যাক এবং কিউ **(Stack and Queue)**

Stack Bangla tutorial/ Queue Bangla tutorial/ Stack and Queue in the C++ STL

Sharif Hasan • January 4, 2022 সর্বশেষ আপডেট January 7, 2022 2 5,684 পড়তে 6 মিনিট লাগতে পারে

স্ট্যাক এবং কিউ (Stack and Queue) বহুল ব্যবহৃত ডাটা স্ট্রাকচার (Data structure) গুলোর মধ্যে অন্যতম। যখন এমন কোন সিচুয়েশন আসে যেখানে আমাদেরকে ডাটার পরিমাণ নির্দিষ্ট করা হয় না, আবার ডেটা রাখা এবং তুলে আনার অপারেশন $O(1)$ এ করতে হয় তখন আমরা স্ট্যাক এবং কিউ ব্যবহার করি। স্ট্যাক এবং কিউ দুইটি আলাদা ডাটা স্ট্রাকচার হলেও আমি একই লিখায় দুইটি নিয়েই আলোচনা করবো। কারণ স্ট্যাক (Stack) আর কিউ (queue) এর ব্যাসিক প্রায় একই। একটি পারলে আরেকটি সহজেই বুঝা যায়।

এই লিখাটি দেখার আগে লিঙ্কড লিস্ট নিয়ে ধারনা থাকা জরুরি। এই লিঙ্কে যেয়ে পড়ে আসতে পারেন।

স্ট্যাক ডাটা স্ট্রাকচারের সংজ্ঞা- **Stack definition**

Stack একটি লিনিয়ার ডেটা স্ট্রাকচার যেখানে LIFO ক্রম অনুসারে ডেটা গুলোর উপর অপারেশন করা হয়।

স্ট্যাক কি -What is stack? স্ট্যাক বুঝার জন্য আমরা একটা উদাহরণ দিতে পারি। ধরা যাক আপনি একটি টেবিলে বই একটার উপর আরেকটি রেখেছেন। এখন আপনাকে সবার নিচের বইটি বের করতে হবে। এর জন্য কিন্তু আপনি সবার নিচের বই শুরুতেই বের করতে পারবেন না। কারণ বই এর স্তূপ বড় হলে উপরের বই পরে গিয়ে এলোমেলো হয়ে যাবে।

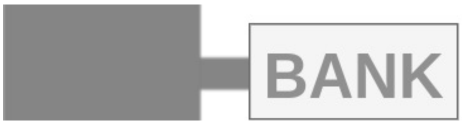
এর জন্য যা করতে হবে, শুরুতে সবার উপরের বই তুলতে হবে। তারপর উপর থেকে দ্বিতীয় বইটি তুলতে হবে। এভাবে সবার শেষে পৌছাতে হবে। এখন যদি লক্ষ করেন তবে দেখুন, সবার উপরের বইটি কিন্তু আপনি সবার শেষে রেখেছেলেন। উপর থেকে দ্বিতীয় বইটি আপনি শেষ ধাপের আগের ধাপে রেখেছিলেন। এভাবে সবার নিচের বইটি আপনি শুরুতে রেখেছিলেন।

তারমানে আমরা বুঝতে পারছি, যে ডেটা স্ট্রাকচারে সবার শেষে এন্ট্রি করা ডেটাকে সবার আগে ডিলিট করা সম্ভব তাই স্ট্যাক। তাই এই ডেটা স্ট্রাকচারকে LIFO (Last In First Out) ডেটা স্ট্রাকচারও বলা হয়।

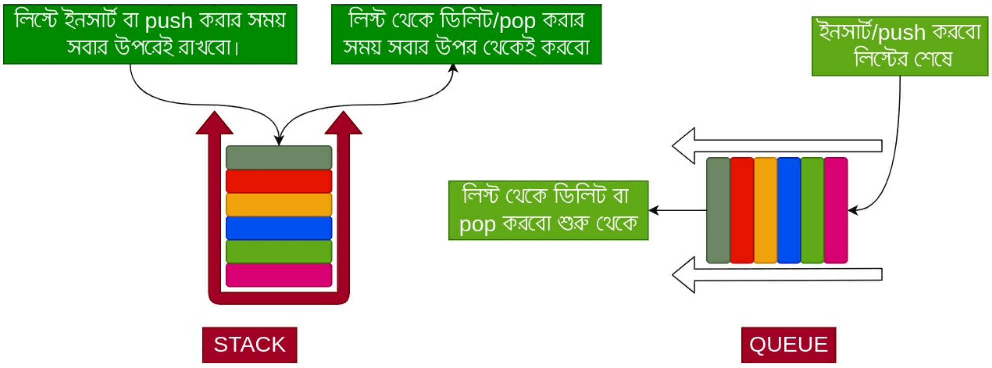
কিউ ডেটা স্ট্রাকচারের সংজ্ঞা-**Queue definition**

Queue একটি লিনিয়ার ডেটা স্ট্রাকচার যেখানে FIFO ক্রম অনুসারে ডেটা গুলোর উপর অপারেশন করা হয়।

এখন আরেকটি কেস বিবেচনা করি চলেন। ধরেন আপনি ব্যাংকে টাকা জমা দিবেন তাই আপনি লাইনে দাঁড়িয়েছেন। লাইনে আপনার পজিশন ধরা যাক ৬। তারমানে আপনার সামনে আরও ৫ জন আপনার আগে এসেছে। তাই তারা আগে জমা দিয়ে আগে চলে যাবে। আপনি সবার শেষে এসেছেন তাই আপনি সবশেষে টাকা জমা দিবেন।



অর্থাৎ যেসব ডেটা স্ট্রাকচারে সবার প্রথমে রাখা ডেটাকে আগে ডিলিট করা যায় তাকে কিউ ডেটা স্ট্রাকচার বলে। কিউ কে আমরা FIFO (First In First Out) নামেও ডাকি।



উপরের চিত্রটি দেখি, এখানে লিস্ট বলার কারণ হলো স্ট্যাক এবং কিউতে শুরুতেই সাইজ বলা থাকে না বিধায় আমরা অ্যারের মত ফিক্সড সাইজ ডেটা স্ট্রাকচার নিয়ে কাজ করতে পারি না। তাই লিঙ্কড লিস্ট ব্যবহার করতে হবে। নিচের টেবিলে স্ট্যাক এবং লিঙ্কড লিস্টের পার্থক্য গুলো দিয়েছি।

স্ট্যাক এবং কিউ এর অপারেশনগুলো – **Operations in stack and queue**

1. **push** অপারেশন: স্ট্যাক বা কিউ তে কোন ডেটা রাখার অপারেশনকে push অপারেশন বলে। কিউ এর ক্ষেত্রে অনেক সময় এই অপারেশনকে enqueue অপারেশন বলা হয়।
2. **pop** অপারেশন: স্ট্যাক বা কিউ থেকে ডেটা ডিলিট করার অপারেশনকে pop অপারেশন বলে। কিউ এর ক্ষেত্রে এই অপারেশনকে dequeue ও বলা হয়।
3. **top** অপারেশন: শুধু স্ট্যাক এর জন্য এই অপারেশনের মাধ্যমে পরীক্ষা করা হয় স্ট্যাকের সবার উপরে কোন ইলিমেন্ট আছে।
4. **front** অপারেশন: শুধু কিউ এর এই অপারেশনের মাধ্যমে দেখা হয় কিউ এর সবার সামনে কোন ইলিমেন্ট আছে।

স্ট্যাক **(STACK)**	কিউ **(QUEUE)**
স্ট্যাক LIFO এর নিয়ম অনুসারে কাজ করে। মানে যে ইলিমেন্ট সবার শেষে ঢুকানো হবে	কিউ FIFO অনুযায়ী কাজ করে। সবার শুরুতে যেই ইলিমেন্ট ইনসার্টবা পুশ করা হবে তাকে সবার আগেই

স্ট্যাক **(STACK)**	কিউ **(QUEUE)**
তাকে সবার আগে বের করা হবে।	বের করা হবে
ইনসার্ট বা ডেটা push করা এবং ডিলিট বা ডেটা pop করা সবসময় লিঙ্কড লিস্টের একই পাশ থেকে হয়। একে আমরা top বলি।	কিউ তে push করা হয় পেছন বা tail থেকে এবং pop করা হয় সামনে থেকে বা front থেকে।
স্ট্যাক ইমপ্লিমেন্ট করার জন্য আমাদের একটি পয়েন্টারের মাধ্যমে লিস্টের আইটেম push/ pop করতে হবে (top/ head pointer)।	কিউতে push/ pop এর জন্য আমাদেরকে দুইটি পয়েন্টার মেইন্টেইন করা লাগবে। একটির মাধ্যমে push করবো (tail pointer)। আরেকটির মাধ্যমে pop করবো (front/head pointer)।
রিকার্শনের মাধ্যমে সমস্যা সমাধানের ক্ষেত্রে স্ট্যাক ব্যবহার করা হয়।	ইটারেটিভ প্রসেসে সমস্যা সমাধানের জন্য সাধারণত কিউ ব্যবহার করা হয়।

Linked list ব্যবহার করে Stack ইমপ্লিমেন্টেশন – Stack Bangla tutorial

লিঙ্কড লিস্ট ব্যবহার করে স্ট্যাক ইমপ্লিমেন্ট করা সোজা।

লিঙ্কড লিস্ট ব্যবহার করে স্ট্যাক ইমপ্লিমেন্ট করা সহজ। আমাদেরকে প্রতিটি pop() অপারেশনের সময় প্রথম নোডটি ডিলিট করে ফেলতে হবে। অন্যান্য অপারেশন যেমন insert() আগের মতই থাকবে। স্ট্যাক এর সাথে মিল রাখার জন্য insert ফাংশনকে আমরা ডাকবো push নামে এবং delete ফাংশনকে আমরা ডাকবো pop নামে। head পয়েন্টার NULL কিনা তা পরীক্ষা করতে আমরা empty() নামে একটি ফাংশন যোগ করবো এবং সবার উপরের ইলিমেন্ট দেখার জন্য একটি ফাংশন তৈরি করবো তার নাম হবে top()। নীচের কোড দেখুন.

```
struct Node{
 int data;
 Node *next;
};
Node *head=NULL;
```

এই অংশটুকু লিঙ্কড লিস্টের মতই রয়েছে। Node একটি স্ট্রাকচার যেখানে দুইটি ফিল্ড data এবং *next রয়েছে। *head আমাদের লিস্টের শুরু বা স্ট্যাকের top কে রিপ্রেজেন্ট করে।

লিস্টে পুশ করার সময় নিয়ম অনুসারে আমাদেরকে সবার শুরুতে বা head এ পুশ করতে হবে। নিচে পুশ ফাংশনটি দেখুন,

```
void push(int a){
 Node *new_node=new Node;
 new_node->data=a;
 new_node->next=head;
 head=new_node;
}
```

সবার প্রথমে new_node নামে একটি নোড তৈরি করেছি। তারপর new_node এর data ফিল্ডে a বসিয়েছি। এখন যেহেতু push/ insert সবার সামনে হচ্ছে তাই শুরুতে যে head ছিল সে এবার দ্বিতীয় তে যাবে এবং new_node আমাদের নতুন head হবে। এর জন্যই new_node->next=head; এবং head=new_node; করেছি।

এখানে লিস্ট খালি নাকি তা চেক করা লাগবে না। কারণ যেহেতু আমরা head এর কোন ফিল্ড চেক করছি না, তাই head==NULL হলেও রানটাইম এরর হবার সম্ভাবনা নেই।

```
void pop(){
 if(head==NULL){
 cout<<"Stack underflow\n";
 }else{
 Node *temp=head;
 head=head->next;
 delete []temp;
 }
}
```

pop() ফাংশনের ক্ষেত্রে আমরা লিঙ্কড লিস্টের শুরুর নোড ডিলিট করার পদ্ধতি মেনেছি। প্রথমেই দেখে নিতে হবে লিস্ট খালি কিনা, যদি লিস্ট খালি না হয় তবে, head->next বা head এর পরের নোড কে নতুন head নোড বানাবো এবং আগের head নোড ডিলিট করে দিবো।

```
int top(){
 if(head==NULL){
 cout<<"Stack is empty\n";
 return -1;
 }else{
 return head->data;
 }
}
```

top() ফাংশনে শুধু দেখেছি head এর ডাটা ফিল্ডে কি ভ্যালু আছে।

Full stack implementation using C++

```
#include<bits/stdc++.h>
using namespace std;

struct Node{
 int data;
 Node *next;
};
Node *head=NULL;

void push(int a){
 Node *new_node=new Node;
 new_node->data=a;
 new_node->next=head;
 head=new_node;
}

void pop(){
 if(head==NULL){
 cout<<"Stack underflow\n";
 }else{
 Node *temp=head;
 head=head->next;
 delete []temp;
 }
}

int top(){
 if(head==NULL){
 cout<<"Stack is empty\n";
 return -1;
 }else{
 return head->data;
 }
}


int main(){
 push(4);
 push(7);
 push(9);

 cout<<"The top is "<<top()<<endl;
 pop();

 cout<<"The top is "<<top()<<endl;
 pop();

 cout<<"The top is "<<top()<<endl;
}
```

Linked list ব্যবহার করে Queue ইমপ্লিমেন্টেশন-Queue Bangla tutorial

লিঙ্কড লিস্ট ব্যবহার করে Queue ইমপ্লিমেন্ট করার জন্য আমাদেরকে শুধু একটি head পয়েন্টারের বদল দুইটি পয়েন্টার ব্যবহার করা লাগবে। অপর পয়েন্টারটি হলো tail নামে। এর মাধ্যমে আমরা সবার শেষে যেই নোড আছে তার ট্র্যাক রাখবো। প্রতিবার ইনসার্ট এর সময় শেষের নোডটি tail পয়েন্টারে এসাইন করে দিতে হবে। আর queue

এর কনভেনশন অনুযায়ী insert কে আমরা enqueue বলবো, delete কে dequeue বলবো এবং queue খালি কি না তা দেখার জন্য একটি empty নামের ফাংশন এবং queue এর প্রথম ইলিমেন্ট দেখার জন্য একটি front নামের ফাংশন তৈরি করবো।

```
1  struct Node{
2   int data;
3   Node *next;
4  };
5  Node *head=NULL;
6  Node *tail=NULL;
```

যেহেতু queue তে আমাদেরকে $O(1)$ টাইমে push/ insert/ enqueue করতে হয়, তাই আমরা সাধারণভাবে লুপের মাধ্যমে শেষ নোড বের করে ইনসার্ট করতে পারবো না। তাই আমাদের যা করতে হবে আরেকটি পয়েন্টার নিতে হবে tail নামে। এই পয়েন্টার এর মাধ্যমে আমরা শেষ নোড কোনটি তার খোজ রাখবো। ৬ নং লাইনে এই পয়েন্টারটি নেয়া হয়েছে।

```
void enqueue(int a){
 Node *new_node=new Node;
 new_node->data=a;
 new_node->next=NULL;
 if(head==NULL){
 head=tail=new_node;
 }else{
 tail->next=new_node;
 tail=new_node;
 }
}
```

উপরের enqueue() ফাংশনে আমরা শুরুতেই new_node নামে একটি নোড তৈরি করে নিয়েছি যেখানে data=a সেট করে নিয়েছি। একইসাথে next=NULL করে নিয়েছি কারণ এই নোডটি লিঙ্কড লিস্টের শেষ নোড হবে।

এখন head যদি NULL হয় তবে বলা যায় লিস্ট বা কিউ খালি আছে। তখন head এবং tail একই কথা হবে। তাই head=tail=new_node করে দিয়েছি। যদি তা না হয় তবে আমরা বলতে পারি অবশ্যই লিস্টের শেষ রয়েছে যার জন্য tail==NULL হবে না। তাই পুর্বের tail এর next হিসেবে new_node কে সেট করে দিয়েছি এবং তারপর নতুন tail হিসেবে new_node কে এসাইন করে দিয়েছি।

```
void dequeue(){
 if(head==NULL){
 cout<<"queue underflow\n";
 }else{
 Node *temp=head;
 head=head->next;
 delete []temp;
 }
}

int front(){
```

```
 if(head==NULL){
 cout<<"queue is empty\n";
 return -1;
 }else{
 return head->data;
 }
}
```

Queue এর dequeue() এবং front() ফাংশনের কাজ করার পদ্ধতি Stack এর মতই। কারণ Stack এ head পয়েন্টার top কে রিপ্রেজেন্ট করে। তাই head এ লেটেস্ট নোডগুলো push করা হয় এবং সেখান থেকেই pop() বা top() অপারেশন করা হয়।

একই ভাবে Queue তেও head পয়েন্টার front কে রিপ্রেজেন্ট করে। সবার আগের নোড গুলো front এর দিকে থাকে। তাই dequeue() বা front() অপারেশন head থেকে করতে হয়। এর জন্য উভয় ক্ষেত্রেই এই ধরনের ফাংশন গুলোর কাজ করার পদ্ধতি একই।

Full queue implementation using C++

```
#include<bits/stdc++.h>
using namespace std;

struct Node{
 int data;
 Node *next;
};
Node *head=NULL;
Node *tail=NULL;
void enqueue(int a){
 Node *new_node=new Node;
 new_node->data=a;
 new_node->next=NULL;
 if(head==NULL){
 head=tail=new_node;
 }else{
 tail->next=new_node;
 tail=new_node;
 }
}

void dequeue(){
 if(head==NULL){
 cout<<"queue underflow\n";
 }else{
 Node *temp=head;
 head=head->next;
 delete []temp;
 }
}

int front(){
 if(head==NULL){
 cout<<"queue is empty\n";
 return -1;
 }else{
 return head->data;
 }
}

```

```

int main(){
 enqueue(4);
 enqueue(7);
 cout<<"At this moment front is "<<front()<<endl;
 dequeue(); //will remove 4

 cout<<"At this moment front is "<<front()<<endl;
 dequeue();// will remove 7

 enqueue(10);

 cout<<"At this moment front is "<<front()<<endl;
}
```

C++ STL Stack and Queue Bangla tutorial

প্রায় সকল প্রোগ্রামিং ল্যাঙ্গুয়েজেই stack এবং queue এর লাইব্রেরি রয়েছে এই দুটি ডেটা স্ট্রাকচারের দিয়ে কাজ করা সহজ করার জন্য। C++ এর STL এর অংশ হিসেবে রয়েছে stack এবং queue. Stack এবং Queue যথাক্রমে “stack” এবং “queue” হেডার ফাইলের অংশ।

```
#include<bits/stdc++.h>

using namespace std;

int main(){
 stack <int> stk;// declaring a stack, with data of type int
 stk.push(9);
 stk.push(10);
 cout<<"Stack has "<<stk.top()<<" on the top\n";

 queue <int> que;// declaring a queue, with data of type int
 que.push(8);
 que.push(3);
 cout<<"Queue has "<<que.front()<<" at the front\n";
}
```

আমরা যদি একটি stack তৈরি করতে চাই যার ডেটা ফিল্ডে int টাইপ ডেটা রাখবো তাহলে আমাদেরকে stack <int> stk; এভাবে নিতে হবে। অর্থাৎ সাধারণ ভাবে নিচের মতো করে আমাদেরকে stack বানাতে হবে।

```
stack <data_type> variable_name;
```

c++, STL stack এ বিভিন্ন মেথডের মধ্যে ৩ টি গুরুত্বপূর্ণ মেথড হলো,

1. top(): সবার উপরের ইলিমেন্ট দেখার জন্য ।
2. pop(): সবার উপরের ইলিমেন্ট ডিলিট করার জন্য ।
3. push(): নতুন ইলিমেন্ট ইনসার্ট করার জন্য।

একই ভাবে আমরা যদি একটি queue নিতে চাই তবে নিচের মত করে নিতে হবে,

```
queue <data_type> variable_name;
```

stack এর মতই এখানে pop() এবং push() ফাংশন বিদ্যমান। নিচে এদের কাজ এর সংক্ষিপ্ত বিবরণ দিলাম।

- front(): সবার সামনের ইলিমেন্ট দেখার জন্য।
- pop(): সবার সামনের ইলিমেন্ট ডিলিট করার জন্য।
- push(): নতুন ইলিমেন্ট ইনসার্ট করানোর জন্য।

source: Funvizeo

আজকে এই পর্যন্তই। Stack আর কিভাবে কাজ করে বুঝার জন্য এই লিঙ্কে এবং Queue কিভাবে কাজ করে তা দেখার জন্য এই লিঙ্কে দেখতে পারেন। লিখাটি কেমন লাগলো নিচে রেটিং দিয়ে আমাকে বুঝতে সহায়তা করুন এবং কোন ভুল বা সাজেশন

জানাতে কমেন্ট করুন। দেখা হবে আরেকটি লিখাতে। সে পর্যন্ত #happy_coding.

লেখাটি কেমন লেগেছে আপনার?

রেটিং দিতে হার্টের উপর ক্লিক করুন।

❤ ❤ ❤ ❤

গড় রেটিং 4.3 / 5. মোট ভোট: 48

#queue **#stack** **#ডাটা স্ট্রাকচার** **#লিঙ্কড লিস্ট**

2 টি মন্তব্য

gUeSS

January 5, 2022 at 11:29 PM

So helpful vaiya. ❤

Reply

Sharif Hasan

January 6, 2022 at 11:20 AM

ধন্যবাদ, সব সময় পাশে থাকবেন ❤

Reply