ডাটা স্ট্রাকচার - Data structures
লিনিয়ার ডাটা স্ট্রাকচার

ডাটা স্ট্রাকচার: লিঙ্কড লিস্ট (Linked list) টিউটোরিয়াল

Linked list Bangla tutorial/ Linked list Bengali Tutorial

Sharif Hasan July 26, 2021 সর্বশেষ আপডেট March 11, 2022 8 6,779 পড়তে 6 মিনিট লাগতে পারে

লিঙ্কড লিস্ট (linked list) হলো একটি ডাটা স্ট্রাকচার যেখানে ডাটা গুলোকে একটার পরে আরেকটা, লিঙ্ক আকারে রাখা হয়। ডাটা রাখার জন্য নোড তৈরি করা হয় একাধিক ফিল্ডের সমন্বয়ে। একটা নোড থেকে পরবর্তী নোডে যাওয়ার জন্য লিঙ্ক থাকে। আমরা যদি লিঙ্কড লিস্ট (linked list) তৈরি করি তবে এর চিত্রটা অনেকটা নিচের মতো দেখা যাবে।

এখানে ১,২ ও ৩ প্রত্যেকে একেকটি নোড যেখানে ২টি ফিল্ড আছে। একটি হলো ডাটা (data) যেখানে আমরা নোড ভ্যালু রাখবো এবং দ্বিতীয়টি হলো next যা মূলত একটি পয়েন্টার, এখানে আমরা পরবর্তী নোডের এড্রেস জমা করে রাখবো।

Data structure: linked list Bangla tutorial

প্রথমের আমরা আসি, লিঙ্কড লিস্ট (linked list) এর প্রতিটি নোড আমরা কিভাবে গঠন করবো? আমরা যারা C/ C++ জানি তারা স্ট্রাকচারের (struct) নাম শুনেছি নিশ্চয়। স্ট্রাকচার হলো C/ C++ এর একটি ব্যাবস্থা যেখানে আমরা একাধিক ডাটাটাইপের ভারিয়েবল একসাথে করে নতুন একটি ডাটাটাইপ গঠন করতে পারি।

নিচে দেখি নেই নোডের গঠনটি কেমন হবে। আমরা একটি নোড তৈরি করবো যেখানে ডাটা রাখার জন্য একটি ফিল্ড থাকবে এবং এই নোডের পরবর্তী নোডের এড্রেস রাখার জন্য একটি পয়েন্টার থাকবে। নিচের কোডটি খেয়াল করি আমরা।

```
struct node{
 int data;
 node *next=NULL;
};
```

(Curly brace এর শেষে সেমিকোলন আছে) এখানে node একটি User defined datatype যার মধ্যে আমরা int টাইপ একটি ফিল্ড (data) এবং পরবর্তী নোডের এড্রেস

রাখার জন্য একটি node* পয়েন্টার নিয়েছি।

আমরা একেকটি নোড কিভাবে তৈরি করবো? এটা খুবই সহজ। যারা C++ ব্যবহার করি তারা খুব সহজেই new কিওয়ার্ড ব্যবহার করে কাজটি করে নিতে পারবো নিচের মতো করে।

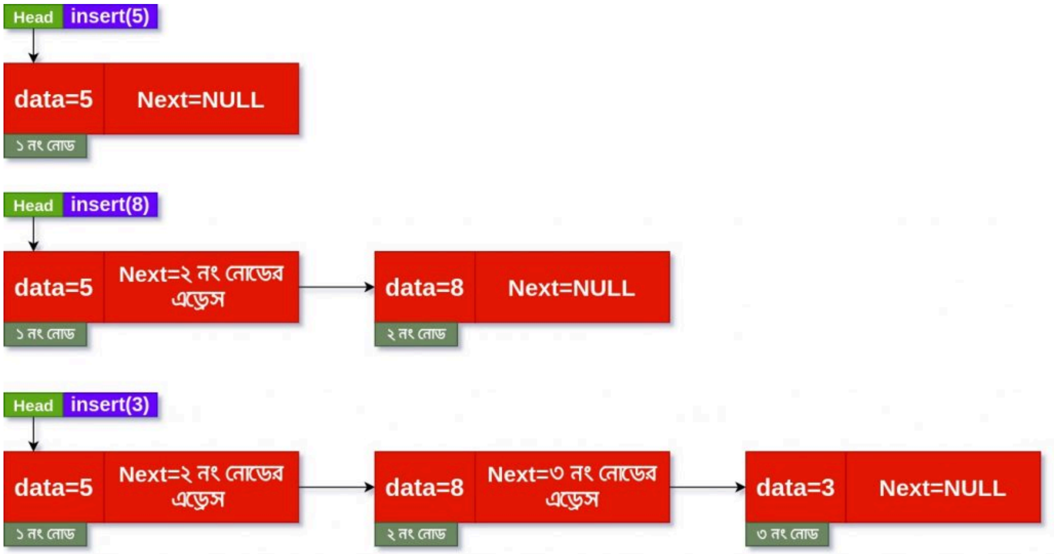
```
node *new_node=new node;
```

এখানে পয়েন্টার ব্যবহার করার কারণ হলো আমরা যখন বিভিন্ন অপারেশন করবো (insert,delete) তখন আমাদের বিভিন্ন সময় কোন একটি নোডের ফিল্ড আপডেট করতে হবে। যার জন্য ওই নোডের এড্রেসে গিয়ে কাজ করতে হয়। এই জন্য আমাদের পয়েন্টার দিয়ে ওই এড্রেসটিকে পয়েন্ট করে রাখতে হয়।

দ্বিতীয়ত, আমরা যদি একটি ফাংশনের মধ্যে সাধারণ ভেরয়েবল তৈরি করি, তবে ফাংশনটির স্কোপ থেকে বেরনোর সাথে সাথেই ভেরিয়েবলটি ডিলিট হয়ে যাবে কারণ এভাবে ভেরয়েবল তৈরি করলে স্ট্যাক মেমরিতে তৈরি হয়। কিন্তু আমরা যখন new কিওয়ার্ড ব্যবহার করবো তখন এর মেমোরি হীপ (heap) এ তৈরি হবে। যার ফলে ফাংশনের স্কোপ শেষের সাথে সাথে ভেরয়েবলটি ডিলিট হবে না। আরও জানতে এই পোস্টটি পরুন মেমোরি ম্যানেজমেন্ট: স্ট্যাক এবং হিপ মেমোরি।

লিঙ্কড লিস্ট এ ডাটা insert করা – Insertion in linked list

লিঙ্কড লিস্ট এ ডেটা রাখতে হলে অবশ্যই একটি নতুন নোড তৈরি করতে হবে। নিচের চিত্রে প্রক্রিয়াটি বর্ণনা করার চেষ্টা করেছি।



প্রথমেই আমদেরকে লিস্টের শুরতে যে নোড থাকবে তার এড্রেসকে কোন একটি ভেরিয়েবলে জমা করে রাখতে হবে। আমরা এর নাম দিলাম head। উপরের চিত্রে খেয়াল করি। প্রথম ধাপে আমরা ৫ ইনসার্ট করেছি। এর জন্য আমদেরকে একটি নতুন নোড তৈরি করতে হবে।

```
node *new_node=new node;
new_node->data=5;
```

যেহেতু এই নোডটি ই আমদের শুরুর নোড, তাই Head নামক পয়েন্টারে এই নোডটির এড্রেস জমা করে রাখবো আমরা। লক্ষকরই, এই নোডের next পয়েন্টার এর মান NULL।

```
if(head==NULL){
    head=new node;
    return;
}
```

দ্বিতীয় ধাপে আমরা যখন ৮ ইনসার্ট করেছি তখন একই ভাবে আমরা একটি নতুন নোড তৈরি করেছি। এই নোডটিকে প্রথম নোডের শেষে রাখতে হবে। কিভাবে রাখবো? উপায় হলো প্রথম নোডের next পয়ন্টার দিয়ে এই নোডটিকে পয়েন্ট করে দেয়া। এর জন্য আমরা শুরুতে ১ নং নোড এক্সেস করবো এবং এর next নামক পয়েন্টারে এই নোডের এড্রেস বসাবো। এখানেও লক্ষ করি, নোড ২ এর next পয়েন্টার NULL।

৩ নং ধাপ ও আমরা একই ভাবে সম্পন্ন করেছি। নোড ১ থেকে next এর মাধ্যমে আমরা নোড ২ এ গিয়েছি। নোড ২ এর next পয়েন্টার যেহেতু NULL, তাই আমরা বুঝতে পারি এটিই শেষ নোড, তাই এই next পয়েন্টার এর মাধ্যমে নোড ৩ কে পয়েন্ট করে দিয়েছি।

এককথায় বলতে গেলে, শুরুর নোড (head node) বাদে বাকি নোডগুলো insert করতে আমরা while লুপের মাধ্যমে শেষ নোড খুঁজে বের করবো। তারপর এর next পয়েন্টারে আমরা নতুন নোডকে পয়েন্ট করবো।

লিঙ্কড লিস্ট এ insert() ফাংশন।

লিঙ্কড লিস্টের সবার শেষে ইনসার্ট করা খুবই সোজাসাপ্টা। আমরা প্রথমে কোড দেখবো। তারপর কোডের ব্যাখ্যা করবো।

```
struct node{
 int data;
 node *next=NULL;
};
```

```
5
6  node *head=NULL;
7  void insert(int n){
8   node *new_node=new node;
9   new_node->data=n;
10
11  if(head==NULL){
12  head=new_node;
13  return;
14  }
15
16  node *current;
17  current=head;
18  while(current->next!=NULL){
19  current=current->next;
20  }
21  current->next=new_node;
22
23 }
```

কোডের প্রথমে ১ থেকে ৪ নং লাইনে আমরা node এর স্ট্রাকচার ডিফাইন করেছি। দেখতে পারছি এখানে দুইটি ফিল্ড আছে, একটি ডাটা রাখার জন্য int টাইপ এবং আরেকটি পয়েন্টার পরবর্তী নোডের এড্রেস রাখার জন্য। এর পর আসি ৬ নং লাইনে। আগেই বলেছিলাম, লিঙ্কড লিস্টের শুরু বুঝাতে একটি পয়েন্টার লাগবে, যেখানে আমরা শুরুর নোডটি রাখবো। এই head পয়েন্টারটিই হলো আমদের সেই ভেরিয়েবল।

এর পরে আসে insert() ফাংশন। এই ফাংশন একটি প্যারামিটার নিবে। এই প্যারামিটারের মাধ্যমে আমরা ডাটা পাস করবো।

এখন পরের if স্টেটমেন্ট এ দেখি, যদি head পয়েন্টার নাল বা খালি হয়, তাহলে আমরা কি বুঝতে পারি? বুঝতে পারি আমাদের লিঙ্কড লিস্টে কোন নোড নাই। তাই এই ক্ষেত্রে আমরা পরীক্ষা করেছি head এর মান NULL নাকি। NULL হলে new_node কে head পয়েন্টার দ্বারা পয়েন্ট করেছি এবং ফাংশন থেকে বেরিয়ে গিয়েছি।

এখন আসা যাক ১৬-১৭ নং লাইনে। এখানে লুপের ভিতর বিবেচ্য নোডের ট্র্যাক রাখার জন্য একটি ভেরিয়েবল নিয়েছি এবং শুরুর মান হিসেবে head নোড (অন্য কথা নোড ১) কে পয়েন্ট করে দিয়েছি।

এবার লুপে আমরা খুঁজেছি লিস্টের শেষ নোড কোনটা। কারণ আমরা লিস্টের শেষ নোডের পরে নতুন নোড রাখবো। আমরা যখন current = শেষ নোডে এসে পরবো তখন লুপ থেমে যাবে। এভাবে current পয়েন্টার আমাদের লিস্টের শেষ নোডকে পয়েন্ট করবে। অতঃপর আমরা current এর next ফিল্ডে নতুন নোড হিসেবে new_node কে যুক্ত করবো।

শেষ নোড বুঝার উপায়: শেষ নোডের next পয়েন্টার সবসময় NULL থাকবে। যতক্ষণ পর্যন্ত `current->next==NULL` না হয় ততক্ষণ আমরা লুপ চালিয়ে পরের নোড গুলোতে

যেতে থাকবও।

লিঙ্কড লিস্ট এ সার্চ করা – Searching in linked list

ধরা যাক আমরা একটি লিস্ট তৈরি করেছি। এখন আমাদেরকে এই লিস্ট থেকে একটি সংখ্যা n খুঁজে বের করতে হবে। আমরা কিভাবে সার্চ করার কাজটি করবো? লিঙ্কড লিস্টে সার্চ করা খুবই সহজ। আমরা কোডটি দেখি আগে,

```
1 bool search(int n){
2  node *current = head; // head পয়েন্টারকে আমরা current নোড বানাবো প্রথমে।
3  while(current!=NULL){ // লিস্টের শেষ নোড পর্যন্ত সর্বচ্চো আমরা লুপ চালাবো।
4  if(current->data==n) return true; // যদি current নোড এর data ফিল্ড n এর সমান হয়
5  current=current->next; // আগের শর্ত পূরণ না হলে current নোড এর next নোড কে cur
6  }
7  return false; // উপরের লুপে true রিটার্ন না হলে এখানে এসে false রিটার্ন হবে।
8 }
```

উপরের কোডটি খুব সোজা একটি কোড। প্রথমেই ২ নং লাইনে আমরা `current=head` করে দিয়েছি। কারণ আমরা head নোড থেকে খোজা শুরু করবো।

তারপর আমরা লুপে ঢুকেছি, যেখানে লুপের শর্ত হলো `current!=NULL`, অর্থাৎ শেষ নোডে চলে আসলে আমাদের current পয়েন্টার এর মান NULL হবে। লুপের ভেতরে আমরা current পয়েন্টারের data ফিল্ডের মান পরীক্ষা করে দেখছি এর মান n এর সমান হয় কিনা। যদি সমান হয় তবে true রিটার্ন করেছি, অন্যথায় current কে বর্তমান নোডের next নোডে পয়েন্ট করে দিয়েছি।

সব শেষ যদি লুপের মধ্যে true রিটার্ন না হয়, তবে আমরা লুপের বাইরে false রিটার্ন করেছি।

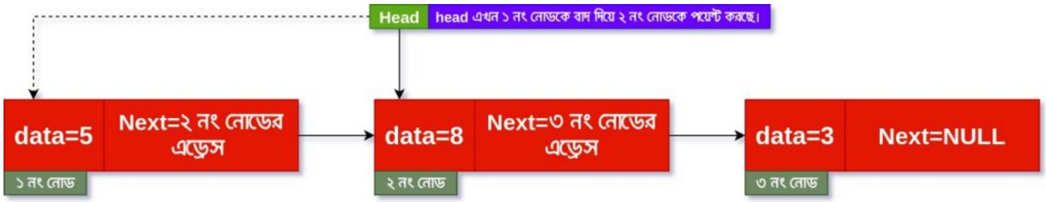
লিঙ্কড লিস্ট এ একটি নোড ডিলিট করা – Deletion of a node in Linked List

লিঙ্কড লিস্টে ডিলিট অপারেশন করার জন্য আমাদেরকে ৩ টি কেস মাথায় রাখতে হবে। যথাক্রমে, (১) শুরুর একটি নোড ডিলিট করার, (২) শেষের একটি নোড ডিলিট করা এবং (৩) মাঝখানের একটি নোড ডিলিট করা।

কিন্তু এই ৩ টি কেসের যাই হোকনা কেন, আমদেরকে প্রথমে যেই নোড ডিলিট করবো তাকে খুঁজে বের করতে হবে। এর জন্য আমরা সার্চ অপারেশন করবো।

শুরু থেকে ডিলিট করা বা head যাকে পয়েন্ট করেছে ওই নোড ডিলিট করা।

head নোড ডিলিট করতে হলে আমাদেরকে শুধু head কে আপডেট করে `head=head->next` করে দিতে হবে। এতে করে head এর জায়গাকে পরবর্তী নোডের এড্রেস দ্বারা আপডেট করা হবে এবং পুর্বের head এর কোন ট্রেস থাকবে না আমাদের।



লিঙ্কড লিস্টে আমাদের সব অপারেশন head থেকে করা লাগে। তাই উপরের চিত্র থেকে বুঝা যাছে আমাদের অপারেশন এখন থেকে ২ নং নোড থেকে শুরু হবে।

শেষ থেকে কোন নোড ডিলিট করা

শেষ থেকে ডিলিট করতে হলে আমাদের কে শুধু শেষ নোডের আগের নোডের next ফিল্ডে NULL সেট করে দিতে হবে। এতে করে আমাদের লিস্টের শেষ নোডের এড্রেস আর সেভ থাকবে না।

এখানে দেখা যাচ্ছে ২ নং নোডের next ফিল্ড NULL করা হয়েছে। এবং ৩ নং নোডের সাথে ২ নং নোডের সকল সংযোগ বিচ্ছিন্ন। এতে করে ৩ নং নোড আমরা আর যেতে পারব না এবং ২ নং নোড আমাদের শেষ নোড হিসেবে বিবেচ্য হবে।

মাঝখান থেকে একটি নোড ডিলিট করা

মাঝখান থেকে একটি নোড ডিলিট করতে হলে আমরা ওই নোডের পরের নোডের সাথে ওই নোডের আগের নোডের একটি কানেকশন করে দিবো আগের নোডের next ফিল্ডের মাধ্যমে।

উপরের চিত্রে ২ নং নোড ডিলেট করা হয়েছে। তাই ১ নং নোডের সাথে ৩ নং নোডের সাথে সংযোগ স্থাপন করা হয়েছে। এতে করে ২ নং নোড কে লিস্ট থেকে বাদ দেয়া গিয়েছে।

```
void delete_node(int n){
 node *current,*previous;
 current=head;
 previous=NULL;
 while(current!=NULL){
 if(current->data==n){ // আমরা যখন সঠিক নোডটি খুঁজে পাবো
 if(previous==NULL){ // যখন previous পয়েন্টার NULL হবে, তার মানে ট্রাগেট নোড একটি
 head=current->next; // পরের নোডটিকে head  নোড করে দিলাম
 delete[] current; // বর্তমান নোডের মেমরি ফ্রি করে দেয়া হলো
 return;
 }else if(current->next==NULL){ // যদি আগের শর্ত পূরণ না করে এই শর্ত পূরণ করে ত
 previous->next=NULL; // শেষের নোডের আগের নোডকে NULL করে দিলাম।
 delete[] current; // বর্তমান নোডের মেমরি ফ্রি করে দিলাম।
 return;
 }else{ // প্রথম না শেষ ও না, তার মানে মাঝের নোড
 previous->next=current->next; //আগের নোডের সাথে পরের নোডের সম্পর্ক করে দিয়েছি
 delete[] current;  // মেমোরি ফ্রি করে দিলাম/
 return;
 }
 }
 previous=current;
 current=current->next;
 }
}
```

আশা করি উপরের তিনটি বেসিক অপারেশন বুঝা গিয়েছে।

উপরের যে insert অপারেশন করেছি, তাকে বলা হয় append. আমরা এছাড়াও, লিস্টের সবার সামনে insert করতে পারব। লিস্টের মাঝেও insert করতে পারব। লিস্টে সবার সামনে insert করতে হলে আপনাদের যা করতে হবে তার কোড নিচের মতো,

```
1 void insert_head(int n){
2  node *new_node=new node;
3  new_node->data=n;
4  new_node->next=head;
5  head=new_node;
6 }
```

এখানে আমরা স্বাভাবিক ভাবেই নতুন নোড তৈরি করেছি। তারপর ৪ নং লাইনে নতুন নোডের next ফিল্ডে আগের head নোডকে পাঠিয়ে দিয়েছি এবং নতুন head নোড হিসেবে নতুন নোডকে বসিয়ে দিয়েছি।

এখন আমরা যদি কোন একটি নোডের পরে একটি নতুন নোড insert করতে চাই তবে আমাদেরকে প্রথমে ওই নোডে যেতে হবে। ধরা যাক এই নোডটি x এই নোডের পরের নোডটি y. এখন আমরা x এবং y এর মাঝে আরেকটি নোড z বসাতে চাই। এটা করার

জন্য আমাদেরকে সার্চের মাধ্যমে প্রথমে x নোডে যেতে হবে। তারপর x নোডের next দ্বারা z কে পয়েন্ট করাতে হবে। তারপর z এর next ফিল্ড দ্বারা y কে পয়েন্ট করাতে হবে। এতেই আমাদের কাজ হয়ে যাবে। এই কোডটা আপনারা নিজে অনুশীলন করুন। বুঝতে সমস্যা হলে কমেন্ট করুন সবাই।

লিখাটা কেমন লাগলো কমেন্টে জানাতে ভুলবেন না। আবার আরেকটা লিখা পর্যন্ত। বিদায়।

লেখাটি কেমন লেগেছে আপনার?

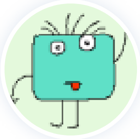
রেটিং দিতে হার্টের উপর ক্লিক করুন।

♥♥♥♥♥

গড় রেটিং 4.4 / 5. মোট ভোট: 121

#Data structures **#অ্যালগরিদম** **#ডাটা স্ট্রাকচার**

8 টি মন্তব্য /



Mahmudul Hasan

August 4, 2021 at 12:58 AM

Awesome vaiya

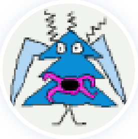
Reply

Sharif Hasan

August 6, 2021 at 11:56 AM

Thank you brother ❣️

Reply



gUeSS

October 14, 2021 at 10:55 PM

Vaiya, ekta node null bolte ki bujhasse? node er data ba next null ok. kintu node null mane ki ei duitai null?

r print function ta kemn hoto jodi bolten...

```
void print(){
node *this_node=head; //copy of root node
while(this_node!=NULL){
printf(“%d “,this_node->data);
this_node=this_node->next;
}
}
```

eTa kaj korse na..

Reply

Sharif Hasan

October 16, 2021 at 2:32 AM

node টাইপের একটি পয়েন্টার নাল মানে হচ্ছে ওই নোডে কোন নোড এসাইন করা হয় নি। তারমানে ওই নোডে ডাটা এবং নেক্সট ফিল্ডের জন্য কোন এড্রেস নেই, অর্থাৎ ডাটা এবং নেক্সট ফিল্ডের অস্তিত্ব নেই।

Reply

Rohan

January 23, 2022 at 10:52 PM

Vaiya doubly linked list niye jodi likhten....

Reply

Sharif Hasan

January 29, 2022 at 11:24 PM

I'll cover it in my next writing <3 .

Reply

Usha

December 10, 2022 at 8:55 PM

অ্যারে এর মৌলিক অপশন গুলো লিখুন??
এটি দ্বারা কি বুঝায়??? প্রশ্নটা কি অপশন না হয়ে অপারেশন হবে??

Reply

Sharif Hasan

December 10, 2022 at 9:28 PM

লিখবো ইনশাআল্লাহ।
ওইটা অপারেশন হবে

Reply