“To My Presentation”

Presented by:
S. Juthi Al Saki
Instructor (Tech) Computer
Mymensingh Polytechnic Institute.

Mymensingh Polytechnic Institute

Technology: Computewr Science & Engineering

Subject Name: Data Structure & Algorithm

Subject Code: 28542

Semester: 4th

Shift: 2nd

অধ্যায়-১ ঃ ডাটা স্ট্রাকচার

১.১ ডাটা এবং ইনফরমেশন এর সংজ্ঞা

ডাটা: ডাটা শব্দের অর্থ হলো উপাত্ত।সাধারণ অর্থে কোন বিষয় বা ঘটনার মানকে ডাটা (Data) বা উপাত্ত বলা হয়। একাধিক Data item নিয়ে একত্রে গ্রুপ আইটেম (Group Item) বলে। Data শব্দ Latin শব্দ Daturn শব্দ-এর বহুবচন। Data-এর কোন নিজস্ব স্বকীয় সত্তা নেই। এই Data-কে proces করে যে output পাওয়া যায় তাকে information বলে। যেমন- রাসেল এর বয়স ৯ বছর। এখানে আলাদাভাবে যদি রাসেল, ৯, এগুলো চিন্তা করা হয় তবে এর কোন অর্থ নেই, কিন্তু পুরো বাক্যটিকে চিন্তা করলে একটি অর্থ পাওয়া যায়।

অধ্যায়-১ ঃ ডাটা স্ট্রাকচার

১.১ ইনফরমেশন এর সংজ্ঞা

তথ্য (Information) : সংগৃহীত ডাটা বা উপাত্ত প্রক্রিয়াকরণের পর অর্থপূর্ণ অবস্থাকে তথ্য বা ইনফরমেশন (Information) বলা হয়। অন্যভাবে বলা যায়, কোন নির্দিষ্ট কাজের জন্য সংগৃহীত ডাটাকে প্রক্রিয়াকরণ করে যে অর্থবহ, নির্ভুল ও যৌক্তিক ফলাফল পাওয়া যায়, তাকে ইনফরমেশন বলে। যেমন : প্রাপ্ত নম্বরের সমষ্টি বা গড়, পরীক্ষার ফল, বয়সের গড় ইত্যাদি। ইনফরমেশন একটি বহুবচন শব্দ। তথ্য বিভিন্ন ধরনের হতে পারে। যেমন: নাম ও বয়সসহ মোবাইল নং, নম্বরভিত্তিক ফলাফল, ছবি, অডিও, ভিডিও, ব্যবসায়িক রিপোর্ট, বৈজ্ঞানিক গবেষণার ফলাফল ইত্যাদি

অধ্যায়-১ ঃ ডাটা স্ট্রাকচার

ডাটা এবং ইনফরমেশনের মধ্যে পার্থক্য ঃ

ডাটা	ইনফরমেশন
১। ডাটা হলো কোন নির্দিষ্ট উপাত্ত।	ডাটা প্রসেসিং এর ফলে ইনফরমেশন পাওয়া যায়।
২। ডাটা প্রসেসিং এর প্রাইমারি ধাপ হলো ডাটা।	ডাটা প্রসেসিং এর সেকেন্ডারি ধাপ হলো ইনফরমেশন।
৩। ডাটা কোন অর্থ প্রকাশ করে না।	ইনফরমেশন নির্দিষ্ট অর্থ প্রকাশ করে।
৪। ইনফরমেশন ডাটা কোন ব্যক্তি, বস্তু বা বিষয়ের নাম, পরিমাণ, ধর্ম,বর্ণ ইত্যাদি নির্দেশ করে। যেমনঃ টেবিল,৬৫০ ইত্যাদি।	ইনফরমেশন কোন বিষয়বস্তুর অর্থপূর্ণ ও ব্যবহারের উপযোগী অবস্থা নির্দেশ করে। যেমনঃ এ,বি,এফ গ্রেড ইত্যাদি।
৫। ইনফরমেশন ডাটা অপক্রিয়াজাত অবস্থায় থাকে।	ইনফরমেশন পক্রিয়াজাত অবস্থায় থাকে।

অধ্যায়-১ ঃ ডাটা স্ট্রাকচার

১.২ স্ট্যান্ডার্ড ডাটা টাইপ উল্লেখকরণ (Mention Standard Data Types)

বিভিন্ন ধরনের কাজের উপর ভিত্তি করে ডাটার ধরন ও সংরক্ষিত মেমোরি পরিসরের দিক বিবেচনা করে ডাটার শ্রেণিবিভাগ করা হয়েছে। আবার প্রোগ্রামিং-এর ক্ষেত্রে বিভিন্ন ভাষায় বিভিন্ন ধরনের ডাটা ব্যবহৃত হয়। ডাটার প্রকৃতির উপর ভিত্তি করেডাটা প্রধানত দুই প্রকারের। যথা-

(ক) আলফাবেটিক ডাটা (Alphabetic Data) ও

(খ) নিউমেরিক ডাটা (Numeric Data)।

অধ্যায়-১ ঃ ডাটা স্ট্রাকচার

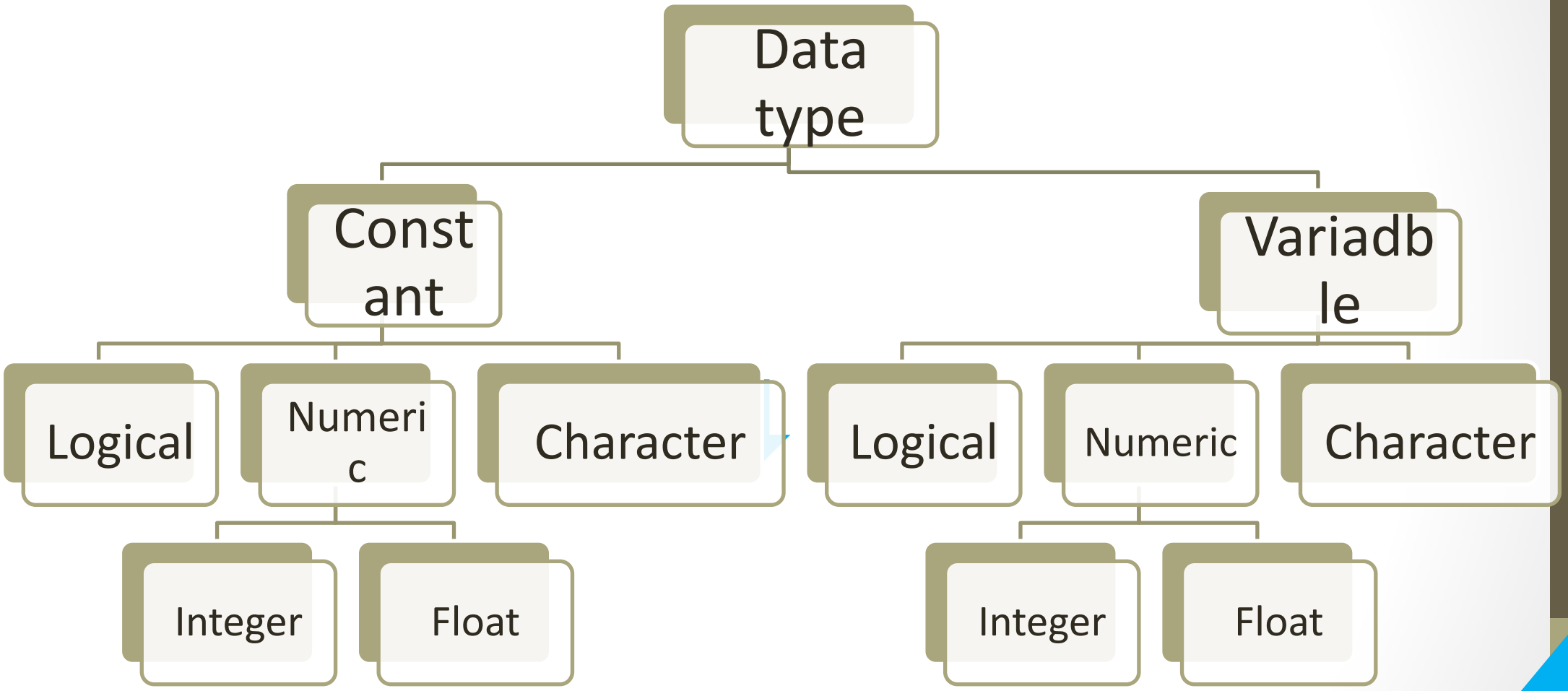
১.২ স্ট্যান্ডার্ড ডাটা টাইপ উল্লেখকরণ (Mention Standard Data Types)

(ক) আলফাবেটিক ডাটা (Alphabetic Data) : এক বা একাধিক অক্ষর বা বর্ণের সমন্বয়ে আলফাবেটিক ডাটা তৈরি হয়। যেমন : A, B, a, 2, #, *, %, "Computer", "Dhaka", "Bangladesh" ইত্যাদি।

আলফাবেটিক ডাটাসমূহকে আবার দুই শ্রেণিতে ভাগ করা হয়। যথা-
(i) ক্যারেক্টার (Character) ও
(ii)স্ট্রিং (String)।

খ) নিউমেরিক ডাটা (Numeric Data) : পূর্ণ বা দশনিক মানবিশিষ্ট বিভিন্ন ধরনের সাংখ্যিক মানসমূহ নিউমেরিক অন্তর্গত। নিউমেরিক ডাটার উদাহরণ হলো : 10, 20, 152.45, 3.5 x 10200

অধ্যায়-১ ঃ ডাটা স্ট্রাকচার



অধ্যায়-১ ঃ ডাটা স্ট্রাকচার

১.৩ ডাটা স্ট্রাকচার বর্ণনা(State Data Structure)

উপাত্ত সংগঠন বা ডাটা স্ট্রাকচার বলতে ডাটাকে কম্পিউটারের স্মৃতিতে রাখার একটি নির্দিষ্ট উপায়কে বোঝায়, যাতে প্রয়োজনে উপাত্ত/ডাটাকে দক্ষতার সাথে ব্যবহার করা যায়। যত্নের সাথে বাছাই করা ডাটা স্ট্রাকচার, ডাটার উপর সবচেয়ে দক্ষ অ্যালগরিদমের ব্যবহার সম্ভব করে তোলে। একটি সুপরিকল্পিত উপাত্ত সংগঠন মেমোরি ও সময় যথাসম্ভব বাঁচিয়ে ডাটার উপর অনেকগুলো জরুরি অপারেশন প্রয়োগ করার সুযোগ দেয়। কোনো একটি প্রোগ্রামিং ভাষাতে প্রদত্ত ডাটা টাইপ রেফারেন্স ও অপারেশন অনুসারে ডাটা স্ট্রাকচার বাস্তবায়ন করা হয়।

অধ্যায়-১ ঃ ডাটা স্ট্রাকচার

১.৩ ডাটা স্ট্রাকচার বর্ণনা(State Data Structure)

কোন নির্দিষ্ট ডাটা মডেল (Data Model) কেমন হবে তা দুটি বিষয়ের ওপর নির্ভর করে। যথা-

(ক) স্ট্রাকচারের আওতাধীন ভাটাকে এমন বৈশিষ্ট্যের অধিকারী হতে হবে, যাতে বাস্তব জীবনে ভাটাসমূহের মধ্যে প্রকৃত সম্পর্কের প্রতিফলন ঘটে।

(খ) স্ট্রাকচার (Structure) অবশ্যই সাধারণ (Simple) হতে হবে যেন যে কেউ প্রয়োজনবোধে কার্যকরভাবে ডাটা Process করতে পারে।

১.৪ ডাটা স্ট্রাকচারের প্রকারভেদ বর্ণনা
(State Types of Data Structure)

বিভিন্ন ধরনের অ্যাপ্লিকেশনের উপর ভিত্তি করে ডাটা স্ট্রাকচারের শ্রেণিবিভাগ করা হয়েছে। ডাটা স্ট্রাকচারকে প্রধানত দুইভাগে ভাগ করা যায়, যথা-

(ক) প্রিমিটিভ ডাটা স্ট্রাকচার (Primitive Data Structure)
(খ) মন-প্রিমিটিভ ডাটা স্ট্রাকচার (Non Primitive Data Sturcture)

অধ্যায়-১ ঃ ডাটা স্ট্রাকচার

প্রিমিটিভ ডাটা স্ট্রাকচার আবার চার প্রকার, যথা-
(i) ইন্টিজার (Integer)
(ii)ফ্লোট (Float)
(iii)ক্যারেক্টার (Character)
(iv)বুলিয়ান (Boolean)।

নন-প্রিমিটিভ ডাটা স্ট্রাকচার আবার দুই ভাগে বিভক্ত, যথা-
(i) লিনিয়ার ডাটা স্ট্রাকচার (Linear Data Structure)
(ii)নন-লিনিয়ার ডাটা স্ট্রাকচার (Non-Linear Data Structure)।

অধ্যায়-১ ঃ ডাটা স্ট্রাকচার

লিনিয়ার ডাটা স্ট্রাকচার আবার চার ভাগে বিভক্ত, যথা-
(i) অ্যারে (Array)
(ii) লিংক লিস্ট (Link List)
(iii) স্ট্যাক (Stack)
(iv) কিউ (Queue)

নন লিনিয়ার ডাটা স্ট্রাকচার আবার দুই ভাগে বিভক্ত। যথা-
1. (Tree)
2. (Graph) ইত্যাদি।

অধ্যায়-১ ঃ ডাটা স্ট্রাকচার

১.৫ বিভিন্ন প্রকার ডাটা অপারেশনের তালিকা(List the Different Types of Data Operation)

Data Structure এর বিভিন্ন Operation গুলো নিম্নে আলোকপাত করা হলো-

ক. ট্রাভার্সিং (Traversing),
খ. সার্চিং (Searching),
গ. ইনসার্টিং (Inserting),
ঘ. ডিলিটিং (Deleting).

এছাড়া দুটি Operation রয়েছে, অপারেশন দুটি বিশেষ বিশেষ অবস্থায় ব্যবহার করা হয়।

ঙ. সর্টিং (Sorting),
চ. মার্জিং (Merging)।

অধ্যায়-১ ঃ ডাটা স্ট্রাকচার

নিম্নে সংক্ষেপে Data Structure-এর Operation গুলোর বর্ণনা দেয়া হলো-

ক. ট্রাভার্সিং (Traversing) : কোন record-কে একমাত্র একবার Access এবং Process করাকে Traversing বলে। Accessing 3 processing-কে কখনো কখনো visiting বলা হয়।

খ. ইনসার্টিং (Inserting) : পূর্বে তৈরি করা কোন Data Structure-এর সাথে নতুন Record যোগ করার কৌশলয়ে Inserting বলে।

গ. সার্চিং (Searching) : একটি নির্দিষ্ট Key Value-এর মাধ্যমে কোন Record-এর Location খুঁজে বের করা পদ্ধতিকে Scarching বলে।

ঘ. ডিলিটিং (Deleting) : ডাটা স্ট্রাকচার থেকে কোন record-কে বাদ দেয়ার কৌশলকে deleting বলে।

ঙ. সর্টিং (Sorting) : ডাটাসমূহকে মানের ক্রমানুসারে সাজানোকে Sorting বলে।

চ.মার্জিং (Merging) : দুইটি ভিন্ন file-এর record-কে একত্রিত করার কৌশলকে Merging বলে।

অধ্যায়-১ ঃ ডাটা স্ট্রাকচার

১.৬. স্ট্যাটিক এবং ডাইনামিক মেমোরি অ্যালোকেশন বর্ণনা (State Static and Dynamic Memory Allocation)

Language-এর ওপর ভিত্তি করে Array-এর দুইভাবে মেমোরি Space নির্দিষ্ট (Allocate) হয়।

1. Static Memory Allocation.
2. Dynamic Memory Allocation.

অধ্যায়-১ ঃ ডাটা স্ট্রাকচার

স্টাটিক মেমোরি অ্যালোকেশন (Static Memory Allocation) : কোন কোন Language যেমন Pascal এবং Fartan-4 Compile করার সময় Statically মেমোরি Allocate হয়। এ কারণে Program নির্বাহের সময় অ্যারের সাইজ অপরিবর্তিত (Fixed) থাকে। এই পদ্ধতিকে স্টাটিক মেমোরি অ্যালোকেশন (Static Memory Allocation) বলা হয়।

ডাইনামিক মেমোরি অ্যালোকেশন (Dynamic Memory Allocation) : কিছু কিছু Language যেমন- C-তে মোট কয়টি ডাটা সংরক্ষণ করতে হবে তা পূর্বেই জানিয়ে দেয়ার বাধ্যবাধকতা নেই। প্রোগ্রাম চলাকালীন সময় ডাটা সংরক্ষণ ও ব্যবহারের জন্য নিজেদের প্রয়োজন অনুযায়ী মেমোরি Allocate হয়। এই পদ্ধতিকে ডাইনামিক মেমোরি অ্যালোকেশন (Dynamic Memory Allocation) বলে।

অধ্যায়-২ ঃ অ্যালগরিদম

২.১ অ্যালগরিদম (State Algorithm)

কোন সমস্যা সমাধানের সুস্পষ্ট ধাপসমূহের ক্রমবিন্যাসকে Algorithm বলে। অ্যালগরিদম অর্থ ধাপে ধাপে সমস্যা সমাধান অর্থাৎ একটি সমস্যাকে কয়েকটি ধাপে ভেঙে প্রত্যেকটি ধাপ পরপর সমাধান করা। অ্যালগরিদম Programming এর গুরুত্বপূর্ণ অংশ। Algorithm সঠিক না হলে Program সঠিকভাবে কাজ করে না।কোন কার্য সম্পাদনের ক্ষেত্রে কতগুলো যৌক্তিক সিদ্ধান্তের মাধ্যমে পর্যায়ক্রমে কার্য সম্পাদনের পরিকল্পনা করা হয়। এই ধরনের পরিকল্পনাকেই ঐ কাজটির অ্যালগরিদম বলা যায়।

অধ্যায়-২ ঃ অ্যালগরিদম

প্রোগ্রাম রচনার ক্ষেত্রে অ্যালগরিদম শিক্ষা গুরুত্বপূর্ণ বিষয়। অ্যালগরিদম প্রোগ্রাম পরিকল্পনার অংশবিশেষ। প্রোগ্রামের পরিকল্পনা যত সহজ ও যথার্থ হয় প্রোগ্রাম ততই সহজ ও নির্ভুল হয়। অ্যালগরিদম লেখনের পর তা থেকে সমাধানের জন্য বিভিন্ন উপায় ব্যবহার করা যেতে পারে। তবে সহজ ও যথার্থ উপায় নির্ধারণের জন্য অ্যালগরিদম শিক্ষা অপরিহার্য। সমস্যাকে পর্যায়ক্রমিক ছোট ছোট ভাগ করে তা সমাধানের উদ্দেশ্যে একজন প্রোগ্রামার অ্যালগরিদমের রচনায় প্রয়াসী হয়ে থাকেন। অর্থাৎ সহজ ও পরিচ্ছন্ন প্রোগ্রাম রচনায় অ্যালগরিদম শিক্ষা বিশেষভাবে সহায়তা করে থাকে। প্রোগ্রাম রচনায় অ্যালগরিদম শিখনের সুবিধা হলো-

(ক) সহজে প্রোগ্রামের উদ্দেশ্য বুঝতে পারা।
(খ) প্রোগ্রামের ভুল নির্ণয়ে সক্ষম হওয়া।
(গ) প্রোগ্রাম পরিবর্তন ও পরিবর্ধনে দক্ষতা অর্জন।
(ঘ) সহজে ও সংক্ষেপে জটিল প্রোগ্রাম লেখতে পারা ইত্যাদি ।

অধ্যায়-২ ঃ অ্যালগরিদম

২.২ অ্যালগরিদম-এর বৈশিষ্ট্য উল্লেখ
(Mention the Characteristics of Algorithm)

Algorithm রচনার ক্ষেত্রে নিচের বিষয়গুলো অত্যন্ত গুরুত্বপূর্ণ।

(ক) Algorithm এর নির্দিষ্ট সংখ্যক input থাকবে।
(খ) প্রতিটি.Algorithm এর output থাকবে।
(গ) Algorithm যথাসম্ভব ছোট ও সহজবোধ্য হতে হয়।
(ঘ) নির্দিষ্ট সংখ্যক ধাপে সমস্যার বর্ণনা করতে হয়।
(ঙ) প্রতিটি ধাপ সুস্পষ্ট এবং সহজভাবে বর্ণনা করতে হয়।

অধ্যায়-২ ঃ অ্যালগরিদম

(চ) মূল সমস্যাকে কতগুলো ধাপে ভেঙে সমস্যাটি সহজ করতে হয়।
(ছ) অ্যালগরিদম নির্বাহের সময়কাল নির্দিষ্ট মাত্রায় থাকা উচিত।
(জ) Algorithm বাস্তবায়নের জন্য ন্যূনতম Memory space ব্যবহার হওয়া উচিত।
(ঝ) Algorithm লেখনে একটি নির্দিষ্ট ভাষা অনুসরণ করতে হয় ।
(ঞ) একই সমস্যার Algorithm বিভিন্ন রকম হতে পারে কিন্তু মূল বক্তব্য একই হয় ।
(ট) প্রতিটি Algorithm প্রোগ্রামের অবশ্যই সমাপ্তি থাকতে হবে।

অধ্যায়-২ ঃ অ্যালগরিদম

২.৩ ফ্লো চার্ট এবং সিউডো কোড (State flow chart and pseudo code)

কোন সমস্যা সমাধানের যাপসমূহকে যখন চিত্রের সাহায্যে দেখানো হয়, তখন তাকে প্রবাহচিত্র বা ফ্লো চার্ট বলা হয়। কোন Algorithm এর সচিত্র (Graphical) উপস্থাপনাই হচ্ছে Flow chart। বিভিন্ন চিত্র বা Symbol বা প্রতীকের সাহায্যে Flow chart অঙ্কন করা হয়। Flow chart-এর মাধ্যমে Program সম্পর্কে সহজে ও দ্রুত ধারণা অর্জন সম্ভব। কাজেই Program রচনা ও Execution-এর জন্য এবং System Analysis-এর জন্য Flow chart অত্যন্ত প্রয়োজনীয় ।

অধ্যায়-২ ঃ অ্যালগরিদম

একটি উন্নত মানের Flow chart নিম্নলিখিত সুবিধা প্রদান করে-

- ✓ সহজে প্রোগ্রামের উদ্দেশ্য বুঝা যায়।
- ✓ প্রোগ্রামের ভুল নির্ণয়ে সহায়তা করে।
- ✓ প্রোগ্রাম পরিবর্তন ও পরিবর্ধনে সহায়তা করে।
- ✓ প্রোগ্রাম রচনায় সহায়তা করে
- ✓ সহজে ও সংক্ষেপে জটিল প্রোগ্রাম লেখা সম্ভব হয়।

Flow chart-এ কতকগুলো জ্যামিতিক ছবি বা Assembly চিহ্ন ব্যবহৃত হয়। Flow chart-এর প্রধান চিহ্ন ৬টি। এ ছাড়াও আরও কতকগুলো চিহ্ন মাঝে মাঝে প্রয়োজন হয়। ANSI (American National Standard Institute) এ চিহ্নগুলো তৈরি করেছে; এগুলো আন্তর্জাতিকভাবে স্বীকৃত।Flow chart দু'ধরনের হতে পারে।

- ❖ সিস্টেম ফ্লোচার্ট (System flow chart)
- ❖ ফ্লোচার্ট (Program flow chart).

অধ্যায়-২ ঃ অ্যালগরিদম

Flow chart আকার নিয়মাবলি Flow chart আঁকার নিয়মাবলি নিম্নে বর্ণনা করা হল-

(ক) প্রবাহ রেখার দ্বারা কোন চিহ্নের পর কোন চিহ্ন হবে তা বোঝানো হয়। সাধারণত ওপর থেকে নিম্নে বা বাম দিক থেকে ডান দিকে প্রবাহ অগ্রসর হয়।

(খ) একাধিক প্রবাহ রেখা পরস্পরকে ছেদ করলেও তাদের মধ্যে কোন Logical সম্পর্ক বা যোগাযোগ বোঝায় না।

(গ) চিহ্নগুলো ছোট বা বড় যে কোন সাইজের হতে পারে কিন্তু তাদের বিশিষ্ট আকৃতি যেন বজায় থাকে।

(ঘ) প্রত্যেক Flow chart-এর একটি নাম থাকবে তাছাড়া রচয়িতার নাম ও তারিখ দিতে হবে।

(ঙ) প্রয়োজনে চিহ্নের সাথে মন্তব্যও দেয়া যায়।

(চ) যতটুকু সম্ভব রেখার জেন কম হওয়া ভাল।

অধ্যায়-২ ঃ অ্যালগরিদম

ফ্লো-কার্ট এর বৈশিষ্ট্য (Characteristics of lowchart)

(ক) ফ্রো-চটি যতটা সম্ভব সহজ-সরল করা হয়।
(খ) চার্ট অ্যালগরিদমের গ্রাফিক্যাল উপস্থাপন হবে।
(গ) অ্যালগরিদমের গুরুত্বপূর্ণ সবকটি ধাপ ফ্লো চার্ট থাকবে।
(ঘ) ফ্লো-চার্ট এর শুরু এবং শেষ থাকবে।
(ঙ) ফ্লো-চাট এটি ব্যবহারের পূর্বে টেস্ট করে নিলে সঠিক আউটপুট পাওয়া যাবে।

নিয়ে ফ্লোচার্টের একটি উদাহরণ দেখানো হলো যেখানে একটি গ্রিন হাউসের টেম্পারেচার অটোমেটিক্যালি 20° সেলসিয়াস হতে 23° সেলসিয়াসের মধ্যে রাখার ব্যবস্থা করা হয়েছে। গ্রিনহাউসের টেম্পারেচার 20° সেলসিয়াসের নিচে গেলে হিটারঅন হবে এবং 23° সেলসিয়াসের বেশি হলে হিটার অফ হয়ে যাবে।

অধ্যায়-২ ঃ অ্যালগরিদম

সুডো কোড (Pseudo Code) সুডো হলো একটি গ্রিক শব্দ। সুডো শব্দের অর্থ হচ্ছে ছদ্ম বা কৃত্রিম। সুডো কোড বলতে এমন কিছু নির্দেশকে বুঝায় যা প্রোগ্রাম রচনার সহায়ক ভূমিকা পালন করে। প্রোগ্রাম উন্নয়নে সুডো কোড একটি জনপ্রিয় পদ্ধতি। প্রোগ্রামিং-এর ক্ষেত্রে সুডো কোড বলতে প্রকৃত প্রোগ্রাম বুঝায় না। প্রোগ্রামের ধরন ও কার্যাবলি তুলে ধরার জন্য কিছুসংখ্যক নির্দেশ বা স্টেটমেন্টের সমাহারকেই সুডো কোড বলে। তাই প্রোগ্রাম রচনার পূর্বে প্রোগ্রামের যে খসড়া তৈরি করা হয়, তাকেই সুডো কোড বলে। সুডো কোড থেকে প্রথমে অ্যালগরিদম ও পরে প্রোগ্রাম তৈরি করা সহজ হয়।নিয়ে দুটি সংখ্যার যোগফল নির্ণয়ের সুডো কোড দেওয়া হলো-

INPUT NUMBER 1
INPUT NUMBER 2
TOTAL =NUMBER 1+ NUMBER 2

অধ্যায়-২ ঃ অ্যালগরিদম

২.৪ অ্যালগরিদমিক নোটেশন ব্যাখ্যা (Explain Algorithmic Notations)

অ্যালগরিদম নোটেশন (Algorithm Notation) :

Algorithm বলতে কোন সমস্যা সমাধানের জন্য নির্দিষ্ট সংখ্যক সুস্পষ্ট ধাপসমূহের ক্রমবিন্যাসকে বুঝায়। প্রকৃতপক্ষে Algorithm হলো PDL বা সুডোকোডের ধারাবাহিক বিন্যাস। এক একটি সমস্যা সমাধানের জন্য Algorithm কে ভিন্ন ভিন্ন ভাবে উপস্থাপন করা হয়ে থাকে। এটি নির্ভর করে সমস্যা এবং তার সমাধানের ওপর। সেজন্য Algorithm এর Notation অবশ্যই গুরুত্বপূর্ণ। Algorithm এর সাহায্যে সমস্যা সমাধানের জন্য এতে বিধিবদ্ধ কিছু সংকেত ব্যবহার করা হয়। Algorithm-এর জন্য এ ধরনের বিধিবদ্ধ সংকেতকে Algorithm Notation বলে।

অধ্যায়-২ ঃ অ্যালগরিদম

২.৫ অ্যালগরিদমের কমপ্লেক্সসিটি
Describe the Complexity of Algorithm)

অ্যালগরিদম বিশ্লেষণ কম্পিউটার বিজ্ঞানের একটি গুরুত্বপূর্ণ বিষয়। অ্যালগরিদম বিশ্লেষণ বলতে অ্যালগরিদমের দক্ষতা (Effciency) পরিমাপ বা তুলনা করা বুঝায়; এজন্য বিশেষ কিছু পদ্ধতি অনুসরণ করা হয়। মনে করা যাক, M একটি জ্যালগরিদম যার ইনপুট ডাটা সাইজ n। এটি বাস্তবায়নের জন্য প্রয়োজনীয় সময় এবং পরিসর অ্যালগরিদমটির দক্ষতা মূল্যায়নের জন্য অন্যতম দুটি পরিমাপক। একটি অ্যালগরিদমের প্রধান অপারেশন কোডসমূহ সম্পাদিত হতে যে পরিমাণ সময় লাগে (যেমন- একটি সার্চিং বা সর্টিং অ্যালগরিদমের ক্ষেত্রে লুপ সংখ্যা) তা দ্বারা অ্যালগরিদমের সময়ের পরিমাপ করা হয়। আর সেটি বাস্তবায়নের জন্য প্রয়োজনীয় সর্বোচ্চ পরিমাণ মেমোরি স্পেস দ্বারা পরিসর পরিমাপ করা হয়।

অধ্যায়-২ ঃ অ্যালগরিদম

২.৫ অ্যালগরিদমের কমপ্লেক্সসিটি
Describe the Complexity of Algorithm)

ডাটা অপারেশনের ভিন্নতার কারণে অ্যালগরিদমের কমপ্লেক্সিটি ভিন্ন ভিন্ন হয়ে থাকে।

এ ক্ষেত্রসমূহ মোটামুটি তিন শ্রেণিতে বিভক্ত।

(1) Worst case: যে কোন সম্ভাব্য Inputএর বেলায় f(n) এর মান সবোর্চ্চ অর্থাৎ Running time সবচেয়ে বেশি।

(2) Average case: f(n) এর প্রত্যাশিত মানকে Average case বলে।

(3) Best case: কখনো কখনো ডাটা প্রসেসিং এর কমপ্লেক্সিটি এতটা কম থাকে যে, f(n) এর মান সর্বনিম্ন মাত্রায় নেমে আসে। এ ক্ষেত্রকে Best case বলে।

অধ্যায়-২ ঃ অ্যালগরিদম

পদ্ধতি : লিনিয়ার সার্চ অ্যালগরিদমে সমস্যাটির সমাধানের জন্য DATA অ্যারের n সংখ্যক উপাদানের প্রত্যেকটির সাথে একটি ITEM উপাদানটি তুলনা করা হয়। অর্থাৎ, প্রথমে দেখা হয় ITEM DATA[1] কি-না, না হলে ITEM = DATA[2] কি-না, না হলে ITEM DATA[3] কি-না ইত্যাদি। এভাবে ITEM = DATA[LOC] পাওয়া গেলে কাঙ্ক্ষিত অবস্থান LOC; না পাওয়া গেলে LOC = 0 অর্থাৎ সংখ্যাটি অ্যারেতে নেই বলে ধরে নেয়া হয়।

অধ্যায়-২ ঃ অ্যালগরিদম

Algorithm: (Linear Search) A linear array DATA with N elements and a specific element ITEM are given. This algorithm finds the location LOC of the element ITEM in the array,or sets LOC = 0 if ITEM is not found on the array.

(i) Set K = 1, LOC = 0 [initialize.]

(ii) Repeat Steps 3 and 4 while LOC = 0 and KsN.

(iii) If ITEM – DATA[K], then Set LOCK [Increment Counter]

(iv) Set K = K+1 [

End of Step 2 loop.

(v) If LOC = 0. then

Write: ITEM is not found in the array DATA

Else:

Write: LOC is the location of ITEM.

[End of If structure)

(vi) Exit

অধ্যায়-২ ঃ অ্যালগরিদম

২.৬ বিভিন্ন ধরনের অ্যালগরিদম উল্লেখকরণ (Mention Different Types of Algorithm)

বিভিন্ন মানদণ্ডের উপর ভিত্তি করে অ্যালগরিদমগুলোকে বিভিন্ন প্রকারে শ্রেণিবদ্ধ করা যেতে পারে। নিচে অ্যালগরিদমের কিছু সাধারণ প্রকার রয়েছে। যেমন :

১.সর্টিং অ্যালগরিদম (Sorting Algorithms) : এই অ্যালগরিদমগুলো একটি নির্দিষ্ট ক্রমে উপাদানগুলোর একটি সংগ্রহ সাজায়, যেমন বর্ণানুক্রমিক বা সংখ্যাসূচক। উদাহরণস্বরূপ : বাবল সর্ট, কুইক সর্ট, মার্জ সর্ট এবং ইনসার্টশন সর্ট।

২. অনুসন্ধান অ্যালগরিদম (Searching Algorithms) : এই অ্যালগরিদমগুলোর লক্ষ্য উপাদানগুলোর একটি সংগ্রহের মধ্যে একটি নির্দিষ্ট উপাদান সনাক্ত করা। উদাহরণস্বরূপ : লিনিয়ার সার্চ, বাইনারি সার্চ এবং হ্যাশিং অ্যালগরিদম।

অধ্যায়-২ ঃ অ্যালগরিদম

২.৬ বিভিন্ন ধরনের অ্যালগরিদম উল্লেখকরণ (Mention Different Types of Algorithm)

৩. গ্রাফ অ্যালগরিদম (Graph Algorithms) : এই অ্যালগরিদমগুলো গ্রাফগুলোর সাথে কাজ করার জন্য ডিজাইন করা হয়েছে, যা নোড (শীর্ষ) এবং সংযোগ (প্রান্ত) নিয়ে গঠিত। গ্রাফ অ্যালগরিদম কানেক্টিভিটি, সংক্ষিপ্ততম পথ, বৃক্ষ বিস্তৃত এবং আরও অনেক কিছু সম্পর্কিত সমস্যার সমাধান করে। উদাহরণস্বরূপ : ডেপথ-ফাস্ট সার্চ (ডিএফএস), ব্রেডথ-ফার্স্ট সার্চ (বিএফএস), ডিসকস্ট্রার অ্যালগরিদম এবং প্রিমের অ্যালগরিদম।

৪. মেশিন লার্নিং অ্যালগরিদম (Machine Learning Algorithms) : এই অ্যালগরিদমগুলো ভবিষ্যদ্বাণী করতে বা প্যাটার্ন বা ডেটার উপর ভিত্তি করে পদক্ষেপ নিতে মেশিন লার্নিংয়ের ক্ষেত্রে ব্যবহার করা হয়। উদাহরণস্বরূপ : ডিসিশন ট্রি, রান্ডম ফরেস্ট, সাপোর্ট ভেক্টর মেশিন (এসভিএম), নিউরাল নেটওয়ার্ক এবং কে-মিনস ক্লাস্টারিং।

অধ্যায়-২ ঃ অ্যালগরিদম

৫. ডায়নামিক প্রোগ্রামিং অ্যালগরিদম (Dynamic Programming Algorithms) : ডায়নামিক প্রোগ্রামিং হলো একটি অ্যালগরিদমিক কৌশল যা সমস্যাগুলোকে ওভারল্যাপিং উপসমস্যাগুলোর মধ্যে ভেঙ্গে এবং এই উপসমস্যাগুলোর সমাধানগুলোকে পুনরায় ব্যবহার করে সমাধান করার জন্য ব্যবহৃত হয়।উদাহরণস্বরূপ : ফিবোনাচি সিকোয়েন্স অ্যালগরিদম, ন্যাপস্যাক সমস্যা এবং দীর্ঘতম সাধারণ পরবর্তী সমস্যা।

৬.গ্রিডি অ্যালগরিদম (Greedy Algorithms) : গ্রিডি অ্যালগরিদম বিশ্বব্যাপী সর্বোত্তম তথ্য খুঁজে পাওয়ারআশায় প্রতিটি পর্যায়ে স্থানীয়ভাবে সর্বোত্তম সার্চ করে। এই অ্যালগরিদমগুলোর লক্ষ্য ভবিষ্যতের পরিণতিবিবেচনা না করেই বর্তমান পদক্ষেপে সর্বোত্তম সমাধান খুঁজে বের করা। উদাহরণস্বরূপ : ন্যূনতম স্প্যানিং ট্রি অ্যালগরিদম (ক্রসকাল এবং প্রিমের অ্যালগরিদম) এবং সংক্ষিপ্ত পথের জন্য ডিসকস্ট্রার অ্যালগরিদম।

৭. ব্যাকট্র্যাকিং অ্যালগরিদম (Backtracking Algorithms): ব্যাকট্র্যাকিং অ্যালগরিদম ক্রমবর্ধমানভাবে প্রার্থী তৈরি করে এবং পছন্দগুলোকে পূর্বাবস্থায় নিয়ে যাওয়ার মাধ্যমে সমস্ত সম্ভাব্য সমাধান অন্বেষণ করে যদি তারা একটি ডেড-এন্ডের দিকে নিয়ে যায়।উদাহরণস্বরূপ : এন-কুইন্স সমস্যা, সুডোকু সমাধানকারী এবং ভ্রমণকারী বিক্রয়কর্মী সমস্যার সমাধান।

অধ্যায়-২ ঃ অ্যালগরিদম

৮.কম্পিউটেশনাল জ্যামিতি অ্যালগরিদম (Computational Geometry Algorithms): অ্যালগরিদম জ্যামিতিক সমস্যাগুলো যেমন ছেদ, উত্তল হলো, ত্রিভুজ এবং প্রক্সিমিটি নির্ধারণ করে। উদাহরণস্বরূপ: গ্রাহাম স্ক্যান, জার্ভিস মার্চ এবং সুইপ লাইন অ্যালগরিদম।

৯. ক্রিপ্টোগ্রাফিক অ্যালগরিদম (Cryptographic Algorithms) : ক্রিপ্টোগ্রাফিক অ্যালগরিদম যোগাযোগ, ডেটা এবং লেনদেন সুরক্ষিত করতে ব্যবহৃত হয়।উদাহরণস্বরূপ : অ্যাডভান্সড এনক্রিপশন স্ট্যান্ডার্ড (AES), RSA অ্যালগরিদম এবং MD5 এবং SHA এরমতো হ্যাশ ফাংশন।

১০. স্ট্রিং ম্যাচিং অ্যালগরিদম (String Matching Algorithms) : এই অ্যালগরিদম একটি প্রদত্ত স্ট্রিংয়েরমধ্যে প্যাটার্ন বা সাবস্ট্রিংগুলো অনুসন্ধান করে।উদাহরণস্বরূপ : Knuth-Morris-Pratt (KMP) অ্যালগরিদম, Boyer- Moore অ্যালগরিদম, এবং RabinKarp অ্যালগরিদম।এগুলো হলো অ্যালগরিদমের বিভিন্ন প্রকারের কয়েকটি উদাহরণ। অ্যালগরিদমগুলো তাদের জটিলতা, অ্যাপ্লিকেশন ডোমেন, ব্যবহৃত ডেটা স্ট্রাকচার এবং আরও অনেক কিছুর উপর ভিত্তি করে আরও শ্রেণিবদ্ধ করা যেতে পারে।

অধ্যায়-৩ ঃ অ্যারে,পয়েন্টার এবং স্ট্রিং

৩.১ অ্যারে,পয়েন্টার এবং স্ট্রিং এর সঙ্গা:

অ্যারে:
সবচেয়ে সরল ডাটা স্ট্রাকচার হলো অ্যারে। অ্যারে শব্দের আভিধানিক অর্থ হলো শ্রেণিবদ্ধ সজ্জা। সাধারণত এক ধরনের নির্দিষ্ট সংখ্যক ডাটা উপাদানের শ্রেণিবদ্ধ সজ্জাকে অ্যারে বলে।

অ্যারের একটি নাম দেয়া হয় এবং একটি অ্যারে কয়টি উপাদান নিয়ে গঠিত তা প্রকাশের জন্য নামের সাথে 1,2,3.....n ইত্যাদি সাবক্রিপ্ট বা ইনডেক্স ব্যবহার করা হয়।

অধ্যায়-৩ ঃ অ্যারে,পয়েন্টার এবং স্ট্রিং

৩.১ অ্যারে,পয়েন্টার এবং স্ট্রিং এর সঙ্গা:

পয়েন্টার:
পয়েন্টার হলো একটি ভেরিয়েবল, যা অন্য ভেরিয়েবলের মেমরি লোকেশন সংঙ্গায়িত করে। অর্থাৎ পয়েন্টার একটি মেমরি ঠিকানা নিয়ে রাখে এবং সে ঠিকানায় অন্য ভেরিয়েবলের মান এক্সেস করতে ব্যবহার করা হয়।

অধ্যায়-৩ ঃ অ্যারে,পয়েন্টার এবং স্ট্রিং

৩.১ অ্যারে,পয়েন্টার এবং স্ট্রিং এর সঙ্গা:

স্ট্রিং:
প্রোগ্রামিং ভাষায় সিঙ্গেল কোটেশন বা ডাবল কোটেশনের ভেতরে যা থাকে তাকে স্ট্রিং বলা হয়।

Char str[]=“Hello World”;

অধ্যায়-৩ ঃ অ্যারে,পয়েন্টার এবং স্ট্রিং

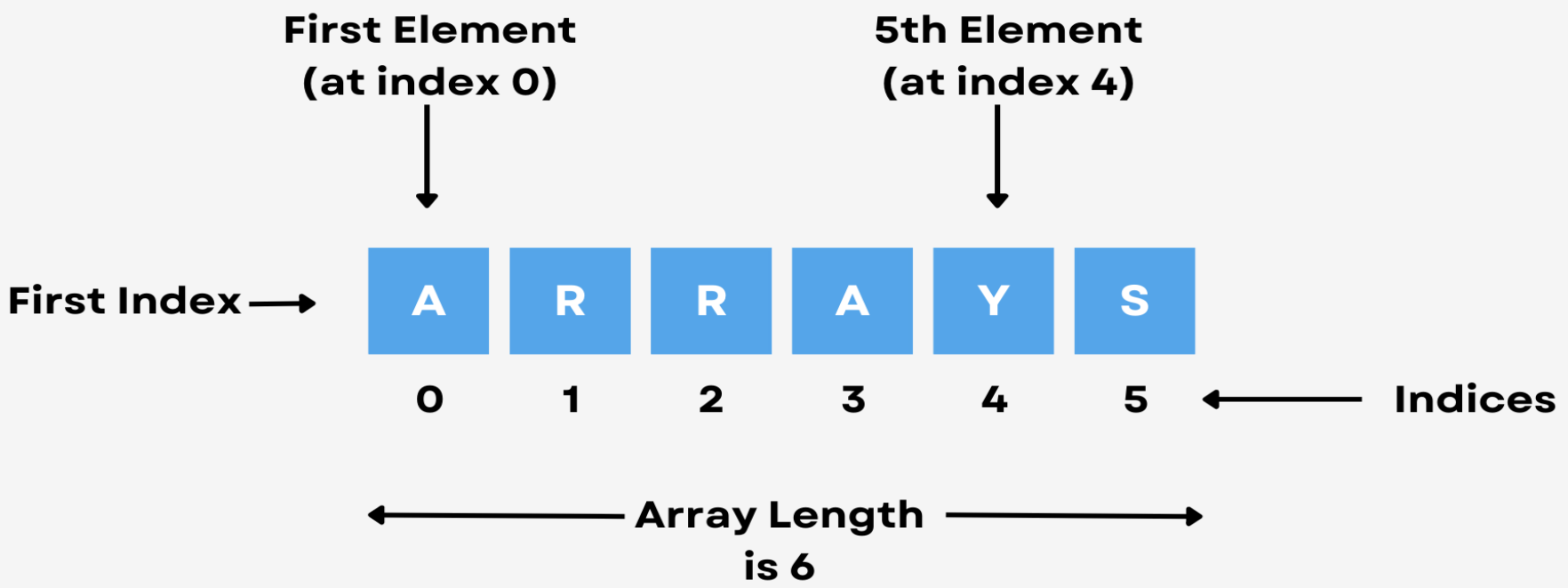
৩.২ ডায়াগ্রাম সহকারে বিভিন্ন ডায়মেনশনাল অ্যারে।

অ্যারেকে সাধারণত দু ভাগে ভাগ করা যায়। যথা:
১। এক মাত্রিক বা লিনিয়ার অ্যারে
২। বহুমাত্রিক বা নন-লিনিয়ার অ্যারে

আবার বহুমাত্রিক বা নন-লিনিয়ার অ্যারে দু প্রকার। যথা-
১। দ্বিমাত্রিক অ্যারে
২। ত্রিমাত্রিক অ্যারে

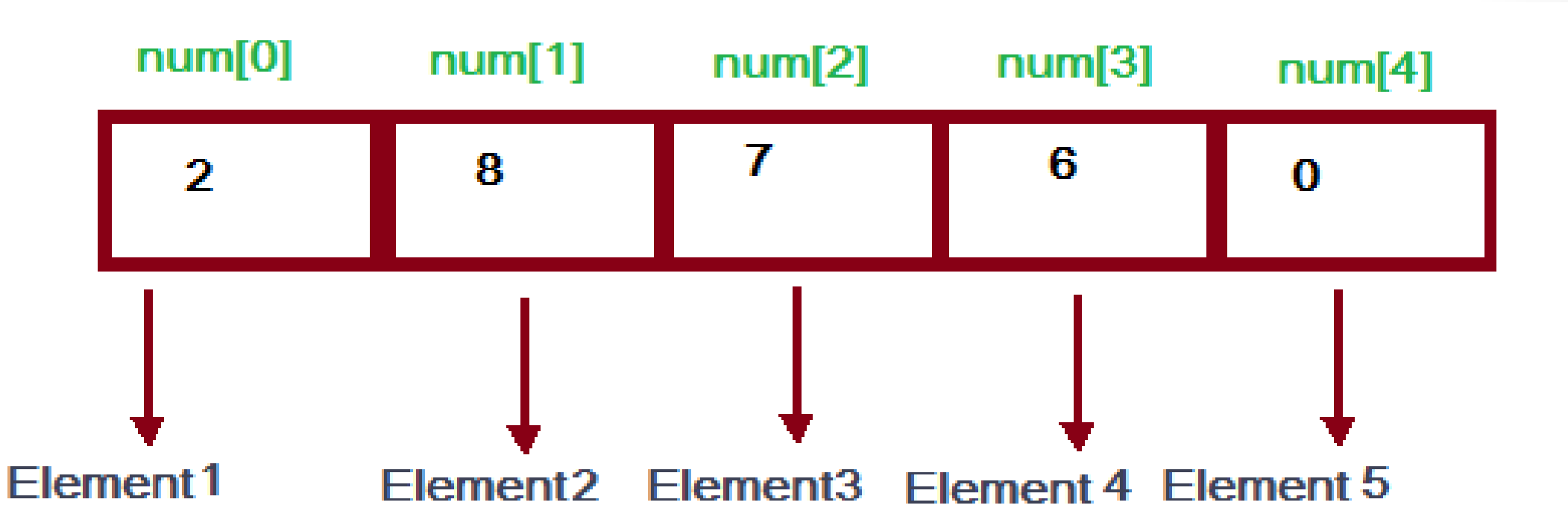
অধ্যায়-৩ ঃ অ্যারে,পয়েন্টার এবং স্ট্রিং

এক মাত্রার অ্যারে বা লিনিয়ার অ্যারে:



অধ্যায়-৩ ঃ অ্যারে,পয়েন্টার এবং স্ট্রিং

এক মাত্রার অ্যারে বা লিনিয়ার অ্যারে:



অধ্যায়-৩ ঃ অ্যারে,পয়েন্টার এবং স্ট্রিং

Algorithm: **TRAVERSING LINEAR ARRAYS**

1. [Initialize counter.] Set K:= LB.

2. Repeat Steps 3 and 4 while K<=UB.

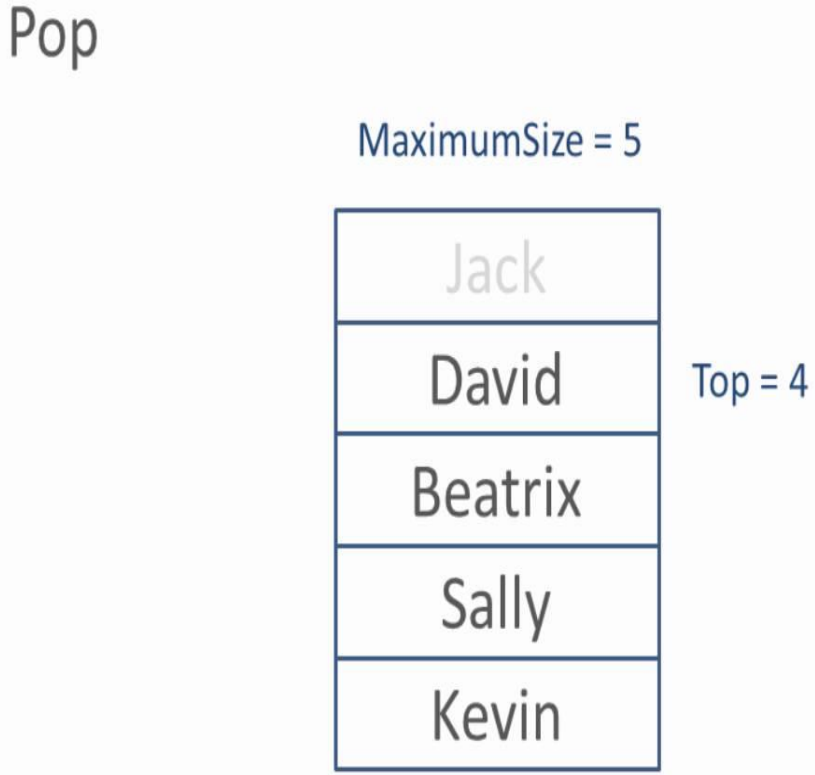
3. [Visit element.] Apply PROCESS to LA[K].

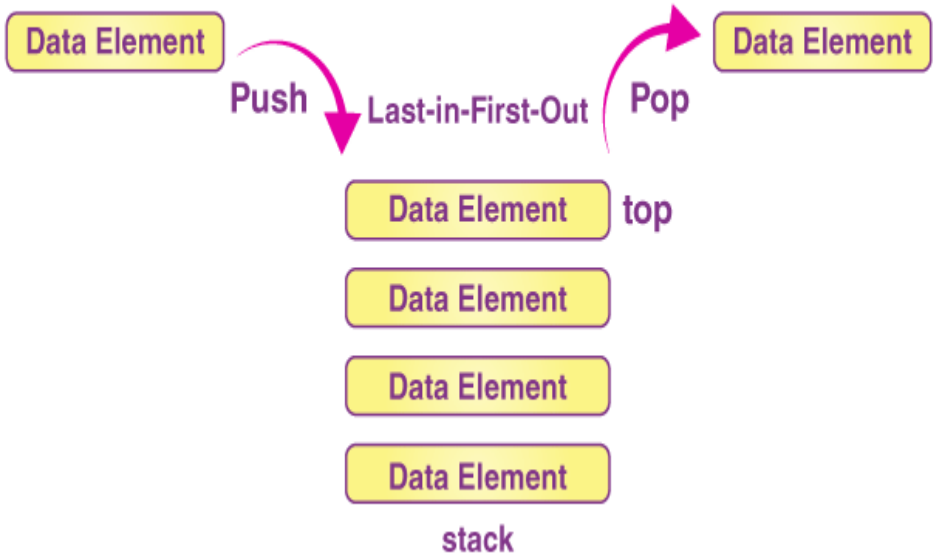
4. [Increase counter.] Set K:=K+1.

[End of step 2 loop.]

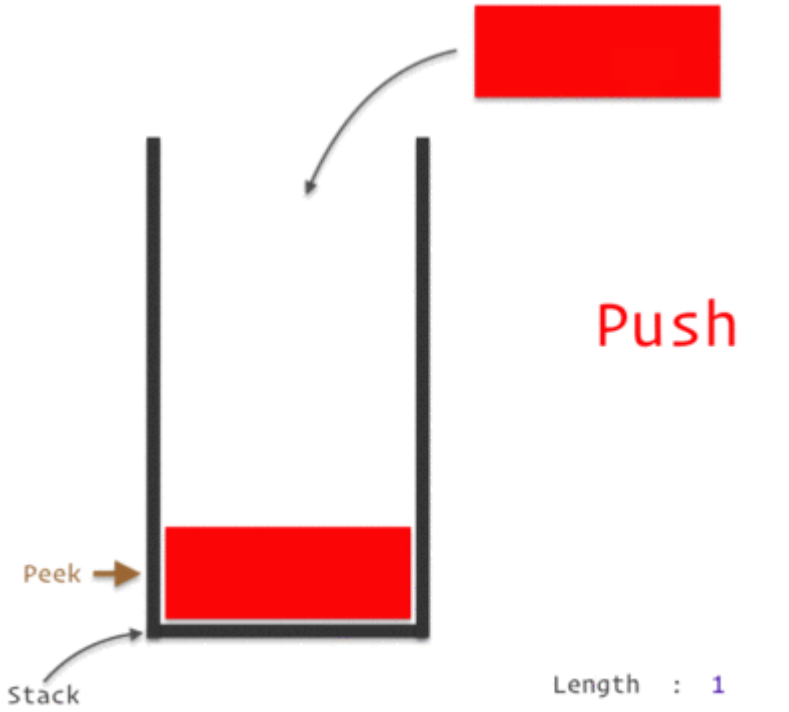
5. Exit

অধ্যায়-৪ ঃ স্ট্যাক ডাটা স্ট্রাকচার



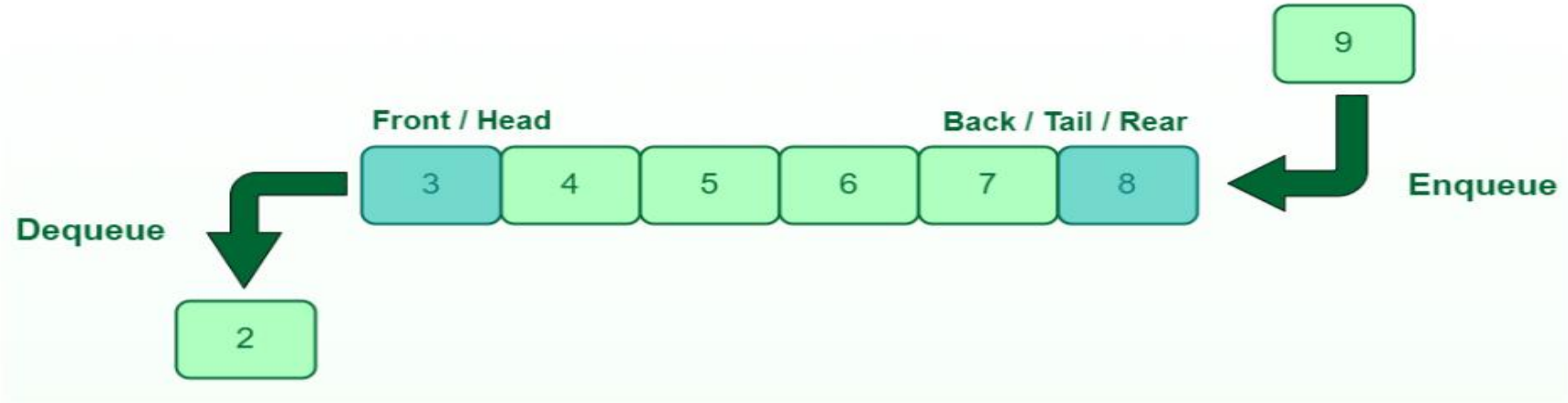


অধ্যায়-৪ ঃ স্ট্যাক ডাটা স্ট্রাকচার



অধ্যায়-৫ ঃ কিউ ডাটা স্ট্রাকচার

কিউ(Queue): কিউ মূলত একটি লিনিয়ার ডাটা স্টাকচার। যার এক প্রান্ত থেকে ডাটা বিয়োজন করা যায়,তাকে ফ্রন্ট(Front) বলা হয় এবং অন্য প্রান্তে ডাটা সংযোজন করা যায় তাকে রেয়ার(Rear) বলা হয়।



Queue Data Structure

অধ্যায়-৫ ঃ কিউ ডাটা স্ট্রাকচার

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

অধ্যায়-৬ ঃ লিংকড লিস্ট

৬.১ লিংকড লিস্ট এর সঙ্গা:

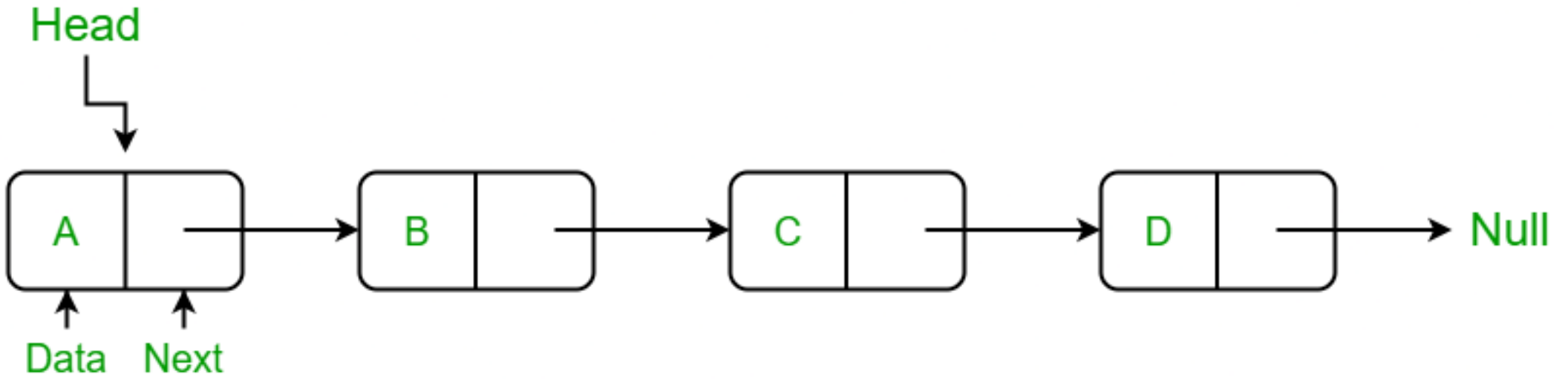
লিঙ্কড লিস্ট (linked list) হচ্ছে একটি লিনিয়ার অ্যারেতে উপাদানের সন্নিবেশ। অ্যারের এক একটি ইলিমেন্ট কে বলা হয় নোড।

প্রত্যেক নোডে দুইটা অংশ থাকে; প্রথম অংশে ডাটা এবং দ্বিতীয় অংশে এর পরের নোডের একটা রেফারেন্স থাকে যার মাধ্যমে পরের নোডের সাথে এই নোডের একটা লিংক হয়।

ডাটা অংশ: যা উপাদানের তথ্য সংরক্ষণ করে,তাকে ইনফরমেশন অংশ বলা হয়।

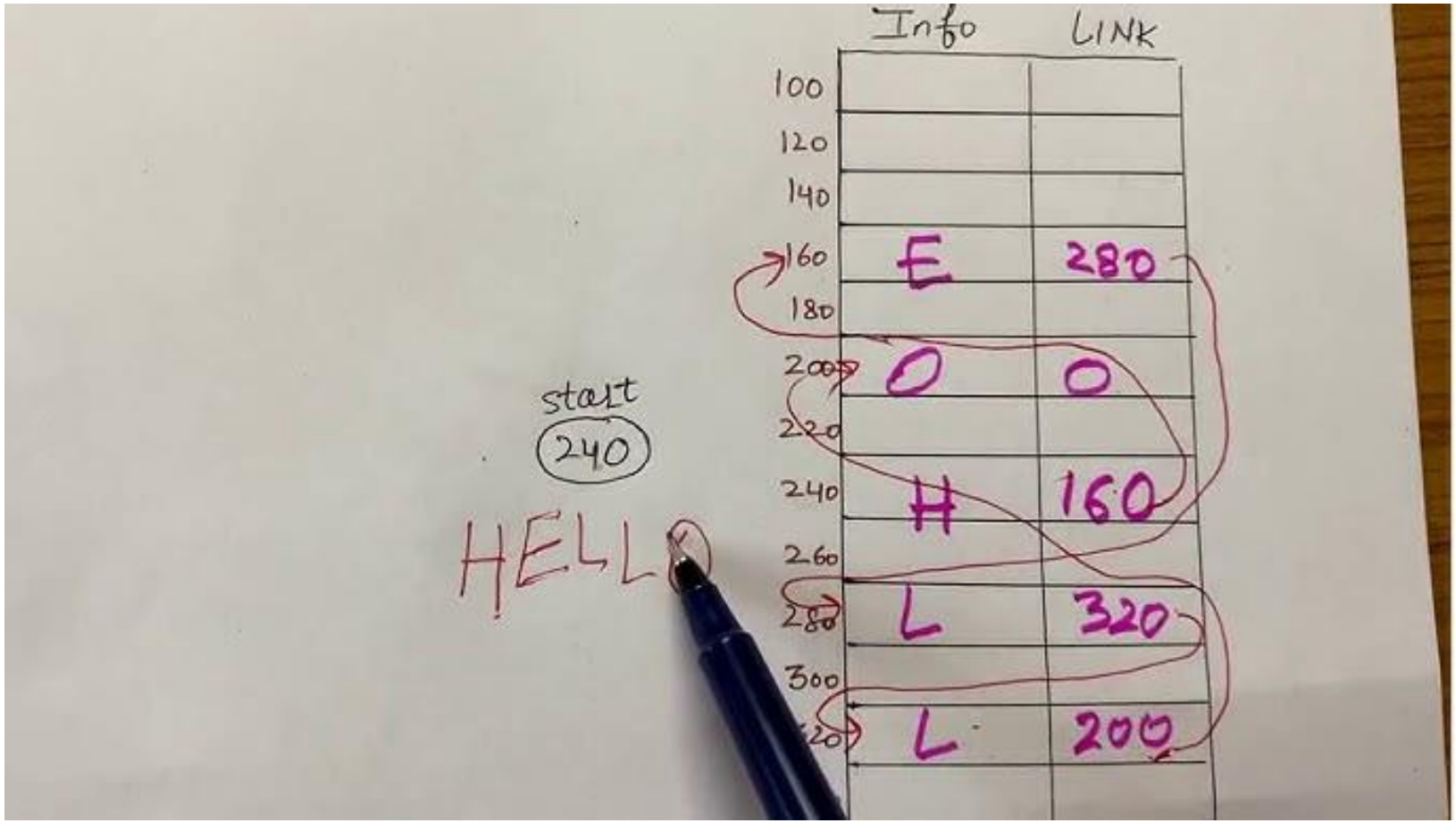
অ্যাড্রেস অংশ: যা পরবর্তী নোডের অ্যাড্রেস সংরক্ষণ করে।

অধ্যায়-৬ ঃ লিংকড লিস্ট



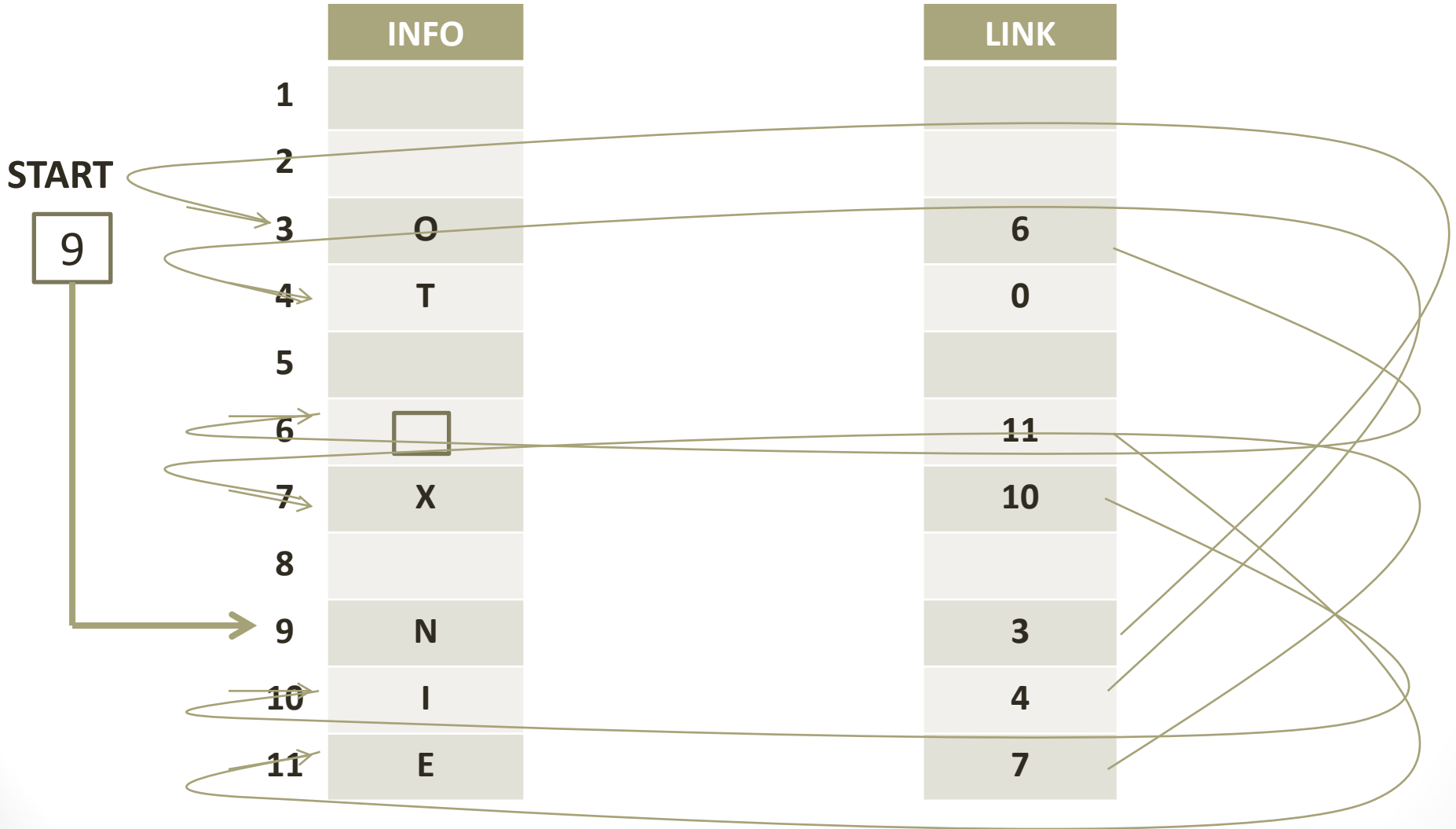
অধ্যায়-৬ ঃ লিংকড লিস্ট

৬.২ লিংকড লিস্ট এর মেমোরি অ্যালোকেশন পদ্ধতি:



অধ্যায়-৬ ঃ লিংকড লিস্ট

৬.২ লিংকড লিস্ট এর মেমোরি অ্যালোকেশন পদ্ধতি:



অধ্যায়-৬ ঃ লিংকড লিস্ট

মনে করি List একটি লিংকড লিস্ট। মেমোরিতে Linked List দুটি Linear array এর মাধ্যমে সজ্জিত থাকে। একটি INFO ও অন্যটি LINK ,যেখানে-

(i) INFO[K] ⟶ List এর Information Part নির্দেশ করে।

(ii) LINK[K] ⟶ Next Pointer,যা পরবর্তী node কে নির্দেশ করে।

List নামক Linked List এর প্রথম উপাদানের লোকেশনকে নির্দেশ করার জন্য START নামের একটি Variable এবং লিস্ট এর সমাপ্তি নির্দেশ করার জন্য Next Pointer NULL নির্দেশ করবে।

অধ্যায়-৬ ঃ লিংকড লিস্ট

START=9, অতএব INFO[9]=N,যা নোডের প্রথম Character
LINK[9]=3,অতএব INFO[3]=0, যা নোডের দ্বিতীয় Character
LINK [3]=6, অতএব INFO[6]= □ (Blank)যা নোডের তৃতীয় Character
LINK [6]=11,অতএব INFO[11]= E,যা নোডের চতুর্থ Character
LINK [11]=7,অতএব INFO[7]= X,যা নোডের 5th Character
LINK [7]=10,অতএব INFO[10]= I,যা নোডের 6th Character
LINK[10]=4,অতএব INFO[4]= I,যা নোডের 7th Character
LINK[4]=0,NULL Value, যা List এর সমাপ্তি নির্দেশ করে।

উপরোক্ত বর্ণনানুযায়ী এটা সহজে প্রতীয়মান হয় যে, চিত্রে প্রদর্শিত ক্রমানুযায়ী INFO অ্যারে লিস্টের বিভিন্ন Character ধারণ করে এবং LINK অ্যারে পরবর্তী নোডের Address নির্দেশ করে।

অধ্যায়-৬ ঃ লিংকড লিস্ট

৬.৩ লিংকড লিস্ট এর ট্রাভার্সিং অ্যালগরিদম

Algorithm:(Traversing in a Linked List)

এই Algorithm মেমোরিতে সংরক্ষিত (Linked List)List এর প্রতি নোড (Node)PROCESS অপারেশন প্রয়োগের মাধ্যমে ট্রাভার্স (Traverse)করে। এ মুহূর্তে প্রসেস করতে হবে যে নোডটি তাকে ভেরিয়েবল PTR নির্দেশ করে।

ধাপ-১ : PTR এর শুরুর Value কে Initialization করে দিতে হবে।

Set PTR:=Start

ধাপ-২: While PTR= NULLনা হওয়া পর্যন্ত ধাপ-৩ এবং ধাপ-৪ কাজ করবে।

ধাপ-৩: Infor[PTR] প্রক্রিয়াটি Exrcute করবে।

ধাপ-৪: পরবর্তী PTR Point করবে।

Set PTR:=Link[PTR]

পুনরায় ধাপ-২ তে ফিরে যাবে।

ধাপ-৫: প্রোগ্রামটি শেষ হবে।

অধ্যায়-৬ ঃ লিংকড লিস্ট

৬.৪ লিংকড লিস্ট এ সার্চিং অ্যালগরিদম

Algorithm: SEARCH(INFO,LINK,START,ITEM,LOC)

এই Algorithm মেমোরিতে সংরক্ষিত (Linked List)List থেকে নোড (Node) এর লোকেশন Loc খুঁজে বের করে, যেখানে List এ প্রথম ITEM টি প্রথম দৃষ্টিগোচর হয় অথবা LOC=NULL নির্ধারণ করে।

Step-1: Set PTR:=Start

Step-2: Repeat Step 3 While PTR≠ NULL

Step-3: If ITEM=INFO[PTR],then:

Set LOC:=PTR, and Exit

Else:

Set PTR:=LINK[PTR]:[PTR now points to the next node.]

[End of if structure.]

[End of step 2 loop.]

Step-4: [Search is successfull.]

Set LOC:=NULL.

Step-5: Exit

অধ্যায়-৬ ঃ লিংকড লিস্ট

৬.৫ লিংকড লিস্ট এ ডাটা সংযোজন, বিয়োজন অ্যালগরিদম

অ্যালগরিদমের সাহায্যে ভিন্ন ভিন্ন অবস্থানে লিংকড লিস্ট এ নতুন নোড সংযোজন করা যায়। সাধারণত তিনটি উপায়ে লিংকড লিস্ট এ নোড সংযোজন করা যায়।

(ক) লিস্টের শুরুতে নতুন নোড সংযোজন করা যায়।
(খ) একটি নির্দিষ্ট লোকেশনে নতুন একটি নোড সংযোজন করা যায়।
(গ) লিস্টের শেষে নতুন নোড সংযোজন করা যায়।

অধ্যায়-৬ ঃ লিংকড লিস্ট

৬.৫ লিংকড লিস্ট এ ডাটা সংযোজন অ্যালগরিদম

Algorithm: INSFIRST(INFO,LINK,START,AVAIL,ITEM)

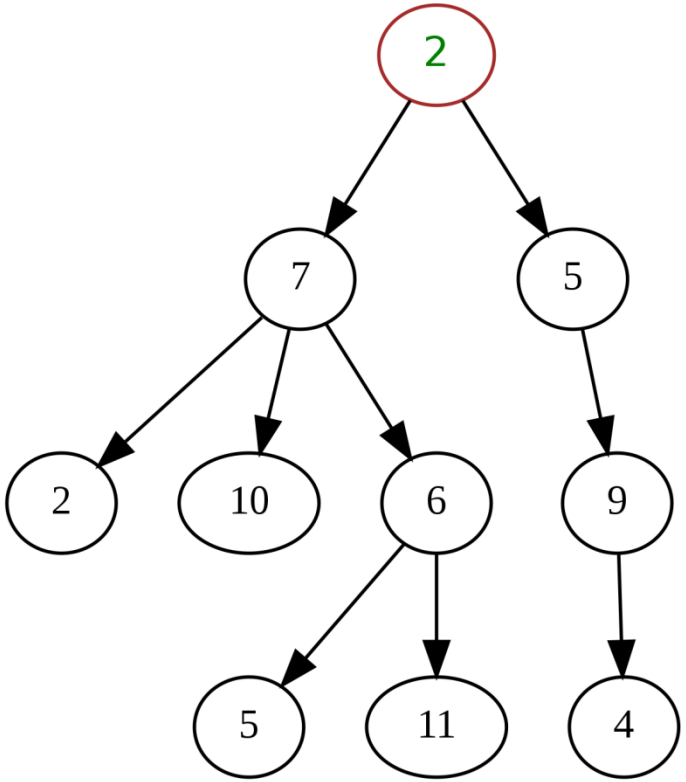
এই Algorithm (Linked List)List এ প্রথম নোড (Node) হিসেবে ITEM Insert করে।

Step-1: [OVERFLOW?] If AVAIL-NULL, then: Write: OVERFLOW, and Exit.
Step-2: [Remove first node fromAVAIL list.]
Set NEW:=AVAIL and AVAIL:=LNK[AVAIL].
Step-3: Set INFO[NEW]:=ITEM.[Copies new data into new node.]
Step-4: Set LINK[NEW]:=START.[New node now points to original first node.]
Step-5: Set START:=NEW.[Changes START so it points to the new nlode.]
Step-6: Exit

অধ্যায়-৭ : ট্রি

৭.১ ট্রি এর সঙ্গা:

ট্রি হচ্ছে কিছু নোডের সমন্বয়ে গঠিত একটা নন-লিনিয়ার এবং Hierarchical Data Structure. যেখানে নোডগুলো একে অপরের সাথে যুক্ত থাকবে কিন্তু কোনো সাইকেল তৈরি করবে না।

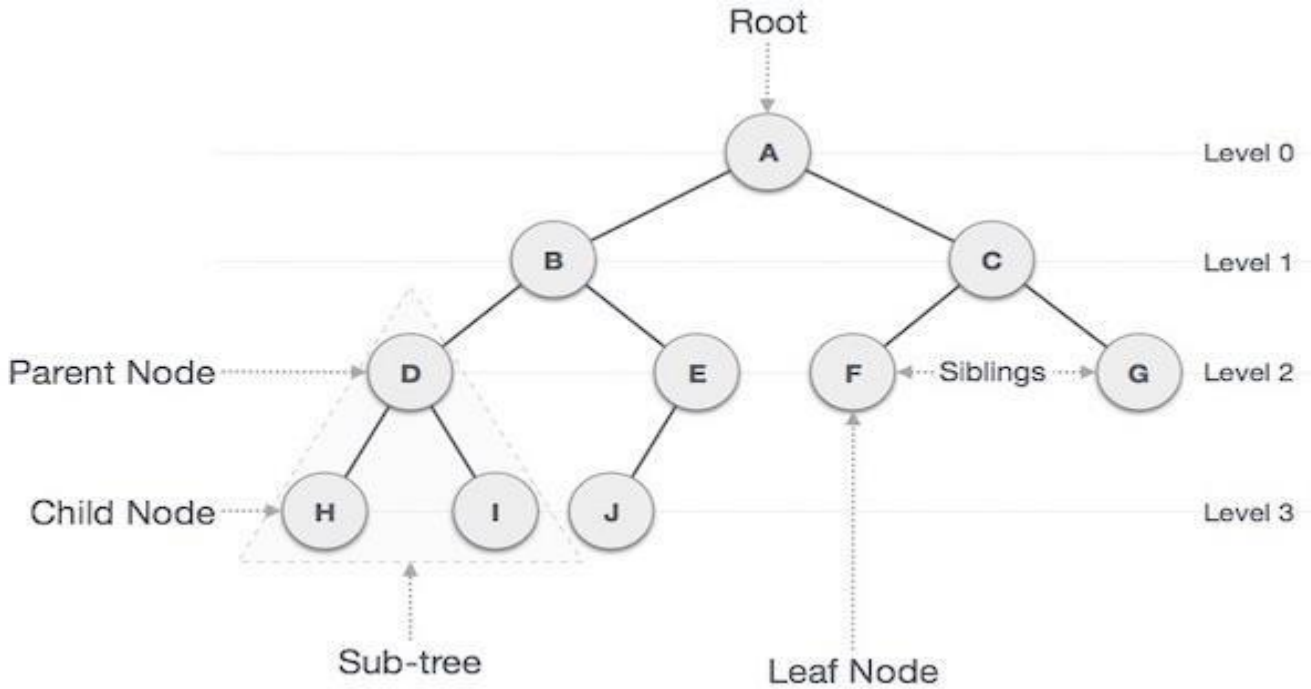


অধ্যায়-৭ : ট্রি

৭.২ ট্রি এর রুট,নোড,লীফ,কীস, সাব ট্রি এবং লেভেল

ট্রি এক ধরনের নন-লিনিয়ার Data Structure. অনেক সময় ডাটার বিভিন্ন উপাদানে মধ্যে ঊর্ধক্রম বা Hierarchical relationship বিদ্যমান থাকে। এখানে Hierarchical বলতে একটি ডাটার অধীনে সম্পর্কযুক্ত একাধিক ডাটা উপাদান কে বুঝানো হয়েছে। যে ডাটা স্ট্রাকচারের মধ্যে এ ধরনের সম্পর্কের সুস্পষ্ট প্রতিফলন ঘটে তাকে ট্রি বলে।

ট্রি এর টারমিনোলজিগুলো ব্যাখা করার জন্য নিচের চিত্রটি লক্ষ করি-



অধ্যায়-৭ ঃ ট্রি

৭.২ ট্রি এর রুট,নোড,লীফ,কীস, সাব ট্রি এবং লেভেল

নোড(Node): নোড হলো কোন ট্রি এর প্রতিটি উপাদান, যা ইনফরমেশন ধারণ করে। A,B,C,D,E,F,G,H,I,J হলো এক একটি নোড।

রুট(Root): একটি ট্রি এর একদম টপ নোড কে বলা হয় রুট। রুট নোড কে অন্য কোন নোড পয়েন্ট করে না। চিত্রে A হলো রুট নোড।

Child Node: একটা ট্রিতে রুট ছাড়া বাকি যে নোডগুলো থাকে, সেগুলোকে Child Nodeবলে। চিত্রে রুট নোড ব্যতীত বাকি B,C,D,E,F,G,H,I,J হলো চাইল্ড নোড।

Parent Node: কোন একটা নোডের যদি এক বা একাধিক চাইল্ড থাকে, তাহলে তাকে বলা হয় Parent Node. চিত্রে A,B,C,D,E হলো Parent Node .

অধ্যায়-৭ ঃ ট্রি

৭.২ ট্রি এর রুট,নোড,লীফ,কীস, সাব ট্রি এবং লেভেল

Siblings: যে নোডগুলোর Parent একই তাদের কে বলে Siblings. চিত্রে F ও G হলো পরস্পর Siblings.

Edge: যে কানেকশন বা লিংকের মাধ্যমে একটা নোড আরেকটা নোডের সাথে যুক্ত থাকে তাকেই Edge বলে।

Leaf: যে নোডের কোন Child Node নেই, তাকে Leaf Node বলে। একে External Node ও বলা হয়। চিত্রে F,G,H,I,J হলো Leaf Node .

Branch : যে নোডের অন্তত একটা Child আছে সেটাই একটা Branch . চিত্রে B,C,D,E হলো Branch.

Subtree: যদি একটি ট্রি এর Left সাইড ও Right সাইডে একটি নোডের অধীনে অধিক নোড থাকে তবে তাদেরকে একত্রে Subtree বলে। চিত্রে B ও D, A এর সাবট্রি এর রুট।

অধ্যায়-৭ : ট্রি

৭.২ ট্রি এর রুট,নোড,লীফ,কীস, সাব ট্রি এবং লেভেল

Depth: রুট থেকে কোন একটা নোডে পৌঁছানোর জন্য Edge এর সংখ্যাই ঐ নোডের Depth. চিত্রে D এর Depth হচ্ছে ২।

Level : কোন একটা নোডের Level হচ্ছে রুট থেকে ঐ নোডে পৌঁছানোর Edge এর সংখ্যার চেয়ে ১ বেশি। সংক্ষেপে Level=Depth+1

অধ্যায়-৭ : ট্রি

৭.৩ চিত্র সহকারে বিভিন্ন প্রকার ট্রি ডাটা স্ট্রাকচারের সঙ্গা

ডাটা স্ট্রাকচারে বিভিন্ন প্রকারের ট্রি স্ট্রাকচার ব্যবহার হয়ে আসছে। এসব ট্রি এর মধ্যে M-Array Tree, Full M-Array Tree,Binary Tree,Complete Binary Tree,Binary Search Tree, (BST),Spanning Tree,Heap Tree ইত্যাদি উল্লেখযোগ্য।

M-Array Tree:

অধ্যায়-৮ সার্চিং অপারেশন

৮.১ সার্চিং অপারেশন-এর বিভিন্ন কৌশল (State different techniques of searching) & উপাদানসমূহ হতে কোন নির্দিষ্ট মানেরSearch (সার্চ) শব্দের আভিধানিক অর্থ হলো অন্বেষণ বা খোঁজ করা।
অর্থাৎ একাধিক উপাদানকে খুঁজে বের করার প্রক্রিয়াকে Searching বলে।

ITEM পাওয়া বা না পাওয়ার ওপর ভিত্তি করে Search-কে ২ ভাগে ভাগ করা যায়—
(i) Successful Search (সফল সার্চ)
(ii) Unsuccessful Search (বিফল সার্চ)।

Algorithm বিভিন্ন ধরনের হতে পারে। Search Algorithm ডাটা স্ট্রাকচারের ওপর নির্ভরশীল। বিভিন্ন ধরনের Searching থাকা সত্ত্বেও এ অধ্যায়ে শুধুমাত্র দুই ধরনের সার্চ পদ্ধতি সম্বন্ধে আলোচনা করা হয়েছে, যথা-

(i) Linear Search (লিনিয়ার সার্চ)
(ii) Binary Search (বাইনারি সার্চ)

অধ্যায়-৮ সার্চিং অপারেশন

৮.২ লিনিয়ার এবং বাইনারি সার্চ-এর বর্ণনা (Explanation of linear and binary search) :

লিনিয়ার সার্চঃ

Linear searching এমন এক ধরনের searching পদ্ধতি, যার মাধ্যমে Sorted বা Unsorted উভয় ধরনে Data সমাবেশ থেকে নির্দিষ্ট ডাটা Search করা সম্ভব। একটি Linear Array থেকে প্রতিটি উপাদানের সাথে পর্যায়ক্রমে তুলনা কর কোন নির্দিষ্ট তথ্য বা ITEM কে খুঁজে বের করার পদ্ধতিকে Linear search বলে।

অধ্যায়-৮ সার্চিং অপারেশন

বাইনারি সার্চ (Binary search) :

বাইনারি সার্চ অন্যান্য সার্চের তুলনায় অনেক কম সময়ে সার্চ প্রক্রিয়া সম্পন্ন করে ধারএটি একটি খুবই গুরুত্বপূর্ণ সার্চ পদ্ধতি। বাইনারি সার্চ মূলত Sorted ডাটার ক্ষেত্রে প্রয়োগ করা হয়। বাইনারি সার্চ পদ্ধতির একটি উদাহরণ দেয়া যাক। টেলিফোন ডাইরেকটরি বা Dictionary থেকে কোন ব্যক্তির নাম, টেলিফোন নাম্বার খুঁজে বের করা যায়। প্রথমে ডিকশনারির মাঝখানে খোলা হয়। এরপর কাঙ্ক্ষিত শব্দটির সাথে তুলনা করা হয়। যদি কাঙ্ক্ষিত শব্দ মাঝখানের শব্দ থেকে ছোট হয়, তবে ডিকশনারির বামদিকে অথবা যদি বড় হয়, তবে ডানদিকের অংশের অর্ধেক করা হয়। এছা প্রতিবার অর্ধেক করে খোলা হয় এবং প্রয়োজনীয় শব্দের সাথে তুলনা করে দেখা হয়। একই প্রক্রিয়া পুনরাবৃত্তি করে কাঙ্ক্ষিত শব্দ বের করা হয়। বাইনারি সার্চেও একই পদ্ধতি অবলম্বন করা হয়ে থাকে। লিনিয়ার সার্চ পদ্ধতির ন্যায় প্রতিটি উপাদানের সাথে তুলনা করার প্রয়োজন হয় না। এজন্য Binary search পদ্ধতি খুবই দ্রুততম সার্চ পদ্ধতি।

অধ্যায়-৮ সার্চিং অপারেশন

৮.৩ লিনিয়ার সার্চ অ্যালগরিদম এর ব্যাখ্যা (Explanation of algorithms for linear & binary search):

Algorithm (Linear search) : Linear search (Array N, item)এখানে DATA হচ্ছে N উপাদানবিশিষ্ট একটি Linear Array এবং ITEM হচ্ছে নির্দিষ্ট তথ্য উপাদান। এই Algorithm ডাটা অ্যারের মধ্যে ITEM-এর অবস্থান LOC খুঁজে বের করে, অথবা সার্চ বিফল হলে LOC : = 0 নির্ধারণ করে।

Step-1 : Set i to 1
Step-2: if i > n then go to step-7
Step-3: if Array [i] = item then go to Step-6:
Step-4: Set is to i + 1
Step-5 : Go to step 2
Step-6: Print element item found at index location i and go to step-8
Step-7: Print element not found
Step-8: Exit.

৮.৪ বাইনারি সার্চ অ্যালগরিদম এর ব্যাখ্যা (Explanation of binary search algorithm) :

Algorithm (Binary search) BINARY (DATA, LB, UB, ITEM, LOC)এখানে DATA একটি Sorted Array, যার লোয়ার বাউন্ড (Lower bound) LB. আপার বাউন্ড (Upper bound) UB ITEM হচ্ছে নির্দিষ্ট তথ্য উপাদান । BEG, END এবং MID Variable তিনটি যথাক্রমে Array উপাদানের প্রতি সেগমেন্টের প্রথশেষ ও মধ্যবর্তী অবস্থান নির্দেশ করে। এই Algorithm-টি DATA থেকে ITEM-এর অবস্থান LOC নির্ণয় করে। অথবা LOCNULL নির্ধারণ করে।

```
1.      [Initialize segment variables.]
        Set BEG= LB, END = UB and POS = -1
2.      Repeat Steps 3 and 4 while BEG<=END
Step-3: Set MID = (BEG+ END)/2
Step-4: IF DATA [MID] = VAL
        SET POS MID
        PRINT POS
        Go to step 6
        ELSE IF DATA [MID] > VAL
        SET END = MID-1
        ELSESET BEG MID + 1
        [End of Loop]
Step-5: IF POS=-1
        Print "Value is not present in the array“
        [End of IF]
Step-6: Exit.
```

৮.৫ লিনিয়ার সার্চ ও বাইনারি সার্চ অ্যালগরিদমের জটিলতার তুলনা (Compare the complexity of & binary search algorithms) :

লিনিয়ার সার্চ অ্যালগরিদমের Complexity (Complexities of linear search algorithm) : লিনিয়ার সার্চ অ্যালগরিদমের • Complexity-কে N সংখ্যক Element বিশিষ্ট DATA Array থেকে ITEM-এর স্থান শনাক্ত করতে যতগুলো তুলনার প্রয়োজন হয়, তার সংখ্যা f(n) দ্বারা পরিমাপ করা হয়। নিম্নে Worst case (অতিমন্দ ক্ষেত্র) এবং Average case (গড় ক্ষেত্র) জটিলতা(Complexity) আলোচনা করা হলো-

(a) Worst Case (অতিমন্দ ক্ষেত্র) : যদি নির্দিষ্ট ITEM-টি DATA নামক অ্যারের শেষ উপাদান হয় অথবা ITEM টি যদি DATA নামক অ্যারেতে অবস্থান না করে, তাহলে সর্বোচ্চ সংখ্যক তুলনার প্রয়োজন হয়।
(b) Average Case Complexity (গড় ক্ষেত্র) : Average Case এ Complexity সম্ভাব্যতার (Probability) উপর ভিত্তি করে নির্ণয় করা হয়।

ধরা যাক, DATA [K]-এর মধ্যে ITEM উপাদানটি থাকার Probability Pe এবং DATA [K] এর মধ্যে ITEM উপাদানটি নাথাকার Probability q. তাহলে-

P1 + P2 + Ps ..... + Pn + q = 1 [Probability-এর সূত্রানুসারে]

যেহেতু DATA[K] এর মধ্যে ITEM থাকলে Algorithm k সংখ্যক তুলনা করে।

অধ্যায়-৮ সার্চিং অপারেশন

লিনিয়ার সার্চ ও বাইনারি সার্চের পার্থক্য (Difference between linear search and binary search) ঃ

লিনিয়ার সার্চ ও বাইনারি সার্চের পার্থক্য নিম্নে দেওয়া হলো–

ক্রমিক নং	লিনিয়ার সার্চ অ্যালগরিদম	বাইনারি সার্চ অ্যালগরিদম
(i)	Sorted বা Unsorted যে কোন data series থেকে ITEM search করা সম্ভব।	শুধুমাত্র Sorted ডাটা থেকে ITEM search করা সম্ভব।
(ii)	ডাটা Searching-এ একদিক থেকে পর্যায়ক্রমে ডাটা তুলনা শুরু করে।	সমগ্র ডাটাগুলোকে দুইভাগে ভাগ করে ঠিক মাঝখান থেকে ডাটা তুলনা শুরু করে।
(iii)	Worst Case Complexity $f(n) = n + 1$	Worst Case Complexity $f(n) = \log_2^n + 1$
(iv)	Average Case Complexity $f(n) = \frac{n+1}{2}$	Worst Case Complexity এবং Average Case complexity একই থাকে।
(v)	Algorithm-টি Run করতে সর্বাধিক n এর সমানুপাতিক সময় প্রয়োজন হয়।	Algorithm-টি Run করতে সর্বাধিক $\log_2^n$ এর সমানুপাতিক সময় প্রয়োজন হয়।

অধ্যায়-৯ ঃ সর্টিং অপারেশন

একটি Data structure-কে পছন্দমতো অ্যারেতে অক্ষর বা সংখ্যামানের ক্রমানুসারে ডাটা উপাদান সাজানোর প্রক্রিয়াকে Sorting বলে। Data structure থেকে দ্রুত ও সহজে প্রয়োজনীয় ডাটা বের করে নেয়ার জন্য Sorting ব্যবহৃত হয়। ডাটা Analysis (বিশ্লেষণ) বা পরিসংখ্যানগত যে-কোন কাজে Unsorted data-এর পরিবর্তে Sorted ডাটা নিয়ে কাজ করলে অনেক সময় বেঁচে যায়।

Sorting-কে দুই ভাগে ভাগ করা যায়—

(i) Ascending order (ছোট থেকে বড় সাজানো)
(ii) Descending order (বড় থেকে ছোট সাজানো)।

সাজানো Data থেকে যে কোন Data অনুসন্ধান করা সহজ হয়ে থাকে। বিভিন্ন রকম গবেষণার ক্ষেত্রে অনেক সময় Sorted ডাটার প্রয়োজন হয়ে থাকে। কম্পিউটারের মেমরিতে Stored (সংরক্ষিত) ডাটা বা উপাদানসমূহ number বা Character হতে পারে।

অধ্যায়-৯ ঃ সর্টিং অপারেশন

৯.১ সর্টিং-এর বিভিন্ন কৌশল (List the sorting techniques)

বিভিন্ন Sorting পদ্ধতির মধ্যে উল্লেখযোগ্য কয়েকটি হচ্ছে-

(i) Bubble sort (বাবল সর্ট)
(ii) Quick sort (কুইক সর্ট)
(iii) Heap sort (হীপ সর্ট)
(iv) Insertion sort (ইনসার্শন সর্ট
(v) Selection sort (সিলেকশন সর্ট)
vi) Merge sort (মার্জ্র সর্ট),
(vii) Radix sort (রেডিক্স সর্ট)
(viii) Shell sort (শেল সর্ট)
(ix) Counting sort (কাউন্টিং সর্ট)
(x) Bucket sort (বাকেট সর্ট

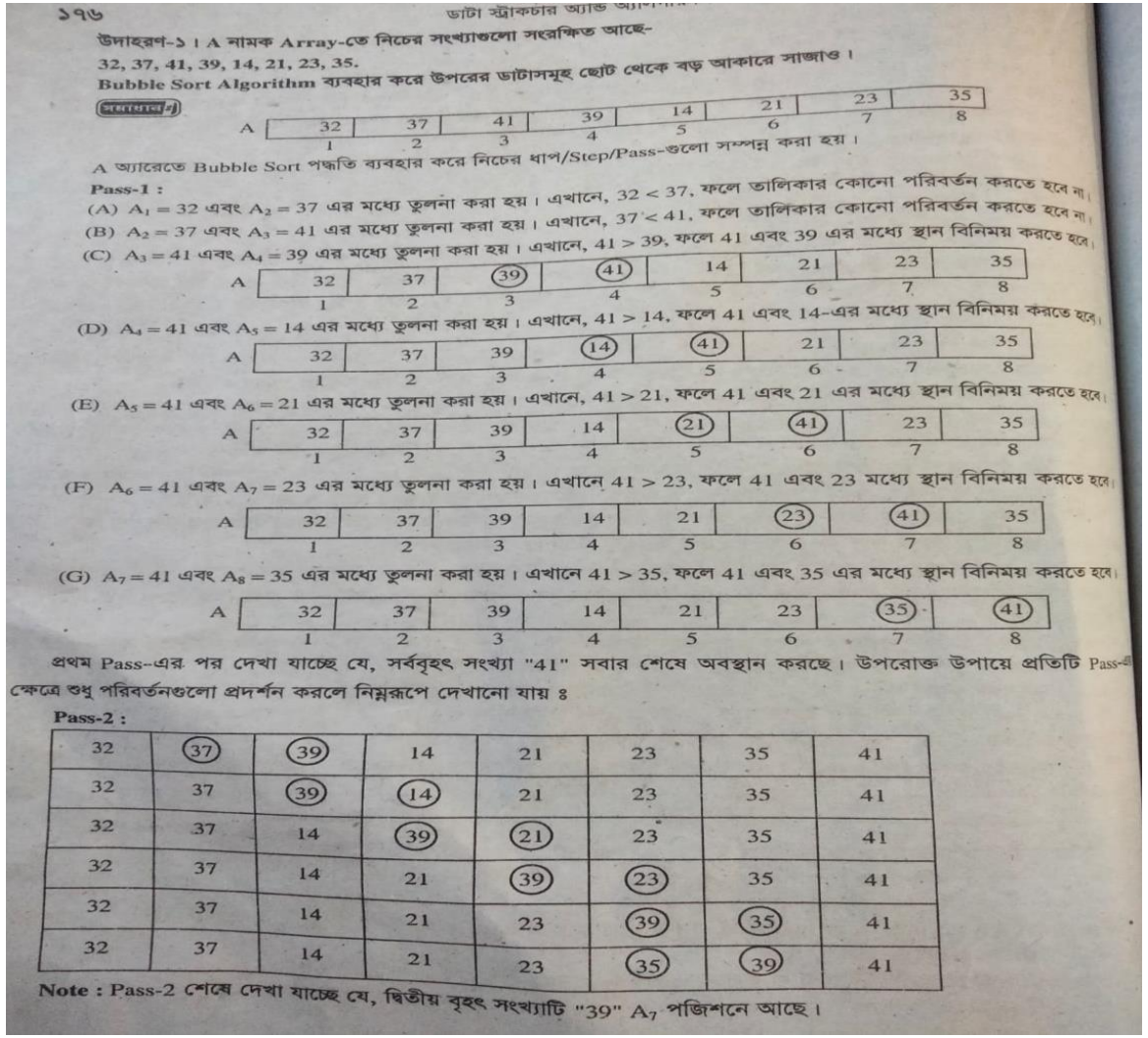
অধ্যায়-৯ ঃ সর্টিং অপারেশন

৯.২ বাবল সর্ট, কুইক সর্ট, এবং মার্জ সর্টের কৌশল ব্যাখ্যা (Describe the technique of bubble sonquick sort, and merge sort) :বাবল সর্টের ব্যাখ্যা (An explanation of bubble sort) :

Bubble বলতে বুদ্ বুঝায়। বুদবুদ যেমন কোন তরলে উ হয়ে সবার উপরে ওঠে অথবা সবার নিচে নেমে আসে, সেরূপ bubble সর্টেও পাশাপাশি দুটি উপাদানকে তুলনা করে প্রয়োজনবো দুটি উপাদানের স্থান পরিবর্তন করে সাজানো হয় এবং এভাবে পর্যায়ক্রমে সকল উপাদান পরস্পরের সাথে তুলনা করে সবচেয়ে কাঅথবা সবচেয়ে ছোট উপাদানকে সবার উপরে স্থান দেয়া হয়। একটি উপাদান সবার উপরে আসার পর অন্যান্য উপাদানসমূহরে একই পদ্ধতিতে সাজানো হয়ে থাকে। এভাবে পর্যায়ক্রমে সকল উপাদানকে সাজিয়ে কোন একটি তালিকার উপাদানসমূহকে ছোঁ থেকে বড় অথবা বড় থেকে ছোট আকারে সাজানোর পদ্ধতিকে Bubble sort বলে।

অধ্যায়-৯ ঃ সর্টিং অপারেশন

উদাহরণ-১।



১৭৬ ডাটা স্ট্রাকচার অ্যান্ড অ্যালগরিদম

উদাহরণ-১ । A নামক Array-তে নিচের সংখ্যাগুলো সংরক্ষিত আছে-
32, 37, 41, 39, 14, 21, 23, 35.
Bubble Sort Algorithm ব্যবহার করে উপরের ডাটাসমূহ ছোট থেকে বড় আকারে সাজাও ।

সমাধান:

	1	2	3	4	5	6	7	8
A	32	37	41	39	14	21	23	35

A অ্যারেতে Bubble Sort পদ্ধতি ব্যবহার করে নিচের ধাপ/Step/Pass-গুলো সম্পন্ন করা হয় ।

Pass-1 :

(A) $A_1 = 32$ এবং $A_2 = 37$ এর মধ্যে তুলনা করা হয় । এখানে, $32 < 37$, ফলে তালিকার কোনো পরিবর্তন করতে হবে না ।

(B) $A_2 = 37$ এবং $A_3 = 41$ এর মধ্যে তুলনা করা হয় । এখানে, $37 < 41$, ফলে তালিকার কোনো পরিবর্তন করতে হবে না ।

(C) $A_3 = 41$ এবং $A_4 = 39$ এর মধ্যে তুলনা করা হয় । এখানে, $41 > 39$, ফলে 41 এবং 39 এর মধ্যে স্থান বিনিময় করতে হবে ।

	1	2	3	4	5	6	7	8
A	32	37	(39)	(41)	14	21	23	35

(D) $A_4 = 41$ এবং $A_5 = 14$ এর মধ্যে তুলনা করা হয় । এখানে, $41 > 14$, ফলে 41 এবং 14-এর মধ্যে স্থান বিনিময় করতে হবে ।

	1	2	3	4	5	6	7	8
A	32	37	39	(14)	(41)	21	23	35

(E) $A_5 = 41$ এবং $A_6 = 21$ এর মধ্যে তুলনা করা হয় । এখানে, $41 > 21$, ফলে 41 এবং 21 এর মধ্যে স্থান বিনিময় করতে হবে ।

	1	2	3	4	5	6	7	8
A	32	37	39	14	(21)	(41)	23	35

(F) $A_6 = 41$ এবং $A_7 = 23$ এর মধ্যে তুলনা করা হয় । এখানে $41 > 23$, ফলে 41 এবং 23 মধ্যে স্থান বিনিময় করতে হবে ।

	1	2	3	4	5	6	7	8
A	32	37	39	14	21	(23)	(41)	35

(G) $A_7 = 41$ এবং $A_8 = 35$ এর মধ্যে তুলনা করা হয় । এখানে $41 > 35$, ফলে 41 এবং 35 এর মধ্যে স্থান বিনিময় করতে হবে ।

	1	2	3	4	5	6	7	8
A	32	37	39	14	21	23	(35)	(41)

প্রথম Pass-এর পর দেখা যাচ্ছে যে, সর্ববৃহৎ সংখ্যা "41" সবার শেষে অবস্থান করছে । উপরোক্ত উপায়ে প্রতিটি Pass-এ ক্ষেত্রে শুধু পরিবর্তনগুলো প্রদর্শন করলে নিম্নরূপে দেখানো যায় ঃ

Pass-2 :

32	(37)	(39)	14	21	23	35	41
32	37	(39)	(14)	21	23	35	41
32	37	14	(39)	(21)	23	35	41
32	37	14	21	(39)	(23)	35	41
32	37	14	21	23	(39)	(35)	41
32	37	14	21	23	(35)	(39)	41

Note : Pass-2 শেষে দেখা যাচ্ছে যে, দ্বিতীয় বৃহৎ সংখ্যাটি "39" A_7 পজিশনে আছে ।

অধ্যায়-৯ ঃ সর্টিং অপারেশন

Pass-3 :

32	(37)	(14)	21	23	35	39	41
32	14	(37)	(21)	23	35	39	41
32	14	21	(37)	(23)	35	39	41
32	14	21	23	(37)	(35)	39	41
32	14	21	23	35	(37)	(39)	41

Pass-4 :

(32)	(14)	21	23	35	37	39	41
14	(32)	(21)	23	35	37	39	41
14	21	(32)	(23)	35	37	39	41
14	21	23	(32)	(35)	37	39	41

Pass-5 :

A	14	21	23	32	35	37	39	41
	1	2	3	4	5	6	7	8

সবশেষে A[1] এবং A[2] এর মধ্যে তুলনা করা হয়। এখানে 14 < 21, ফলে কোনো পরিবর্তন প্রয়োজন নেই। এর মানে ডাটাগুলো Ascending order-এ মেমরিতে সংরক্ষিত আছে।

অধ্যায়-৯ ঃ সর্টিং অপারেশন

মানের নিম্নক্রমানুসারে সাজানোর জন্য বাবল সর্ট অ্যালগরিদম (Algorithm of bubble sort in descending order) Algorithm: (Bubble sort) BUBBLE (DATA, N)

এখানে DATA হচ্ছে N উপাদানবিশিষ্ট অ্যারে। এই অ্যালগরিদম DATA-এর উপাদানসমূহকে মানের নিম্নক্রমানুসারে সজ্জিত করে।

1. K = 1 to N-1 পর্যন্ত 2 এবং 3 স্টেপের লুপের পুনরাবৃত্তি হবে।
2. PIR = 1 নির্ধারণ করা হবে। [পাস পয়েন্টার PTR এর প্রাথমিককরণ]
3. PIR SNK পর্যন্ত (While লুপ পুনরাবৃত্তি হবে। [পাস এর নির্বাহ.
যদি DATA [PTR] < DATA [PTR + 1] হয়, তাহলে DATA [PTR] এবং DATA [PTR + 1] এর পরস্পর বিনিময় হবে।
[If structure সমাপ্ত হবে।]
b. PIR = PTR+ নির্ধারণ করা হবে।
[Inner Loop সমাপ্ত হবে।
[Step 1 আউটার লুপ সমাপ্ত হবে।
4. অ্যালগরিদম সমাপ্ত হবে

অধ্যায়-৯ ঃ সর্টিং অপারেশন

৯.৩ বাবল সর্ট, কুইক সর্ট, এবং মার্জ সর্ট এর অ্যালগরিদম লেখ (Write the Algorithms for Bubble, Quick, Heap, Insertion, Selection and Merge Sort)

বাবল সর্টের অ্যালগরিদম (Algorithm of bubble sort) Algorithm: (Bubble sort) BUBBLE (DATA, N)

এখানে DATA হচ্ছে N উপাদানবিশিষ্ট অ্যারে। এই অ্যালগরিদম DATA-এর উপাদানসমূহকে সজ্জিত করে।

1. K = 110 N - 1 পর্যন্ত 2 এবং 3 স্টেপের গ্রুপের পুনরাবৃত্তি হবে।
2. PIR = 1 নির্ধারণ করা হবে। [পাস পয়েন্টার PTR এর প্রাথমিককরণ]
3. PIR SNK পর্যন্ত (While লুপ পুনরাবৃত্তি হবে। (পাস এর নির্বাহ)

a. যদি DATA [PTR] < DATA [PTR + 1] হয়, তাহলে DATA [PTR] এবং DATA (PTR 1 এর পরস্পর স্থান বিনিময় হবে।

[If স্ট্রাকচার সমাপ্ত হবে।}

b. FIR = PTR + নির্ধারণ করা হবে।

[Inner Loop সমাপ্ত হবে।

[Step | আউটার লুপ সমাপ্ত হবে।

4. অ্যালগরিদম সমাপ্ত হবে।

অধ্যায়-৯ ঃ সর্টিং অপারেশন

৯.৪ বিভিন্ন Sorting Algorithm বাবল সর্ট, কুইক সর্ট এবং মার্জ সর্টএর Complexity তুলন (Compare the complex ty of dubile sort, quick sort & merge sort algorithm) :

বাবল সর্ট অ্যালগরিদমের 'টিলতা (The complexities of Bubble sort algorithm) : কোন Algorithm এর Complexity বলতে ঐ Program রান করত যে সময় প্রয়োজন হয়, তাকে বুঝায়। সর্ট-এর সময় যতগুলো তুলনার প্রয়োজন হয় তা র Complexity পরিমাপ করা হয়। Bubble sort এর Complexity বা তুলনার সংখ্যা f(n) নিম্ন উপায়ে সহজে নির্ণয় করা যায়।

কুইক সর্ট অ্যালগরিদমের জটিলতা (The complexities of Quick sort algorithm)।
n সংখ্যক উপাদানের জন্য sorting time f(n) নির্ণয়ের জন্য দুটি উপায় বর্ণনা করা হলো-
(i) অতি মন্দ ক্ষেত্র (Worst Case)
(ii) গড় ক্ষেত্র (Average Case).

অধ্যায়-৯ ঃ সর্টিং অপারেশন

মার্জ সর্টের অ্যালগরিদমের জটিলতা (The complexities of Merge sort algorithm) :

N উপাদানবিশিষ্ট A অ্যারেকেsorting করার জন্য Merge Sort Algorithm এর সর্বাধিক login পাসের প্রয়োজন হয়। উপরন্তু প্রতিটি pass-এর মোট n সংখ্যক উপাদানকে Merge করতে n সংখ্যক তুলনার প্রয়োজন হয়। সে হিসেবে Worst Case এবং Average Case উভয়ের জন্য অ্যালগরিদমের Complexity -f (n) n log

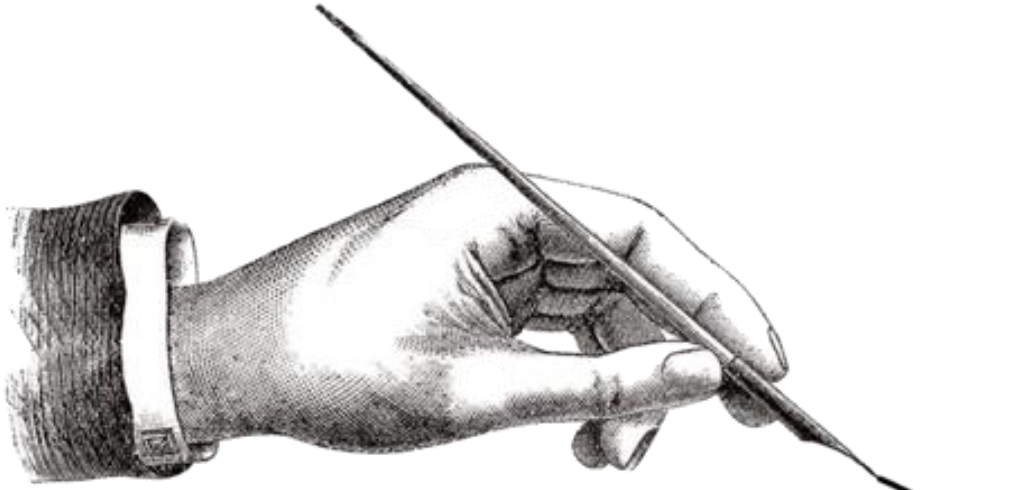
অতএব, Merge Sort এর Complexity হীপ Sort এর Worst এবং Quick Sort এর Average Complexity-এর সমান।

অধ্যায়-৯ ঃ সর্টিং অপারেশন

অ্যালগরিদমের Complexity - বিভিন্ন Sorting অ্যালগরিদমের Complexity-এর পারস্পরিক তুলনা (Comparisons Among Complexities ofDifferent Sorting Algorithm) :

নিচে ছকের মাধ্যমে Sorting Algorithm এর Complexity এর পারস্পরিক তুলনা দেয়া হলো–

SL	Algorithm	Worst Case	Average Case
(i)	Bubble Sort	$\frac{n(n-1)}{2} = O(n^2)$	$\frac{n(n-1)}{2} = O(n^2)$
(ii)	Quick Sort	$\frac{n(n+3)}{2} = O(n^2)$	$1.4n \log_n = O(n\log_2 n)$
(iii)	Heap Sort	$3n \log_2 n = O(n \log_2 n)$	$3n \log_2 n = O(n \log_2 n)$
(iv)	Insertion Sort	$\frac{n(n-1)}{2} = O(n^2)$	$\frac{n(n-1)}{4} = O(n^2)$
(v)	Selection Sort	$\frac{n(n-1)}{2} = O(n^2)$	$\frac{n(n-1)}{2} = O(n^2)$
(vi)	Merge Sort	$n \log_2 n = O(n \log_2 n)$	$n \log_n = O(n \log_2 n)$



Thank You

Subject Name: Data Structure & Algorithm