DGITechChronicle

DGI TECH CHRONICLE

IT EDITION

Vol I Issue I (Jul-Dec 2020)

EDITORIAL BOARD

Dr. Aadarsh Malviya
(Assistant Professor)

Editor in Chief

In this issue, we delve into a captivating array of topics and developments, all tailored to the inquisitive minds of the future engineers. As an engineering college community, we stand at the forefront of technological breakthroughs, and it is our mission to empower you with the knowledge and insights to not only keep pace but to lead in this ever-accelerating race of innovation.

Paras Soni
12607 (CSIT)

Co- Editor

Tushar Tayal
12626 (CSIT)

Editor- Design

Shikha Kumari
12617(CSIT)

Editor- Text

DGITechChronicle

CSIT EDITION

VISION

Promoting technologists by imparting profound knowledge in information technology, all while instilling ethics through specialized technical education.

MISSION

Delivering comprehensive knowledge in information technology, preparing technologists to excel in a rapidly evolving digital landscape.

Building a culture of honesty and responsibility in tech, promoting smart and ethical leadership.

Empowering individuals with specialized technical skills and ethical values to drive positive change and innovation in the tech industry.

Program Educational Objectives (PEO)

To enable graduates to think logically, pursue lifelong learning and will have the capacity to understand technical issues related to computing systems and to design optimal solutions.

To enable graduates to develop hardware and software systems by understanding the importance of social, business and environmental needs in the human context.

To enable graduates to gain employment in organizations and establish themselves as professionals by applying their technical skills to solve real world problems and meet the diversified needs of industry, academia and research.

Program Specific Outcome (PSO)

To adapt to emerging technologies and develop innovative solutions for existing and newer problems.

To create and apply appropriate techniques IT tools to complex engineering activities with an understanding of the limitations.

To create and apply appropriate techniques IT tools to complex engineering activities with an understanding of the limitations.

Program Outcome (PO)

Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

Modern tool usage: Create, select, and apply appropriate techniques, resources,& modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one’s own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Hamming Cut Matching Algorithm

The Hamming Cut Matching Algorithm, a gem in the realm of computer science, stands out for its efficiency and versatility in solving a variety of matching problems. Named after Richard Hamming, this algorithm has found applications in diverse fields, from error detection in data transmission to DNA sequence analysis.

At its core, the Hamming Cut Matching Algorithm leverages the Hamming distance, a metric that measures the dissimilarity between two strings of equal length. In the context of the algorithm, these strings often represent binary codes or sequences. The algorithm's prowess lies in its ability to identify the minimum cut that separates unmatched pairs, optimizing the matching process.

One of the algorithm's notable applications is in error detection and correction, where it excels at identifying and rectifying discrepancies in transmitted data. By analyzing the Hamming distance between the received and expected data, the algorithm efficiently pinpoints errors, ensuring the integrity of the information.

In bioinformatics, the Hamming Cut Matching Algorithm plays a crucial role in DNA sequence analysis. It aids in identifying similarities and differences between genetic codes, providing valuable insights into evolutionary relationships and genetic mutations.

The algorithm's elegance lies in its simplicity and speed, making it a go-to solution for scenarios where rapid and accurate matching is essential. As technology continues to evolve, the Hamming Cut Matching Algorithm remains a cornerstone in the toolkit of computer scientists, contributing to advancements in communication, genetics, and bevond.

Tushar Tayal
12626 (CSIT)

Fog Computing

Fog Computing, a revolutionary paradigm in the field of distributed computing, is reshaping the way we process and analyze data in the digital age. Unlike traditional cloud computing that centralizes data processing in remote data centers, Fog Computing brings computational resources closer to the edge of the network, enabling faster and more efficient data processing.

At its core, Fog Computing extends the capabilities of cloud computing by decentralizing computation and storage resources to the edge of the network, near the data source. This proximity reduces latency, enhances real-time processing, and minimizes the strain on network bandwidth. Devices and sensors at the network's edge, such as IoT devices and autonomous vehicles, benefit from Fog Computing's ability to process data locally, leading to quicker decision-making and improved overall system performance.

One of Fog Computing's key advantages is its ability to handle the massive influx of data generated by the Internet of Things (IoT) devices. By processing data closer to the source, Fog Computing mitigates the challenges posed by transmitting large volumes of data to distant cloud servers, promoting efficiency and reducing response times.

In addition to its speed and efficiency, Fog Computing enhances data security and privacy. By keeping sensitive data closer to its origin, potential security risks associated with transmitting data over long distances are minimized. This makes Fog Computing an attractive solution for industries where data security is paramount, such as healthcare and finance.

As we navigate an increasingly interconnected world, Fog Computing stands as a transformative force, unlocking new possibilities for real-time data processing, low-latency applications, and secure, decentralized computing at the edge of the network. The evolution of Fog Computing continues to shape the digital landscape, promising a future where data is processed seamlessly and efficiently, enhancing our connected experiences.

Paras Soni
12607 (CSIT)

Dynamic Memory Allocation

Dynamic memory allocation refers to the process of allocating memory for variables at runtime, as opposed to the static memory allocation that occurs at compile-time. In many programming languages, dynamic memory allocation is a crucial feature that allows programs to manage memory efficiently and handle data structures of varying sizes.

Commonly used programming languages like C, C++, and Python provide mechanisms for dynamic memory allocation through functions or keywords like `malloc`, `free`, `new`, `delete`, `malloc`, `calloc`, `realloc`, etc.

Here's a brief overview of dynamic memory allocation in C and C++:

C Language:

1. **malloc:**
   - **Prototype:** `void* malloc(size_t size);`
   - Allocates a specified number of bytes of memory.
   - Returns a pointer to the beginning of the allocated block or `NULL` if the allocation fails.

c Copy code

```c
int* arr = (int*)malloc(5 * sizeof(int));
```

1. **calloc:**
   - **Prototype:** `void* calloc(size_t num_elements, size_t element_size);`
   - Allocates memory for an array of elements, each of a specified size, and initializes them to zero.
   - Returns a pointer to the beginning of the allocated block or `NULL` if the allocation fails.

c Copy code

```c
int* arr = (int*)calloc(5, sizeof(int));
```

Dynamic Memory Allocation

1. **realloc:**
   - **Prototype:** `void* realloc(void* ptr, size_t new_size);`
   - Changes the size of the memory block pointed to by `ptr` to the specified `new_size`.
   - Returns a pointer to the beginning of the new block or `NULL` if the allocation fails.

c Copy code

2. **new and delete (for single variable):**
   - **Dynamic Allocation:** `int* num = new int;`
   - **Dynamic Deallocation:** `delete num;`
   - `new` and `delete` can also be used for a single variable.

Dynamic memory allocation provides flexibility, but it also requires careful management to avoid memory leaks (failure to deallocate memory) and other memory-related issues. In languages with automatic memory management (e.g., Java, Python), these low-level operations are often abstracted away, reducing the risk of memory-related bugs.

C++ Language:

1. **new and delete:**
   - **Dynamic Allocation:** `int* arr = new int[5];`
   - **Dynamic Deallocation:** `delete[] arr;`
   - `new` allocates memory for a single variable or an array of variables.
   - `delete` deallocates memory allocated by `new`.

2. **new and delete (for single variable):**
   - **Dynamic Allocation:** `int* num = new int;`
   - **Dynamic Deallocation:** `delete num;`
   - `new` and `delete` can also be used for a single variable.