**CU / AIT / COMPUTER SCIENCE AND ENGINEERING / 3 / 20CSP382**

****

**APEX INSTITUTE OF TECHNOLOGY**

**(AIT)**

Department of Computer Science and Engineering

B.E. – Computer Science and Engineering

SEMESTER : Sixth (6th)

SUBJECT NAME : Technical Training

SUBJECT CODE : 20CSP382

FACULTY : Dr. Prasenjit Das

**Contents of Course File**

|  |  |  |
| --- | --- | --- |
| **Sr. No.** | **Particular** | **Page No.** |
| 1 | University-Vision and Mission | 3 |
| 2 | Department-Vision and Mission | 4 |
| 3 | PEO | 5 |
| 4 | PO | 6 |
| 5 | PSO | 7 |
| 6 | Course Objectives | 7 |
| 7 | Course Outcomes | 7 |
| 8 | Mapping of COs/POs/PSOs | 7 -8 |
| 10 | Experiment 1…10  Aim  Objective  Input/Apparatus Used  Procedure/Algorithm/Code  Observations/Outcome  Question: Viva Voce | 11-79 |
| 11 | LAB MST  Question Paper (Mapped with COs)  Scheme of Evaluation  Sample Answer Sheets  CO-Wise Mark Sheets  List of Slow Learners and Advance Learners  Action Taken | 80-82 |
| 12 | Best Practices Adopted | 82 |
| 13 | Final Award Sheet | 82 |
| 14 | List of Slow Learners and Advance Learners | 82 |
| 15 | Overall Observation and Action Taken (Recommendation to BOS) | 83 |

Vision of Chandigarh University

To be globally recognized as a Centre of Excellence for Research, Innovation, Entrepreneurship and disseminating knowledge by providing inspirational learning to produce professional leaders for serving the society.

Mission of Chandigarh University

Providing world class infrastructure, renowned academicians and ideal environment for Research, Innovation, Consultancy and Entrepreneurship relevant to the society.

Offering programs & courses in consonance with National policies for nation building and meeting global challenges.

Designing Curriculum to match international standards, needs of Industry, civil society and for inculcation of traits of Creative Thinking and Critical Analysis as well as Human and Ethical values.

Ensuring students delight by meeting their aspirations through blended learning, corporate mentoring, professional grooming, flexible curriculum and healthy atmosphere based on co-curricular and extra-curricular activities.

Creating a scientific, transparent and objective examination/evaluation system to ensure an ideal certification.

Establishing strategic relationships with leading National and International corporates and universities for academic as well as research collaborations.

Contributing for creation of healthy, vibrant and sustainable society by involving in Institutional Social Responsibility (ISR) activities like rural development, welfare of senior citizens, women empowerment, community service, health and hygiene awareness and environmental protection

Vision of the Department

To be recognized as a centre of excellence for Computer Science & Engineering education and research, through effective teaching practices, hands-on training on cutting edge computing technologies and excellence in innovation, for creating globally aware competent professionals with strong work ethics whom would be proficient in implementing modern technology solutions and shall have entrepreneurial zeal to solve problems of organizations and society at large.

Mission of the Department

M1: To provide relevant, rigorous and contemporary curriculum and aligned assessment system to ensure effective learning outcomes for engineering technologies.

M2: To provide platform for industry engagement aimed at providing hands-on training on advanced technological and business skills to our students.

M3: To provide opportunities for collaborative, interdisciplinary and cutting-edge research aimed at developing solutions to real life problems

M4: To imbibe quest for innovation, continuous learning and zeal to pursue excellence through hard work and problem-solving approach

M5: To foster skills of leadership, management, communication, team spirit and strong professional ethics in all academic and societal endeavours of our students

Program Education Objectives

**PE01** To be able to explore areas of research, technology application & innovation and make a positive impact in different types of institutional settings such as corporate entities, government bodies, NGOs, inter-government organizations, & start-ups.

**PEO2** To be able to design, and implement technology and computing solutions to the organizational problems, effectively deploy knowledge of engineering principles, demonstrate critical thinking skills & make the intellectual connections between quantitative and qualitative tools, theories and context to solve the organizational problems

**PEO3** To be able to work with, lead & engage big and small teams comprising diverse people in terms of gender, nationality, region, language, culture & beliefs. To understand stated and unstated differences of views, beliefs & customs in diverse & inter disciplinary team settings

**PEO4** To be able to continuously learn and update one’s knowledge, engage in lifelong learning habits and acquire latest knowledge to perform in current work settings

**PEO5** To continuously strive for justice, ethics, equality, honesty, and integrity both in personal and professional pursuits. Able to understand and conduct in a way that is responsible and respectful.

Program Outcomes

**PO1 Engineering Knowledge**: Apply knowledge of mathematics, science, engineering fundamentals and an engineering specialization to the solution of complex engineering problems.

**PO2 Problem Analysis**: Identify, formulate, research literature and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences and engineering sciences.

**PO3 Design/ Development of Solutions**: Design solutions for complex engineering problems and design system components or processes that meet specified needs with appropriate consideration for public health and safety, cultural, societal and environmental considerations.

**PO4 Conduct investigations of complex problems** using research-based knowledge and research methods including design of experiments, analysis and interpretation of data and synthesis of information to provide valid conclusions.

**PO5 Modern Tool Usage**: Create, select and apply appropriate techniques, resources and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

**PO6 The Engineer and Society**: Apply reasoning informed by contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to professional engineering practice.

**PO7 Environment and Sustainability**: Understand the impact of professional engineering solutions in societal and environmental contexts and demonstrate knowledge of and need for sustainable development.

**PO8 Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of engineering practice.

**PO9 Individual and Team Work**: Function effectively as an individual, and as a member or leader in diverse teams and in multi-disciplinary settings.

**PO10 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.

**PO11 Project Management and Finance**: Demonstrate knowledge and understanding of 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

**PO12 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.

COURSE OBJECTIVES

# The Course aims to:

1. To provide hands-on for programming languages C and C++.
2. To provide in-depth knowledge of the various techniques of problem solving
3. To make the students industry ready and take up challenges of IT Industry

COURSE OUTCOMES

On completion of this course, the students shall be able to

1. Compare and contrast programming skills in languages such as C and C++.
2. Interpret the concepts of various data structures for real-time problems.
3. Sketch an efficient algorithm to solve the real-time problem.
4. Design a solution for existing coding challenges in the field of IT and improve employability.
5. Correlate and develop a solution to improve the complexity of the real-time problem.

Mapping of COs/POs/PSOs

|  |  |  |
| --- | --- | --- |
| **Mapping Between COs and Pos** | | |
| **S.No.** | **Course Outcome (CO)** | **Mapped Programme Outcome (PO)** |
| 1 | Compare and contrast programming skills in languages such as C and C++. | 1. to 5 |
| 2 | Interpret the concepts of various data structures for real-time problems. | 1. to 5 |
| 3 | Sketch an efficient algorithm to solve the real-time problem | 1 to 5 |
| 4. | Design a solution for existing coding challenges in the field of IT and improve employability | 1 to 5 |
| 5. | Correlate and develop a solution to improve the complexity of the real-time problem | 1to 5 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | Engineering Knowledge | Problem analysis | Design/development of solutions | Conduct investigations of complex problems | Modern tool usage | The engineer and society | Environment and sustainability | Ethics | Individual or team work | Communication | Project management and finance | Life-long Learning |
|  |  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| CSP-382 | CO1 | 2 | 2 | 3 | 3 | 2 | - | - | - | - | 1 | - | 2 |
| Technical Training | CO2 | 2 | 2 | 3 | - | 1 | - | - | - | 2 | 3 | - | - |
| CO3 | 3 | 2 | 3 | 3 | 3 | 2 | 1 | - | 3 | - | 2 | 3 |
| CO4 | 2 | 2 | 2 | 2 | 2 | 2 | - | - | - | 3 | - | - |
| CO5 | 2 | 3 | 2 | 2 | 3 | 2 | 2 | 3 | - | - | 1 | 2 |

1=addressed to small extent

2= addressed significantly

3=major part of course

Syllabus

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **20-CSP-382** | **Technical Training Lab** | **L** | **T** | **P** | **S** | **C** | **CH** |
| Version 1.00 | | 0 | 0 | 2 | 0 | 1 | 2 |
| Pre-requisites/ Exposure | 1. Knowledge of Basics of Programming. 2. Knowledge of basic data structure. | | | | | | |

**COURSE DESCRIPTION**

The course provides hands on for technical coding. Experiments/programs are aimed to enhance the skills of programming in the languages of C and C++. Students who have already done basic coding will be exposed to problems generally asked in various placement activities. The idea is to make the students industry ready.

**TEXT BOOKS**

**T1** Reema Thareja, "Data Structures Using C", Oxford Publications, 2016.

**T2** Sartaj Sahni, "Fundamentals of Algorithms",University Press, 2017.

**REFERENCE BOOKS**

**R1** E Balaguruswamy , “Object Oriented Programming”, McGraw Hill 2020

**R2** Reema Thareja, "programming in C", Oxford Publications, 2016.

**COURSE CONTENT**

# Unit I 8 Contact Hours

1. A data structure needs to be implemented in such a way that we have the references i.e. the addresses of the values. None of the addresses are in continuous memory block. Each time a new value needs to be stored, we need to allocate memory. Use a particular Data Structure to implement other data structures.
2. Two lists are to be maintained containing integer values. The values are not stored in continuous memory locations. Let one list be 7->1->5 and another list be 5->9->2.Create a third list such that the third list contains the sum of the first two list i.e 517+295 =812 and save it as 2->1>8
3. Names of persons are saved in a list. Create function to check if the names are palindrome or not.
4. You are in the process of creating a text editor (like notepad /Ms Word). The text editor saves particular word in form a string. We need to provide the feature of Find and Replace a particular substring in the given text editor.

# Unit II 8 Contact Hours

1. Characters are sorted in a form of a list (non continuous). We have to check if the list of characters has a given character in it or not.
2. Three list are maintained with the integer roll nos of the students. One list contains the roll nos. and name of the students, second list contains the roll nos and the course name (BE, BCA etc) for the student and the third list contains the roll no. and CGPA of the student. Find a way to check the common roll nos. in the 3 lists. Note: the 3 lists are sorted by the roll nos.
3. A set of strings are saved in a data structure. Represent the strings in form a non linear data structure in such a way that the searching takes the minimal time.
4. Without comparing the integer values in an array, sort the array of integers in ascending order.

# Unit III 8 Contact Hours

9. List of Marks obtained by students of your class is maintained in form a linked List. Apply a sorting algorithm with complexity O (n^2) to sort the list.

10. a) Two arrays are maintained such that the marks are in descending order. We need to reverse the two arrays using another data structure so that the arrays are in ascending order. Hint: Use LIFO

10. b) A thief enters a house. He is carrying a sack of some capacity say ‘m’. He can fill the sack with objects in the house and can carry a fraction of the object. The weights of each object are given by wi. For carrying each of the object in the sack, he earns a profit of pi. Write a code to maximize his profit.

10. c) A thief enters a house. He is carrying a sack of some capacity say ‘m’. He can fill the sack with objects in the house and can carry a fraction of the object. The weights of each object are given by wi. For carrying each of the object in the sack, he earns a profit of pi. Write a code to maximize his profit.

MODE OF EVALUATION: The performance of students is evaluated as follows:

|  |  |  |
| --- | --- | --- |
|  | **Practical** | |
| **Components** | **Continuous Internal Assessment (CAE)** | **Semester End Examination (SEE)** |
| **Marks** | 60 | 40 |
| **Total Marks** | 100 | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **CO vs PO/PSO** | **PO1** | **PO2** | **PO3** | **PO4** | **PO5** | **PO6** | **PO7** | **PO8** | **PO9** | **PO10** | **PO11** | **PO12** | **PSO1** | **PSO2** | **PSO3** | **PSO4** |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **CO1** |  |  |  | 3 | 2 | - |  |  |  |  |  |  |  | - | -- |  |
|  | 2 | 2 | 3 |  |  |  | -- | - | - | 1 | - | 2 | 2 |  |  | 3 |
| **CO2** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | 2 | 2 | 3 | - | 1 | - | - | - | 2 | 3 | - | - | 3 | 2 | 1 | -- |
| **CO3** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | 3 | 2 | 3 | 3 | 3 | 2 | 1 | - | 3 | - | 2 | 3 | 3 | 3 | - | - |
| **CO4** | 2 |  |  |  |  |  |  |  |  |  |  |  |  | 3 |  |  |
|  |  | 2 | 2 | 2 | 2 | 2 | - | - | - | 3 | - | - | 1 |  | - | 3 |
| **CO5** |  |  |  |  |  |  |  |  | - |  |  |  |  | - | 3 |  |
|  | 2 | 3 | 2 | 2 | 3 | 2 | 2 | 3 |  | - | 1 | 2 |  |  |  | 2 |
| 1=addressed to small extent  2= addressed significantly  3=major part of course | | | | | | | | | | | | | | | | |

Experiment 1

Experiment 1: A data structure needs to be implemented in such a way that we have the references i.e. the addresses of the values. None of the addresses are in continuous memory block. Each time a new value needs to be stored, we need to allocate memory. Write a program to implement the following:

* + - 1. Addition of a new value at a given position
      2. Print all the values in the list
      3. Delete a given value from a location

**CO Attained:** CO2 and CO3

**Objective:**

In this algorithm, we want to perform the basic operations on a linked list. .

**Apparatus Used:**C/C++, Dev C++

**Program Logic:**

For a given linked lists, students are required to complete the basic operations like insertion , deletion and traversing a linked list.

**Algorithm :**

* Traverse the linked list
* Reach the node after which the insertion is to be done
* Save the address in temp
* Create the node (memory allocation)
* Provide the address of the new node to temp

**Code:**

#include<bits/stdc++.h>

using namespace std;

class Node{

public:

int val;

Node \*next;

Node(){

this->val = INT\_MIN;

this->next = NULL;

}

Node(int x){

val = x;

this->next = NULL;

}

};

class LL{

Node \*head;

public:

LL(){

this->head = NULL;

}

void insertAtBegin(int x){

Node \*n1 = new Node(x);

n1->next = head;

head = n1;

}

void insertAtPos(int x, int pos){

int n = getSize();

if(pos>n || pos<0){return ;}

if(pos==0){insertAtBegin(x); return ;}

Node \*p1 = head;

while(--pos){

p1 = p1->next;

}

Node \*n1 = new Node(x);

n1->next = p1->next;

p1->next = n1;

}

void insertAtEnd(int x){

Node \*n1 = new Node(x);

if(head==NULL){

head=n1;

return ;

}

Node \*temp = head;

while(temp->next){

temp = temp->next;

}

temp->next = n1;

}

void deleteBegin(){

if(!head){

cout<<"Underflow: No data to delete.";

return ;

}

head = head->next;

}

void deletValue(int x){

if(!head){

cout<<"Underflow: No data to delete.";

return ;

}

Node \*prev=NULL , \*curr=head;

bool flag=true;

while(curr){

if(curr->val == x){

flag=false;

Node \*temp = curr;

prev->next = curr->next;

curr->next = prev;

// delete(curr);

// delete(temp);

}

prev = curr;

curr = curr->next;

}

if(flag){

cout<<"There exist no element which is asked to delete.\n";

}else{

cout<<"Deleted\n";

}

}

void deleteAtPos(int x, int pos){

int n = getSize();

if(pos>n || pos<=0){return ;}

if(pos==1){deleteBegin(); return ;}

Node \*p1 = head, \*prev=nullptr;

while(--pos){

prev = p1;

p1 = p1->next;

}

if(p1->val==x){

prev->next = p1->next;

free(p1);

}

}

void deleteEnd(){

if(!head){

cout<<"Underflow: No data to delete.";

return ;

}

Node \*temp = head;

while(temp->next->next){

temp = temp->next;

}

temp->next = NULL;

}

void sorting(){

Node \*temp1 = head, \*temp2 = head;

while(temp1->next){

temp2 = temp1->next;

while(temp2){

if(temp2->val < temp1->val){

int temp = temp2->val;

temp2->val = temp1->val;

temp1->val = temp;

}

temp2 = temp2->next;

}

temp1 = temp1->next;

}

}

void printList(){

if(!head){

cout<<"Underflow: No data to print.";

return ;

}

Node \*temp = head;

while(temp){

cout<<temp->val<<"->";

temp = temp->next;

}

}

Node\* getHead(){return this->head;}

int getSize(){

if(!head)return 0;

if(!head->next)return 1;

Node \*curr = head;

int ans=0;

while(curr){

curr = curr->next;

ans++;

}

return ans;

}

};

int main(){

LL l1;

l1.insertAtEnd(11); //

l1.insertAtEnd(13); //

l1.insertAtEnd(12); //

l1.insertAtEnd(19); //

l1.insertAtEnd(12); //

l1.insertAtEnd(13);

l1.insertAtPos(999,1);

l1.deleteAtPos(12,2);

l1.printList();

return 0;

}

Output:

999,13,12,19,12, 13,11

VIVA Questions:

Q1. Why do we need a linked list

Q2. What other data structures could have been used instead of Linked list

Q3. How to identify the last node of Linked list

Q4. How can we add the linked list in reverse order.

Q5. What is the time complexity of the algorithm?

Experiment 2: Two lists are to be maintained containing integer values. The values are not stored in continuous memory locations. Let one list be 7->1->5 and another list be 5->9->2. Create a third list such that the third list contains the sum of the first two list i.e 517+295 =812 and save it as 2->1>8.

**CO Attained**: CO1 and CO2

**Objective:**

Student will be able to traverse a linked list and will learn how to reverse the same

**Apparatus Used:**C/C++, Dev C++

**Program Logic:**

Traverse the linked list from head to last node. While traversing, another pointer will take care of the last node traversed. This way the traversal in reverse order will result in the formation of numbers to be added.

**Algorithm**

1. Read the linked list 1 and reverse the numbers

1. Read the Linked list 2 and reverse the numbers
2. Add the two numbers and save it in 3rd Linked list
3. Print the list in reverse order

Node\* addTwoNumbers(Node\* l1, Node\* l2) {

if(!l1)return NULL;

if(!l2)return NULL;

Node\* ptr = new Node(0);

Node\* prev = ptr;

int cary = 0;

while(l1 || l2){

int sum = (l1 != NULL ? l1->val : 0) + (l2 != NULL ? l2->val : 0) + cary;

cary = sum/10;

Node\* temp = new Node(sum%10);

prev->next = temp;

prev = prev->next;

if(l1)l1=l1->next;

if(l2)l2=l2->next;

}

if(cary){

Node\* temp = new Node(cary);

prev->next = temp;

}

return ptr->next;

}

Outcome:

L1:7->1->5

L2:5->9->2

Output: 2->1->8

VIVA Questions:

Q1. Difference between doing the experiment with arrays and Linked list

Q2. What is the time complexity of the experiment

Q3 What other DS could have been used

Q4 . What if there is no carry while calculating the sum? Will the logic work with negative numbers?

Q4. Can the time/space complexity be improved? If so How?

Experiment 3: Names of persons are saved in a list. Create function to check if the names are palindrome or not.

CO attained CO1, CO3, CO5

**Objective:**

Student will learn about characters saved in non continows memory locations i.e a single linked list.

**Apparatus Used:**C/C++, Dev C++

**Program Logic:**

Characters are generally stored in strings. Now students have already reversed a string in C/C++. But if the same are saved in non continuous memory locations, then reversing it will require the characters to be stored in separate Linked List. When the two linked lists (original & reversed) are compared character by character, the palindrome can be checked.

Algorithm:

1 Save the characters in a linked list

2 Read the list from head to last node

1. Save the characters in a string
2. Reverse the string and check for palindrome

#include <bits/stdc++.h>

using namespace std;

class Node {

public:

int data;

Node(int d) { data = d; }

Node\* ptr;

};

// Function to check if the linked list

// is palindrome or not

bool isPalin(Node\* head)

{

// Temp pointer

Node\* slow = head;

// Declare a stack

stack<int> s;

// Push all elements of the list

// to the stack

while (slow != NULL) {

s.push(slow->data);

// Move ahead

slow = slow->ptr;

}

// Iterate in the list again and

// check by popping from the stack

while (head != NULL) {

// Get the top most element

int i = s.top();

// Pop the element

s.pop();

// Check if data is not

// same as popped element

if (head->data != i) {

return false;

}

// Move ahead

head = head->ptr;

}

return true;

}

// Driver Code

int main()

{

// Addition of linked list

Node one = Node(1);

Node two = Node(2);

Node three = Node(3);

Node four = Node(2);

Node five = Node(1);

// Initialize the next pointer

// of every current pointer

five.ptr = NULL;

one.ptr = &two;

two.ptr = &three;

three.ptr = &four;

four.ptr = &five;

Node\* temp = &one;

// Call function to check palindrome or not

int result = isPalin(&one);

if (result == 1)

cout << "isPalindrome is true\n";

else

cout << "isPalindrome is false\n";

return 0;

}

Another Solution

#include<bits/stdc++.h>

using namespace std;

struct Node

{

char data;

Node \*next;

};

class List

{

public:

Node \*head;

List()

{

head=NULL;

}

void insert(int x)

{

Node \*newn = new Node;

newn->data = x;

newn->next=NULL;

if(head == NULL)

head = newn;

else

{

Node \*trav=head;

while(trav->next!=NULL)

trav=trav->next;

trav->next=newn;

}

}

void traverse()

{

if(head == NULL)

cout<<"Empty List\n\n";

else

{

Node \*trav = head;

while(trav!=NULL)

{

cout<<trav->data;

trav=trav->next;

}

}

cout<<"\n";

}

List reverse()

{

List L;

if(head == NULL)

L.head = NULL;

else

{

Node \*trav = head;

while(trav!=NULL)

{

L.insert(trav->data);

trav=trav->next;

}

}

L.head= recrev(L.head);

return L;

}

Node \*recrev(Node \*node)

{

if(node==NULL || node->next==NULL)

return node;

Node \*ahead = recrev(node->next);

node->next->next=node;

node->next=NULL;

return ahead;

}

bool compare(List L)

{

bool flag = true;

Node \*trav1 = this->head;

Node \*trav2 = L.head;

while(trav1!=NULL || trav2!=NULL)

{

if(trav1->data != trav2->data)

flag=0;

trav1=trav1->next;

trav2=trav2->next;

}

return flag;

}

};

main()

{

cout<<"\n\nPalindrome Check...Enter String: \n";

string str;

cin>>str;

List L1,L2;

for(int i=0; i<str.length(); i++)

{

L1.insert(str[i]);

}

L2 = L1.reverse();

if(L1.compare(L2))

cout<<"Palindrome String !!";

else

cout<<"Not Palindrome !!";

cout<<"\n\n\n";

}

Input : radar

Output: Palindrome String

Input : Bus

Output: Not Palindrome

VIVA Questions:

Q1. Difference between doing the experiment with arrays and Linked list

Q2. What is the time complexity of the experiment

Q3 What other DS could have been used

Q4. Can the time/space complexity be improved? If so How?

Q5. Where is the Null value of the string stored?

Experiment 4: You are in the process of creating a text editor (like notepad /Ms Word). The text editor saves particular word in form a string. We need to provide the feature of Find and Replace a particular substring in the given text editor.

**CO Attained:** CO2 and CO4

**Objective:**

Student will be able to learn how find /searching in a text editor works

**Apparatus Used:**C/C++, Dev C++

**Program Logic:**

Apply any pattern matching algorithm. The text and pattern needs to converted to its respective hash values. If the Hash values turn out to be same, the two strings match. (KMP Pattern matching Algorithm)

Algorithm

* + - 1. Take the string
      2. Calculate its hash value
      3. Calculate the hash value of the pattern to be matched
      4. If the hash values of step 2 and step 3 are equal , it’s a palindrome
      5. Else it is not a present in the string

Code:

#include <stdio.h>

#include <math.h>

#include <stdbool.h>

int calc\_hash(char \*arr, int a, int n){

int pm = 2;

int ans = 0;

int k = n;

for(int i=a ; i<n+a ; i++){

ans += (arr[i])\*(pow(pm,(k--)-1));

}

return ans;

}

int main()

{

char str[] = "CABDCAC";

char pat[] = "ABD";

int n = sizeof(str)/sizeof(str[0])-1;

int m = sizeof(pat)/sizeof(pat[0])-1;

int hash = calc\_hash(pat,0,m);

printf("Pattern Hash : %d\n",hash);

bool flag = false;

for(int i=0 ; i<=n-m ; i++){

int curr\_hash = calc\_hash(str,i,m);

if(hash == curr\_hash){

flag = true;

break;

}

}

if(flag)printf("Found");

else printf("Not Found");

return 0;

}

Outcome:

Input text : “hello World”

Pattern: ello

Output: Found

Input text : “hello World”

Pattern: abc

Output: Not found

VIVA Questions:

Q1. Difference between doing the experiment with arrays and Linked list

Q2. What is the time complexity of the experiment

Q3 What other DS could have been used

Q4. Can the time/space complexity be improved? If so How?

Q5. Can we solve the problem using Brute Force?

Experiment 5: Characters are sorted in a form of a list (non continuous). We have to check if the list of characters has a given character in it or not.

CO Attained: CO3, CO4 and CO5

* + - 1. Take the string
      2. Calculate its hash value
      3. Calculate the hash value of the pattern to be matched
      4. If the hash values of step 2 and step 3 are equal , it’s a palindrome
      5. Else it is not a present in the string

**Objective:**

Student will be able to traverse a linked list and will learn how to reverse the same

**Apparatus Used:**C/C++, Dev C++

**Program Logic:**

Traverse the linked list from head to last node. While traversing, another pointer will take care of the last node traversed. This way the traversal in reverse order will result in the formation of numbers to be added.

Algorithm

1

#include<bits/stdc++.h>

using namespace std;

class Node{

public:

int val;

Node \*next;

Node(){

this->val = INT\_MIN;

this->next = NULL;

}

Node(int x){

val = x;

this->next = NULL;

}

};

class LL{

Node \*head;

public:

LL(){

this->head = NULL;

}

void insertAtBegin(int x){

Node \*n1 = new Node(x);

n1->next = head;

head = n1;

}

void insertAtEnd(int x){

Node \*n1 = new Node(x);

if(head==NULL){

head=n1;

return ;

}

Node \*temp = head;

while(temp->next){

temp = temp->next;

}

temp->next = n1;

}

bool isFound(int x){

Node \*temp = head;

while(temp){

if(temp->val == x){

return true;

}

temp = temp->next;

}

return false;

}

void printList(){

if(!head){

cout<<"Underflow: No data to print.";

return ;

}x`

Node \*temp = head;

while(temp){

cout<<temp->val<<"->";

temp = temp->next;

}

}

};

int main(){

LL l1,l2,l3;

l1.insertAtEnd(1);

l1.insertAtEnd(5);

l1.insertAtEnd(7);

cout<<(l1.isFound(12) ? "Found\n" : "Not Found\n");

return 0;

}

Outcome:

Input text : “hello World”

Pattern: ello

Output: Found

Input text : “hello World”

Pattern: abc

Output: Not found

VIVA Questions:

Q1. Difference between doing the experiment with arrays and Linked list

Q2. What functions of Strings can be used

Q3 What if we use pointers in strings? Will it improve the complexity?

Q4. Can the time/space complexity be improved? If so How?

Q5. What are the other algorithms to solve the problem?

Experiment 6 : Three list are maintained with the integer roll nos of the students. One list contains the roll nos. and name of the students, second list contains the roll nos and the course name (BE, BCA etc) for the student and the third list contains the roll no. and CGPA of the student. Find a way to check the common roll nos. in the 3 lists. Note: the 3 lists are sorted by the roll nos.

Cos attained:

CO2, CO3 and CO4

**Objective:**

Objective is to be able to find intersection among 3 lists which are already sorted.

**Apparatus Used:**C/C++, Dev C++

**Program Logic:**

There are many logics to work across the problem. One method is to take 3 pointer and move them across the 3 arrays so to find the common elements. Another logic is to use Binary Search algorithm.

**Algorithm:**

1. Let the current element traversed in ar1[] be x, in ar2[] be y and in ar3[] be z. We can have following cases inside the loop.

2. If x, y and z are same, we can simply print any of them as common element and move ahead in all three arrays.

3. Else If x < y, we can move ahead in ar1[] as x cannot be a common element.

4. Else If x > z and y > z), we can simply move ahead in ar3[] as z cannot be a common element.

#include<bits/stdc++.h>

using namespace std;

int isPresent(vector<int>&arr, int k){

int n = arr.size();

int l=0, r=n-1, ans=0;

while(l<=r){

int m = (l+r)/2;

if(arr[m]==k){

ans = 1;

break;

}else if(arr[m]<k){

l=m+1;

}else{

r=m-1;

}

}

return ans;

}

void print1d(vector<int>&v){

for(auto &x : v){

cout<<x<<"";

}

cout<<"\n";

}

int main(){

vector<int>v1 = {1,2,3,4,5,8,9};

vector<int>v2 = {2,4,8,9,11,13};

vector<int>v3 = {1,2,3,5,9,12,18,18};

vector<int>ans;

int n1=v1.size(), n2=v2.size(), n3=v3.size();

int n = min({n1,n2,n3});

if(n1==n){

for(int i=0 ; i<n ; i++){

if(isPresent(v2,v1[i]) && isPresent(v3,v1[i])){

ans.push\_back(v1[i]);

}

}

}else if(n2==n){

for(int i=0 ; i<n ; i++){

if(isPresent(v1,v2[i]) && isPresent(v3,v2[i])){

ans.push\_back(v2[i]);

}

}

}else{

for(int i=0 ; i<n ; i++){

if(isPresent(v2,v3[i]) && isPresent(v1,v3[i])){

ans.push\_back(v3[i]);

}

}

}

print1d(ans);

return 0;

}

Output:

<int>v1 = {1,2,3,4,5,8,9};

vector<int>v2 = {2,4,8,9,11,13};

vector<int>v3 = {1,2,3,5,9,12,18,18};

Common elements are 2,9

VIVA Questions:

Q1. Can the merging be done using a Sort algorithm?

Q2. What is the tme complexity of the experiment

Q3 What if the elements are not sorted in the array?

Q4. Can the time/space complexity be improved? If so How?

Q5. Can we apply the same logic for characters saved in the Data Structure?

Experiment 7 : A set of strings are saved in a data structure. Represent the strings in form a non linear data structure in such a way that the searching takes the minimal time

CO attained: CO2, CO4

**Objective:**

Objective is to be able to learn about Tries.

**Apparatus Used:**C/C++, Dev C++

**Program Logic:**

Longest common Suffix in a string.

**Algorithm**

1. Insert all the words one by one in the trie. After inserting we perform a walk on the trie.
2. In this walk, go deeper until we find a node having more than 1 children(branching occurs) or 0 children (one of the string gets exhausted).
3. This is because the characters (nodes in trie) which are present in the longest common prefix must be the single child of its parent, i.e- there should not be branching in any of these nodes.

#include <bits/stdc++.h>

using namespace std;

class Node{

public:

Node \*alpha[26] = {0};

bool end = false;

bool containsKey(char ch){

return alpha[ch-'a'] != NULL;

}

void setNode(char ch, Node \*node){

alpha[ch-'a'] = node;

}

Node \*getNode(char ch){

return alpha[ch-'a'];

}

void setEnd(){

end = true;

}

bool isEnd(){

return end;

}

};

class Trie{

Node \*root;

public:

Trie(){

root = new Node();

}

void insertWord(string s){

Node \*n1 = root;

int n = s.size();

for(int i=0 ; i<n ; i++){

if( !(n1->containsKey(s[i])) ){

n1->setNode(s[i],new Node());

}

n1 = n1->getNode(s[i]);

}

n1->setEnd();

}

bool searchWord(string s){

int n = s.size();

Node \*n1 = root;

for(int i=0 ; i<n ; i++){

if( !(n1->containsKey(s[i]))){

return false;

}

n1 = n1->getNode(s[i]);

}

return n1->isEnd();

}

};

int main()

{

Trie t1;

t1.insertWord("Hello");

t1.insertWord("Hell");

t1.insertWord("Heed");

t1.insertWord("Heap");

t1.insertWord("Help");

t1.insertWord("Helo");

cout<<t1.searchWord("Hello")<<"\n";

cout<<t1.searchWord("Hell");

return 0;

}

Outcome

Text : Hello World

Pattern :

Output: found

VIVA Questions:

Q1. What is a Trie

Q2. How it is different than a Tree data structure

Q3 What other DS could have been used

Q4. Can the time/space complexity be improved? If so How?

Q5. Can we implement a tRie using Linked List

Experiment 8: Without comparing the integer values in an array, sort the array of integers in ascending order.

CO ATTAINED: CO2, CO4

**Objective:**

Objective is to be able to learn Counting sort in which unlike other sorts the elements are not compared with each other.

**Apparatus Used:**C/C++, Dev C++

**Program Logic:**

Counting sort is effective when range is not greater than number of objects to be sorted. It can be used to sort the negative input values.

**Algorithm**

countingSort(array, n) // 'n' is the size of array

1max = find maximum element in the given array

2create count array with size maximum + 1

3Initialize count array with all 0's

4for i = 0 to n

5find the count of every unique element and

6store that count at ith position in the count array

7for j = 1 to max

8Now, find the cumulative sum and store it in count array

9for i = n to 1

10Restore the array elements

11Decrease the count of every restored element by 1

12end countingSort

#include <bits/stdc++.h>

using namespace std;

void count\_sort(vector<int>&v){

map<int,int>um;

for(auto &x : v){

um[x]++;

}

int itr = 0;

for(auto &x : um){

int num = x.first;

int times = x.second;

while(times--){

v[itr++] = num;

}

}

}

void print1d(vector<int>&v){

for(auto &x : v){

cout<<x<<"";

}

cout<<"\n";

}

int main()

{

vector<int>v = {2,5,3,4,1,6,7,3,5,9};

print1d(v);

count\_sort(v);

print1d(v);

return 0;

}

Outcome:

1,2,3,4,5,6,7,8,9

VIVA Questions:

Q1. What is the name of the technique used in the sort?

Q2. What is the space complexity of the sort?

Q3. Can we sort the elements with some other Algorithm? If so how.

Q4. What is the time complexity of the algorithm?

Q5. Name other sorting techniques which are non comparable (non comparison algorithms)

Experiment 9 a: List of Marks obtained by students of your class is maintained in form a linked List. Apply a sorting algorithm with complexity O (n^2) to sort the list.

**CO Attained:** CO3 and CO5

**Objective:** Sort a given set of elements using the Quick sort method and determine the required to sort the elements. Repeat the experiment for different values of n, the number elements in the list to be sorted and plot a graph of the time taken versus n. The elements caread from a file or can be generated using the random number generator.

**Resource:** C/C++, Dev C++

**Program Logic:** Quick Sort is a Divide and Conquer algorithm. It picks an element as pivot and partitions the given array around the picked pivot.

There are many different versions of QuickSort that pick pivot in different ways.

1. Always pick the first element as a pivot.

2. Always pick last element as a pivot (implemented below)

3. Pick a random element as a pivot.

4. Pick the median as a pivot.

1 Algorithm QuickSort(p, q)

2 // Sorts the elements a[p],..., a[q] which reside in the global

3 // array a[l:n] into ascending order; a[n+1]is considered to

4 //be defined and must be >= all the elements in a[l:n].

5 {

6 if (p <q) then //If there are more than one element

7 {

8 // divide P into two sub problems.

9 j :=Partition(a,p, q + 1);

10 // j is the position of the partitioning element.

11 // Solve the sub problems.

12 QuickSort(p,j-1);

13 QuickSort(j+ l,q);

14 // There is no need for combining solutions.

15 } }

#include<bits/stdc++.h>

using namespace std;

class Node{

public:

int val;

Node \*next;

Node(){

this->val = INT\_MIN;

this->next = NULL;

}

Node(int x){

val = x;

this->next = NULL;

}

};

class LL{

Node \*head;

public:

LL(){

this->head = NULL;

}

void insertAtBegin(int x){

Node \*n1 = new Node(x);

n1->next = head;

head = n1;

}

void insertAtEnd(int x){

Node \*n1 = new Node(x);

if(head==NULL){

head=n1;

return ;

}

Node \*temp = head;

while(temp->next){

temp = temp->next;

}

temp->next = n1;

}

void sorting(){

Node \*temp1 = head, \*temp2 = head;

while(temp1->next){

temp2 = temp1->next;

while(temp2){

if(temp2->val < temp1->val){

int temp = temp2->val;

temp2->val = temp1->val;

temp1->val = temp;

}

temp2 = temp2->next;

}

temp1 = temp1->next;

}

}

void printList(){

if(!head){

cout<<"Underflow: No data to print.";

return ;

}

Node \*temp = head;

while(temp){

cout<<temp->val<<"->";

temp = temp->next;

}

}

};

int main(){

LL l1;

l1.insertAtEnd(11); // 11

l1.insertAtEnd(12); // 11 12

l1.insertAtEnd(13); // 11 12 13

l1.insertAtEnd(14); // 11 12 13 14

l1.insertAtBegin(90); // 90 11 12 13 14 15

l1.insertAtBegin(80); // 80 90 11 12 13 14 15

l1.sorting();

l1.printList();cout<<"\n"; // 11->13->14

return 0;

}

**Outcome:**

11,12, 13 ,14,80,90

VIVA Questions:

Q1. Which sort have we used in the experiment?

Q2. What is the worst case complexity of the sort?

Q3. What is the wordt case scenario of Quick Sort?

Q4 What other DS could have been used?

Q5. Can the time/space complexity be improved? If so How?

Experiment 9b : Two arrays are maintained such that the marks are in descending order. We need to reverse the two arrays using another data structure so that the arrays are in ascending order. Hint: Use LIFO

**CO Attained:** CO1, CO3, CO4

**Objective:**

To learn about the usage of a data structure using stack (LIFO). This is one of the common questions asked in placement

**Resource:** C/C++, Dev C++

**Algorithm:**

1. Let the size of A1[] be m and the size of A2[] be n.
2. Create a temporary array temp of size m and copy the contents of A1[] to it.
3. Create another array visited[] and initialize all entries in it as false. visited[] is used to mark those elements in temp[] which are copied to A1[].
4. Sort temp[]
5. Initialize the output index ind as 0.
6. Do following for every element of A2[i] in A2[]
7. Binary search for all occurrences of A2[i] in temp[], if present then copy all occurrences to A1[ind] and increment ind. Also mark the copied elements visited[]
8. Copy all unvisited elements from temp[] to A1[]

#include<bits/stdc++.h>

using namespace std;

void reverse\_using\_stack(vector<int>&v){

stack<int>st;

for(auto &x : v){

st.push(x);

}

int itr=0;

while(st.size()){

v[itr++] = st.top();

st.pop();

}

}

void print1d(vector<int>&v){

for(auto &x : v){

cout<<x<<"";

}

cout<<"\n";

}

int main(){

vector<int>v1 = {5,4,3,2,1};

vector<int>v2 = {9,8,7,6,5};

reverse\_using\_stack(v1);

reverse\_using\_stack(v2);

print1d(v1);

print1d(v2);

return 0;

}

Outcome:

1,2,3,4,5,6,7,8,9

VIVA Questions:

Q1. What data structure is best suited for the experiment?

Q2. What is the space complexity of the Experiment?

Q3. What is the time complexity of the experiment?

Q4. Can we use Array /continuous memory allocation for the experiment?

Q5. Give scenarios where Stack is Used in real life.

Experiment 10a: A thief enters a house. He is carrying a sack of some capacity say ‘m’. He can fill the sack with objects in the house and can carry a fraction of the object. The weights of each object are given by wi. For carrying each of the object in the sack, he earns a profit of pi. Write a code to maximize his profit.

**CO Attained:** CO2, CO4, CO5

**Objective:** To learn the 0-1 knapsack using Dynamic programming

**Resource:** C/C++, Dev C++

**Algorithm:**

The two sequences v = <v1, v2, …, vn> and w = <w1, w2, …, wn>

Dynamic-0-1-knapsack (v, w, n, W)

for w = 0 to W do

c[0, w] = 0

for i = 1 to n do

c[i, 0] = 0

for w = 1 to W do

if wi ≤ w then

if vi + c[i-1, w-wi] then

c[i, w] = vi + c[i-1, w-wi]

else c[i, w] = c[i-1, w]

else

c[i, w] = c[i-1, w]

CODE:

#include <bits/stdc++.h>

using namespace std;

// Returns the value of maximum profit

int knapSackRec(int W, int wt[], int val[], int i, int\*\* dp)

{

// base condition

if (i < 0)

return 0;

if (dp[i][W] != -1)

return dp[i][W];

if (wt[i] > W) {

// Store the value of function call

// stack in table before return

dp[i][W] = knapSackRec(W, wt, val, i - 1, dp);

return dp[i][W];

}

else {

// Store value in a table before return

dp[i][W] = max(val[i]

+ knapSackRec(W - wt[i], wt, val,

i - 1, dp),

knapSackRec(W, wt, val, i - 1, dp));

// Return value of table after storing

return dp[i][W];

}

}

int knapSack(int W, int wt[], int val[], int n)

{

// double pointer to declare the

// table dynamically

int\*\* dp;

dp = new int\*[n];

// loop to create the table dynamically

for (int i = 0; i < n; i++)

dp[i] = new int[W + 1];

// loop to initially filled the

// table with -1

for (int i = 0; i < n; i++)

for (int j = 0; j < W + 1; j++)

dp[i][j] = -1;

return knapSackRec(W, wt, val, n - 1, dp);

}

// Driver Code

int main()

{

int val[] = { 60, 100, 120 };

int wt[] = { 10, 20, 30 };

int W = 50;

int n = sizeof(val) / sizeof(val[0]);

cout << knapSack(W, wt, val, n);

return 0;

}

**Outcome:**

**Input:**N = 3, W = 50, profit[] = { 60, 100, 120 }, weight[] = { 10, 20, 30 }  
**Output:**240

VIVA Questions:

Q1. What is the approach used in the experiment?

Q2. Can Knapsack be applied only for 0-1 ?

Q3 How can we check that the profit obtained is the maximum

Q4. Can the time/space complexity be improved? If so How?

Q5. Can the problem be solved using Greedy Approach?

Experiment 10 b: Write a code for the experiment above with the case that the thief cannot choose any object in its fraction. He can either pick the object or he can leave the object.

CO Attained: CO3, CO4

**Objective:** To learn the 0-1 knapsack using Greedy method. Here unlike 0-1 , we can pick the elements in fractions.

**Resource:** C/C++, Dev C++

**Algorithm:**

1. Calculate the ratio(value/weight) for each item.
2. Sort all the items in decreasing order of the ratio.
3. Initialize res =0, curr\_cap = given\_cap.
4. Do the following for every item “i” in the sorted order:
5. If the weight of the current item is less than or equal to the remaining capacity then add the value of that item into the result
6. Else add the current item as much as we can and break out of the loop.
7. Return res.

**Code:**

#include <stdio.h>

int n = 5; /\* The number of objects \*/

int c[10] = {12, 1, 2, 1, 4}; /\* c[i] is the \*COST\* of the ith object; i.e. what

YOU PAY to take the object \*/

int v[10] = {4, 2, 2, 1, 10}; /\* v[i] is the \*VALUE\* of the ith object; i.e.

what YOU GET for taking the object \*/

int W = 15; /\* The maximum weight you can take \*/

void simple\_fill() {

int cur\_w;

float tot\_v;

int i, maxi;

int used[10];

for (i = 0; i < n; ++i)

used[i] = 0; /\* I have not used the ith object yet \*/

cur\_w = W;

while (cur\_w > 0) { /\* while there's still room\*/

/\* Find the best object \*/

maxi = -1;

for (i = 0; i < n; ++i)

if ((used[i] == 0) &&

((maxi == -1) || ((float)v[i]/c[i] > (float)v[maxi]/c[maxi])))

maxi = i;

used[maxi] = 1; /\* mark the maxi-th object as used \*/

cur\_w -= c[maxi]; /\* with the object in the bag, I can carry less \*/

tot\_v += v[maxi];

if (cur\_w >= 0)

printf("Added object %d (%d$, %dKg) completely in the bag. Space left: %d.\n", maxi + 1, v[maxi], c[maxi], cur\_w);

else {

printf("Added %d%% (%d$, %dKg) of object %d in the bag.\n", (int)((1 + (float)cur\_w/c[maxi]) \* 100), v[maxi], c[maxi], maxi + 1);

tot\_v -= v[maxi];

tot\_v += (1 + (float)cur\_w/c[maxi]) \* v[maxi];

}

}

printf("Filled the bag with objects worth %.2f$.\n", tot\_v);

}

int main(int argc, char \*argv[]) {

simple\_fill();

return 0;

}

**Outcome:**

***Input:****arr[] = {{60, 10}, {100, 20}, {120, 30}}, W = 50****Output:****240****Explanation:****By taking items of weight 10 and 20 kg and 2/3 fraction of 30 kg.   
Hence total price will be 60+100+(2/3)(120) = 240*

**VIVA Questions:**

Q1. What is the approach used in the experiment?

Q2. Can Knapsack be applied only for 0-1 ?

Q3 How can we check that the profit obtained is the maximum

Q4. Can the time/space complexity be improved? If so How?

Q5. Can we solve the problem using Dynamic Programming

Experiment 10 c: Implement a Stack using Two Queues. Namely Push and PoP operations

**CO Attained :** CO2, CO3

**Objective:** To learn about the stacks and Queues operations. That is to be able to delete elements from the stack and simultaneously putting those into queue

**Resource:** C/C++, Dev C++

**Algorithm:**

Follow the below steps to implement the push(s, x) operation:

Enqueue x to q1 (assuming the size of q1 is unlimited).

Follow the below steps to implement the pop(s) operation:

One by one dequeue everything except the last element from q1 and enqueue to q2.

Dequeue the last item of q1, the dequeued item is the result, store it.

Swap the names of q1 and q2

Return the item stored in step 2.

**Code:**

#include <stdio.h>

#include <stdlib.h>

void push1(int);

void push2(int);

int pop1();

int pop2();

void enqueue();

void dequeue();

void display();

void create();

int st1[100], st2[100];

int top1 = -1, top2 = -1;

int count = 0;

void main()

{

int ch;

printf("\n1 - Enqueue element into queue");

printf("\n2 - Dequeu element from queue");

printf("\n3 - Display from queue");

printf("\n4 - Exit");

create();

while (1)

{

printf("\nEnter choice");

scanf("%d", &ch);

switch (ch)

{

case 1:

enqueue();

break;

case 2:

dequeue();

break;

case 3:

display();

break;

case 4:

exit(0);

default:

printf("Wrong choice");

}

}

}

/\*Function to create a queue\*/

void create()

{

top1 = top2 = -1;

}

/\*Function to push the element on to the stack\*/

void push1(int data)

{

st1[++top1] = data;

}

/\*Function to pop the element from the stack\*/

int pop1()

{

return(st1[top1--]);

}

/\*Function to push an element on to stack\*/

void push2(int data)

{

st2[++top2] = data;

}

/\*Function to pop an element from th stack\*/

int pop2()

{

return(st2[top2--]);

}

/\*Function to add an element into the queue using stack\*/

void enqueue()

{

int data, i;

printf("Enter data into queue");

scanf("%d", &data);

push1(data);

count++;

}

/\*Function to delete an element from the queue using stack\*/

void dequeue()

{

int i;

for (i = 0;i <= count;i++)

{

push2(pop1());

}

pop2();

count--;

for (i = 0;i <= count;i++)

{

push1(pop2());

}

}

/\*Function to display the elements in the stack\*/

void display()

{

int i;

for (i = 0;i <= top1;i++)

{

printf(" %d ", st1[i]);

}

}

Outcome:

current size: 3

3

2

1

current size: 1

VIVA Questions:

Q1. How are elements inserted in a stack?

Q2. Can we implement stack using a Linked list?

Q3 What are the ways of deleting an element from the stack

Q4. Name few real time applications of stack.

Q5. What is the difference between Enquue and Dequeue algorithm of stack.

Experiment 10d: Flood fill (also known as seed fill) is an algorithm that determines the area connected to a given node in a multi-dimensional array. It is used in the “bucket” fill tool of a paint program to fill connected, similarly colored areas with a different color and in games such as Go and Minesweeper for determining which pieces are cleared. When applied on an image to fill a particular bounded area with color, it is also known as boundary fill.

CO Attained: CO2, CO3, CO4

**Objective:** To learn the flood fill algorithm used mainly in games like Minesweeper

**Resource:** C/C++, Dev C++

**Algorithm:**

* Create an empty [queue](https://www.geeksforgeeks.org/queue-data-structure/) lets say Q.
* Push the starting location of the pixel as given in the input and apply replacement color to it.
* Iterate until **Q** is not empty and pop the front node (pixel position).
* Check the pixels adjacent to the current pixel and push into the queue if valid (had not been colored with replacement color and have the same color as the old color).

#include <iostream>

#include <queue>

#include <iomanip>

using namespace std;

// Below arrays detail all eight possible movements

int row[] = { -1, -1, -1, 0, 0, 1, 1, 1 };

int col[] = { -1, 0, 1, -1, 1, -1, 0, 1 };

// check if it is possible to go to pixel (x, y) from the

// current pixel. The function returns false if the pixel

// has a different color, or it's not a valid pixel

bool isSafe(vector<vector<char>> const &mat, int x, int y, char target)

{

return (x >= 0 && x < mat.size()) && (y >= 0 && y < mat[0].size())

&& mat[x][y] == target;

}

// Flood fill using BFS

void floodfill(vector<vector<char>> &mat, int x, int y, char replacement)

{

// base case

if (mat.size() == 0) {

return;

}

// create a queue and enqueue starting pixel

queue<pair<int, int>> q;

q.push({x, y});

// get the target color

char target = mat[x][y];

// target color is same as replacement

if (target == replacement) {

return;

}

// break when the queue becomes empty

while (!q.empty())

{

// dequeue front node and process it

pair<int, int> node = q.front();

q.pop();

// (x, y) represents the current pixel

int x = node.first, y = node.second;

// replace the current pixel color with that of replacement

mat[x][y] = replacement;

// process all eight adjacent pixels of the current pixel and

// enqueue each valid pixel

for (int k = 0; k < 8; k++)

{

// if the adjacent pixel at position (x + row[k], y + col[k]) is

// is valid and has the same color as the current pixel

if (isSafe(mat, x + row[k], y + col[k], target))

{

// enqueue adjacent pixel

q.push({x + row[k], y + col[k]});

}

}

}

}

// Utility function to print a matrix

void printMatrix(vector<vector<char>> const &mat)

{

for (int i = 0; i < mat.size(); i++)

{

for (int j = 0; j < mat[0].size(); j++) {

cout << setw(3) << mat[i][j];

}

cout << endl;

}

}

int main()

{

// matrix showing portion of the screen having different colors

vector<vector<char>> mat =

{

{ 'Y', 'Y', 'Y', 'G', 'G', 'G', 'G', 'G', 'G', 'G' },

{ 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'G', 'X', 'X', 'X' },

{ 'G', 'G', 'G', 'G', 'G', 'G', 'G', 'X', 'X', 'X' },

{ 'W', 'W', 'W', 'W', 'W', 'G', 'G', 'G', 'G', 'X' },

{ 'W', 'R', 'R', 'R', 'R', 'R', 'G', 'X', 'X', 'X' },

{ 'W', 'W', 'W', 'R', 'R', 'G', 'G', 'X', 'X', 'X' },

{ 'W', 'B', 'W', 'R', 'R', 'R', 'R', 'R', 'R', 'X' },

{ 'W', 'B', 'B', 'B', 'B', 'R', 'R', 'X', 'X', 'X' },

{ 'W', 'B', 'B', 'X', 'B', 'B', 'B', 'B', 'X', 'X' },

{ 'W', 'B', 'B', 'X', 'X', 'X', 'X', 'X', 'X', 'X' }

};

// start node

int x = 3, y = 9; // having target color `X`

// replacement color

char replacement = 'C';

// replace the target color with a replacement color

floodfill(mat, x, y, replacement);

// print the colors after replacement

printMatrix(mat);

return 0;

}

Outcome:

Input: arr[][] = {

{1, 1, 1, 1, 1, 1, 1, 1},

{1, 1, 1, 1, 1, 1, 0, 0},

{1, 0, 0, 1, 1, 0, 1, 1},

{1, 2, 2, 2, 2, 0, 1, 0},

{1, 1, 1, 2, 2, 0, 1, 0},

{1, 1, 1, 2, 2, 2, 2, 0},

{1, 1, 1, 1, 1, 2, 1, 1},

{1, 1, 1, 1, 1, 2, 2, 1}}

X = 4, Y = 4, C = 3

Output:

1 1 1 1 1 1 1 1

1 1 1 1 1 1 0 0

1 0 0 1 1 0 1 1

1 3 3 3 3 0 1 0

1 1 1 3 3 0 1 0

1 1 1 3 3 3 3 0

1 1 1 1 1 3 1 1

1 1 1 1 1 3 3 1

Viva Questions:

Q1. Under what approach of Algorithm design does flood fill algorithm fall?

Q2. Can we use the algorithm for Google Maps , if so how?

Q3. What other applications of flood fill algorithm we can use?

Q4. What is the significance of 0s and 1 s in the algorithm

Q5. What is the time/space complexity of the algorithm?

Experiment 10: The hierarchy of a family is to be maintained in such a way that the starting from Great grandfather, we need to depict till the grandsons/granddaughters. Note that at no level, the parent has more than 2 children. Depict such hierarchy using a suitable data structure. In such data structure, apply the following traversals and Count the number of Leaf and Non-leaf nodes:

1. In-order
2. Pre-order
3. Post-order
4. Level Order
5. Diagonal Order

**CO Attained**: CO1, CO2, CO3, CO4, CO5

**Objective:** To learn about the various traversals in a Tree.

**Resource:** C/C++, Dev C++

**Algorithm:**

In-oder

1. Traverse the left subtree, i.e., call Inorder(left->subtree)
2. Visit the root.
3. Traverse the right subtree, i.e., call Inorder(right->subtree)

Pre-order

1. Visit the root.
2. Traverse the left subtree, i.e., call Preorder(left->subtree)
3. Traverse the right subtree, i.e., call Preorder(right->subtree)

Post-Order

1. Traverse the left subtree, i.e., call Postorder(left->subtree)
2. Traverse the right subtree, i.e., call Postorder(right->subtree)
3. Visit the root

Level Order

1. Run a for loop for counter i, i.e. current height from 1 to h (height of the tree).
2. Use DFS to traverse the tree and maintain height for the current node.
3. If the Node is NULL then return;
4. If level is 1 print(tree->data);
5. Else if the level is greater than 1, then
6. Recursively call to for tree->left, level-1.
7. Recursively call to for tree->right, level-1.

**Code:**

In-order

#include <stdio.h>

#include <stdlib.h>

/\* A binary tree node has data, pointer to left child

and a pointer to right child \*/

struct node {

int data;

struct node\* left;

struct node\* right;

};

/\* Helper function that allocates a new node with the

given data and NULL left and right pointers. \*/

struct node\* newNode(int data)

{

struct node\* node

= (struct node\*)malloc(sizeof(struct node));

node->data = data;

node->left = NULL;

node->right = NULL;

return (node);

}

/\* Given a binary tree, print its nodes in inorder\*/

void printInorder(struct node\* node)

{

if (node == NULL)

return;

/\* first recur on left child \*/

printInorder(node->left);

/\* then print the data of node \*/

printf("%d ", node->data);

/\* now recur on right child \*/

printInorder(node->right);

}

/\* Driver code\*/

int main()

{

struct node\* root = newNode(1);

root->left = newNode(2);

root->right = newNode(3);

root->left->left = newNode(4);

root->left->right = newNode(5);

// Function call

printf("\nInorder traversal of binary tree is \n");

printInorder(root);

getchar();

return 0;

}

Pre-Order

#include <stdio.h>

#include <stdlib.h>

/\* A binary tree node has data, pointer to left child

and a pointer to right child \*/

struct node {

int data;

struct node\* left;

struct node\* right;

};

/\* Helper function that allocates a new node with the

given data and NULL left and right pointers. \*/

struct node\* newNode(int data)

{

struct node\* node

= (struct node\*)malloc(sizeof(struct node));

node->data = data;

node->left = NULL;

node->right = NULL;

return (node);

}

/\* Given a binary tree, print its nodes in inorder\*/

void printInorder(struct node\* node)

{

if (node == NULL)

return;

/\* first recur on left child \*/

printInorder(node->left);

/\* then print the data of node \*/

printf("%d ", node->data);

/\* now recur on right child \*/

printInorder(node->right);

}

/\* Driver code\*/

int main()

{

struct node\* root = newNode(1);

root->left = newNode(2);

root->right = newNode(3);

root->left->left = newNode(4);

root->left->right = newNode(5);

// Function call

printf("\nInorder traversal of binary tree is \n");

printInorder(root);

getchar();

return 0;

}

Post –Order

#include <stdio.h>

#include <stdlib.h>

/\* A binary tree node has data, pointer to left child

and a pointer to right child \*/

struct node {

int data;

struct node\* left;

struct node\* right;

};

/\* Helper function that allocates a new node with the

given data and NULL left and right pointers. \*/

struct node\* newNode(int data)

{

struct node\* node

= (struct node\*)malloc(sizeof(struct node));

node->data = data;

node->left = NULL;

node->right = NULL;

return (node);

}

/\* Given a binary tree, print its nodes according to the

"bottom-up" postorder traversal. \*/

void printPostorder(struct node\* node)

{

if (node == NULL)

return;

// first recur on left subtree

printPostorder(node->left);

// then recur on right subtree

printPostorder(node->right);

// now deal with the node

printf("%d ", node->data);

}

/\* Driver code\*/

int main()

{

struct node\* root = newNode(1);

root->left = newNode(2);

root->right = newNode(3);

root->left->left = newNode(4);

root->left->right = newNode(5);

// Function call

printf("\nPostorder traversal of binary tree is \n");

printPostorder(root);

getchar();

return 0;

}

Level Order

#include <stdio.h>

#include <stdlib.h>

/\* A binary tree node has data,

pointer to left child

and a pointer to right child \*/

struct node {

int data;

struct node \*left, \*right;

};

/\* Function prototypes \*/

void printCurrentLevel(struct node\* root, int level);

int height(struct node\* node);

struct node\* newNode(int data);

/\* Function to print level order traversal a tree\*/

void printLevelOrder(struct node\* root)

{

int h = height(root);

int i;

for (i = 1; i <= h; i++)

printCurrentLevel(root, i);

}

/\* Print nodes at a current level \*/

void printCurrentLevel(struct node\* root, int level)

{

if (root == NULL)

return;

if (level == 1)

printf("%d ", root->data);

else if (level > 1) {

printCurrentLevel(root->left, level - 1);

printCurrentLevel(root->right, level - 1);

}

}

/\* Compute the "height" of a tree -- the number of

nodes along the longest path from the root node

down to the farthest leaf node.\*/

int height(struct node\* node)

{

if (node == NULL)

return 0;

else {

/\* compute the height of each subtree \*/

int lheight = height(node->left);

int rheight = height(node->right);

/\* use the larger one \*/

if (lheight > rheight)

return (lheight + 1);

else

return (rheight + 1);

}

}

/\* Helper function that allocates a new node with the

given data and NULL left and right pointers. \*/

struct node\* newNode(int data)

{

struct node\* node

= (struct node\*)malloc(sizeof(struct node));

node->data = data;

node->left = NULL;

node->right = NULL;

return (node);

}

/\* Driver program to test above functions\*/

int main()

{

struct node\* root = newNode(1);

root->left = newNode(2);

root->right = newNode(3);

root->left->left = newNode(4);

root->left->right = newNode(5);

printf("Level Order traversal of binary tree is \n");

printLevelOrder(root);

return 0;

}

Diagonal Order

#include <bits/stdc++.h>

using namespace std;

// Tree node

struct Node

{

int data;

Node \*left, \*right;

};

/\* root - root of the binary tree

d - distance of current line from rightmost

-topmost slope.

diagonalPrint - multimap to store Diagonal

elements (Passed by Reference) \*/

void diagonalPrintUtil(Node\* root, int d,

map<int, vector<int>> &diagonalPrint)

{

// Base case

if (!root)

return;

// Store all nodes of same

// line together as a vector

diagonalPrint[d].push\_back(root->data);

// Increase the vertical

// distance if left child

diagonalPrintUtil(root->left,

d + 1, diagonalPrint);

// Vertical distance remains

// same for right child

diagonalPrintUtil(root->right,

d, diagonalPrint);

}

// Print diagonal traversal

// of given binary tree

void diagonalPrint(Node\* root)

{

// create a map of vectors

// to store Diagonal elements

map<int, vector<int> > diagonalPrint;

diagonalPrintUtil(root, 0, diagonalPrint);

cout << "Diagonal Traversal of binary tree : \n";

for (auto it :diagonalPrint)

{

vector<int> v=it.second;

for(auto it:v)

cout<<it<<" ";

cout<<endl;

}

}

// Utility method to create a new node

Node\* newNode(int data)

{

Node\* node = new Node;

node->data = data;

node->left = node->right = NULL;

return node;

}

// Driver program

int main()

{

Node\* root = newNode(8);

root->left = newNode(3);

root->right = newNode(10);

root->left->left = newNode(1);

root->left->right = newNode(6);

root->right->right = newNode(14);

root->right->right->left = newNode(13);

root->left->right->left = newNode(4);

root->left->right->right = newNode(7);

/\* Node\* root = newNode(1);

root->left = newNode(2);

root->right = newNode(3);

root->left->left = newNode(9);

root->left->right = newNode(6);

root->right->left = newNode(4);

root->right->right = newNode(5);

root->right->left->right = newNode(7);

root->right->left->left = newNode(12);

root->left->right->left = newNode(11);

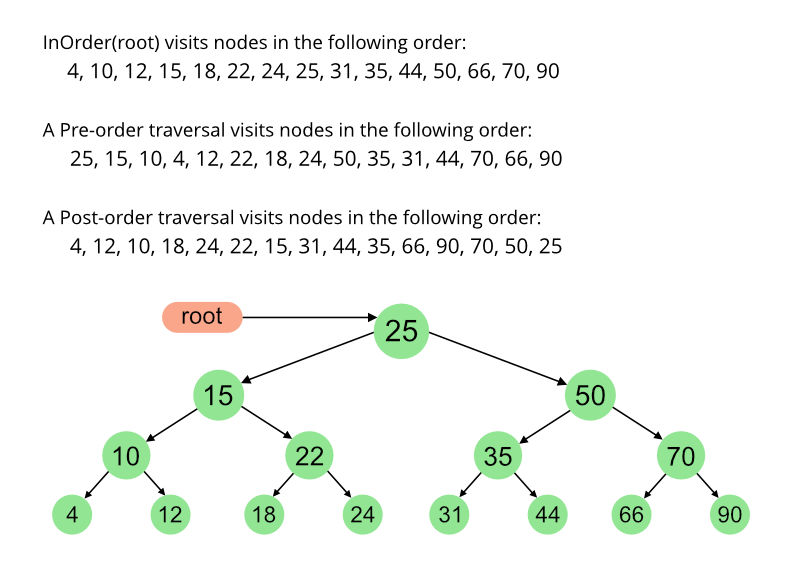
root->left->left->right = newNode(10);\*/

diagonalPrint(root);

return 0;

}

Outcome:



VIVA Questions:

Q1. What is the approach used in the experiment?

Q2. How are the traversals of graph related to each other?

Q3 Give a real time scenario where each of these traversals can be use.

Q4. Can the time/space complexity be improved? If so How?

Q5. Give real time scenarios of Level order and diagonal order traversal of tree.

**LAB MST**

**Question Paper (Mapped with COs)**

**Scheme of Evaluation**

1. Set of Unique questions need to be prepared and distributed among the students.

2. Students need to take their seats according to their UIDs in Sequential fashion.

3. No student can change their allocated seat during the practical, to ensure this, you need to make all the computers in working mode. Ask the respective Lab Instructors to install the required Operating System/ Application Software in your guidance.

4. Faculty will visit to respective student desk for the evaluating the Lab components, as mentioned in the bullet 5.

5. Lab Evaluation Components are

a) Worksheet - 5 Marks

b) Conduct - 5 Marks

c) Viva - 10 Marks

Total Marks: 20

**Sample Answer Sheets**

**CO-Wise Mark Sheets**

Group A

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Sr. No.** | **Student UID** | **Student Name** | **Conduct (CO1)** | **Worksheet (CO3)** | **Viva-voce (CO2)** | **Total (20 Marks)** |
| **(05 marks)** | **(05 marks)** | **(10 marks)** |
| 1 |  |  |  |  |  |  |
| 2 |  |  |  |  |  |  |
| 3 |  |  |  |  |  |  |

Group B

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Sr. No.** | **Student UID** | **Student Name** | **Conduct (CO1)** | **Worksheet (CO3)** | **Viva-voce (CO2)** | **Total (20 Marks)** |
| **(05 marks)** | **(05 marks)** | **(10 marks)** |
| 1 |  |  |  |  |  |  |
| 2 |  |  |  |  |  |  |
| 3 |  |  |  |  |  |  |

**List of Slow Learners**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **UID** | **Name** | **MarksObtd\_Percentage** | **MarksObtd** | **MaxMarks** |
|  |  |  |  |  |
|  |  |  |  |  |

**Action Taken**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Schedule for Remedial Classes  Aug-Dec 2022** | | | | | | |
| **S.No** | **Day** | **Subject CODE** | **Subject Name** | **Room No** | **Faculty Name** | **Timing** |
|  |  |  |  |  |  |  |

**Best Practices Adopted**

1. Introductory first session about the subject .
2. Post experiment discussions.
3. Real World to The Work /Instructor Lead Hands on Learning through Lab work.
4. Self-Assessment Questionnaires. (Through Assignment Task)
5. Peer Teaching
6. Promote Student Engagement through discussions and presentations.
7. Case Studies

**Final Award Sheet**

|  |  |  |  |
| --- | --- | --- | --- |
| **UID** | **CO2 MM:10** | **CO3 MM:15** | **CO4 MM:15** |
|  |  |  |  |

**List of Slow Learners**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **UID** | **Name** | **MarksObtd\_Percentage** | **MarksObtd** | **MaxMarks** |
|  |  |  |  |  |

**Overall Observation and Action Taken (Recommendation to BOS)**