SPPU

Stacks

Queues

According to New Revised Credit System Syllabus

Second Year Degree Course In

INFORMATION TECHNOLOGY (Sem - II)

DATA STRUCTURES AND FILES

Includes

- Sample Ques. Papers for Theory Exams (50 Marks)

Dr. SACHIN R. SAKHARE
NITIN N. SAKHARE

www.pragationline.com
www.facebook.com/niralibooks

NIRALI PRAKASHAN
ADVANCEMENT OF KNOWLEDGE

A TEXT BOOK OF

DATA STRUCTURES AND FILES

FOR
SEMESTER – II

SECOND YEAR DEGREE COURSE IN INFORMATION TECHNOLOGY

Strictly According to New Revised Credit System Syllabus of Savitribai Phule Pune University

(w.e.f June 2016)

Dr. SACHIN R. SAKHARE
Ph. D. (Comp. Sci. & Engg.)
Professor & Head,
Computer Engineering Department,
Vishwakarma Institute of Inform. Technology
Kondhwa (Bk), Pune.

NITIN N. SAKHARE
M. E. (Comp. Networks)
Assistant Professor,
Computer Engineering Department,
Vishwakarma Institute of Inform. Technology
Kondhwa Bk., Pune

Price ₹ 350.00

NIRALI PRAKASHAN
ADVANCEMENT OF KNOWLEDGE

N 3585

DATA STRUCTURES AND FILES (SE IT) **ISBN 978-93-86353-17-7**

First Edition : January 2017

Published By :
NIRALI PRAKASHAN
Abhyudaya Pragati, 1312, Shivaji Nagar,
Off J.M. Road, Pune – 411005
Tel - (020) 25512336/37/39, Fax - (020) 25511379
Email : niralipune@pragationline.com

Polyplate

Printed By :
SHREE OM PRINTERS PVT. LTD
Survey No. 28/25, Dhayri Near Pari Company
PUNE - 411 041
Tel - (020) 24690371

☞ DISTRIBUTION CENTRES

PUNE

Nirali Prakashan : 119, Budhwar Peth, Jogeshwari Mandir Lane, Pune 411002, Maharashtra
Tel : (020) 2445 2044, 66022708, Fax : (020) 2445 1538
Email : bookorder@pragationline.com, niralilocal@pragationline.com

Nirali Prakashan : S. No. 28/27, Dhyari, Near Pari Company, Pune 411041
Tel : (020) 24690204 Fax : (020) 24690316
Email : dhyari@pragationline.com, bookorder@pragationline.com

MUMBAI

Nirali Prakashan : 385, S.V.P. Road, Rasdhara Co-op. Hsg. Society Ltd.,
Girgaum, Mumbai 400004, Maharashtra
Tel : (022) 2385 6339 / 2386 9976, Fax : (022) 2386 9976
Email : niralimumbai@pragationline.com

☞ DISTRIBUTION BRANCHES

JALGAON

Nirali Prakashan : 34, V. V. Golani Market, Navi Peth, Jalgaon 425001,
Maharashtra, Tel : (0257) 222 0395, Mob : 94234 91860

KOLHAPUR

Nirali Prakashan : New Mahadvar Road, Kedar Plaza, 1st Floor Opp. IDBI Bank
Kolhapur 416 012, Maharashtra. Mob : 9850046155

NAGPUR

Pratibha Book Distributors : Above Maratha Mandir, Shop No. 3, First Floor,
Rani Jhanshi Square, Sitabuldi, Nagpur 440012, Maharashtra
Tel : (0712) 254 7129

DELHI

Nirali Prakashan : 4593/21, Basement, Aggarwal Lane 15, Ansari Road, Daryaganj
Near Times of India Building, New Delhi 110002
Mob : 08505972553

BENGALURU

Pragati Book House : House No. 1, Sanjeevappa Lane, Avenue Road Cross,
Opp. Rice Church, Bengaluru – 560002.
Tel : (080) 64513344, 64513355,Mob : 9880582331, 9845021552
Email:bharatsavla@yahoo.com

CHENNAI

Pragati Books : 9/1, Montieth Road, Behind Taas Mahal, Egmore,
Chennai 600008 Tamil Nadu, Tel : (044) 6518 3535,
Mob : 94440 01782 / 98450 21552 / 98805 82331,
Email : bharatsavla@yahoo.com

PREFACE

It gives us great pleasure in publishing this text book on "**Data Structure and Files**" for the students of Second Year Degree Course in Information Technology. This book is strictly written according to **New Revised Credit System Syllabus** of Savitribai Phule Pune University (2015 Pattern).

As per the policy of the University, Engineering Syllabi is revised every five years. Last revision was in the year 2012. New revision is coming little earlier, as university has introduced **Online System of Examination** from year 2012.

As per the **New Credit System**, the **Online Examinations** Phase-I will be conducted based on First & Second Units and Phase II on Third & Fourth Units. The **Online** examinations will have objective types of questions with multiple choices. End Sem. Theory Examination will be based on all the six units and that will be conducted in traditional way and the Theory Course will have 4 credits.

It is our objective to keep the presentation systematic, consistent, intensive and clear presentation of concept through explanatory notes and figures. So we are sure that this book will cater for all your needs for this subject.

Main feature of this book is, **Complete Coverage** of the New Credit System Syllabus with large number of **Worked (Solved) Programs Examples and Exercises.**

We have given Separate Book of Multiple Choice Questions (MCQ's) which will be very useful to the students especially for Online Examinations.

We take this opportunity to express our sincere thanks to Shri. Dineshbhai Furia, Shri. Jignesh Furia, Mrs. Nirali Verma and Shri. M. P. Munde and entire team of Nirali Prakashan namely Mrs. Deepali Lachake (Co-ordinator), who really have taken keen interest and untiring efforts in publishing this text.

The advice and suggestions of our esteemed readers to improve the text are most welcomed, and will be highly appreciated.

Pune **Authors**

SYLLABUS

Unit I : Stacks and Queues **8 Hours**

Concept of stack, stack as ADT, Implementation of stack using linked organization. Concept of implicit and explicit stack, Applications of stack.

Concept of queues as ADT, Implementation of queue using linked organization. Concept of circular queue, double ended queue and priority queue. Applications of queues.

Unit II : Trees **10 Hours**

Difference in linear and non-linear data structure, Trees and binary trees-concept and terminology. Expression tree. Conversion of general tree to binary tree. Binary tree as an ADT. Recursive and non-recursive algorithms for binary tree traversals, Binary search trees, Binary search tree as ADT, Applications of trees

Unit III : Graphs **8 Hours**

Graph as an ADT, Representation of graphs using adjacency matrix and adjacency list, Depth First Search and Breadth First Search traversal. Prim's and Kruskal's algorithms for minimum spanning tree, shortest path using Warshall's and Dijkstra's algorithm, topological sorting.

Unit IV : Tables **8 Hours**

Symbol Table: Notion of Symbol Table, OBST, Huffman's algorithm, Heap data structure, Min and Max Heap, Heap sort implementation, applications of heap

Hash tables and scattered tables: Basic concepts, hash function, characteristics of good hash function, different key-to-address transformations techniques, synonyms or collisions, collision resolution techniques- linear probing, quadratic probing, rehashing, chaining without replacement and chaining with replacement

Unit V : Advance Trees **7 Hours**

Concept of threaded binary tree. Preorder and In-order traversals of in-order threaded binary tree, Concept of red and black trees, AVL Trees, B trees, B+ trees, Splay trees

Unit VI : File Organization **7 Hours**

External storage devices, File, File types and file organization (sequential, index sequential and Direct access), Primitive operations and implementations for each type and comparison

CONTENTS

Unit I : Stacks and Queues

Unit II : Trees

Unit I

CHAPTER 1
STACKS

1.1 INTRODUCTION

Stack is a linear data structure where the element which is inserted last can only be taken out first. Thus, it is called LIFO (Last In First Out) type of data structure.

Stack is also defined as a data structure where all addition and deletion are made only at one end called top. It is similar to a real life situation of stack of things. If we keep things stacked one on to the another, we can take out the thing at the top. Similarly, we can keep a new thing on the top.

Stack can be implemented using :

1. Arrays
2. Linked lists

Following four operations can be done on a stack :

1. **Push Operation :** In this, an element is stored at a location indicated by top.
2. **Pop Operation :** In this operation, the element at the top is removed.
3. **Stack Full :** When all the locations reserved for stack are occupied, we can't insert any more elements. This condition is called as stack full condition.
4. **Stack Empty :** When there is no element left on a stack, we can't pop any element. This condition is called as stack empty condition.

1.2 STACK USING ARRAY

To represent a stack using array, we require an array of some size to be declared say int a[4]. This will create a space for storing elements on stack. The size of the stack is 4. We will require one more variable say top which will be an index to the top element of the stack. Initially, the stack is empty. The variable top will be initialized to −1.

Push Operation :

Now, if we want to store a number 10 on the stack, we can increment top and the element 10 will be stored at location a[0].

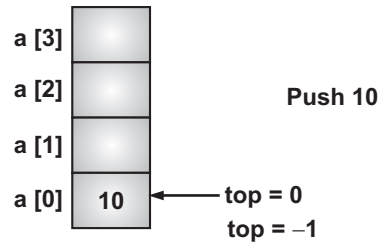


Fig. 1.1 (a) : Push (10)

Next, we store a number 20 on the stack, top will be incremented to 1 and element 20 will be stored at location a[1].

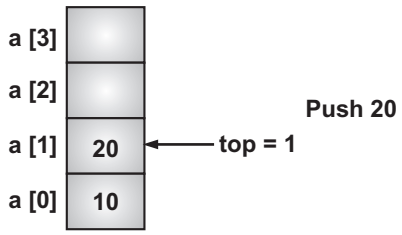


Fig. 1.1 (b) : Push (20)

Next, we store a number 30 on the stack, top will be incremented to 2 and element 30 will be stored at location a[2].

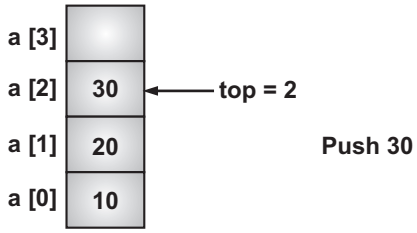


Fig. 1.1 (c) : Push (30)

Next, we store a number 40 on the stack, top will be incremented to 3 and element 40 will be stored at location a[3].

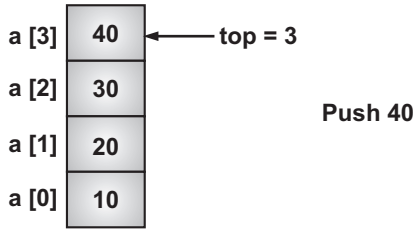


Fig. 1.1 (d) : Push (40)

Now there is no space left on the stack, hence, the stack is full. The condition for stack full is top becomes equal to maximum size of array –1.

Pop Operation :

Now, suppose we want to remove an element from the stack, we can access element at the top which is given by the index value in top. Consider the stack where we have already pushed four elements. If we carry out pop operation, the element at the top i.e. 40 will be accessed and top is decremented to 2.

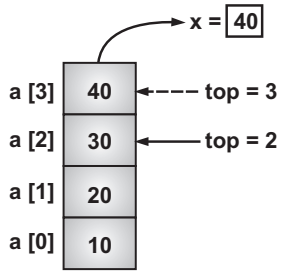


Fig. 1.2 (a) : x = pop( )

The next pop operation will remove 30 from the stack and top will be decremented to 1.

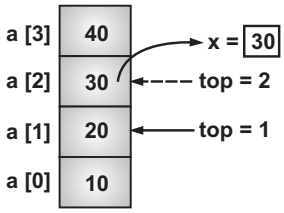


Fig. 1.2 (b) : x = pop( )

Another pop operation will remove 20 from the stack and top will be decremented to 0.

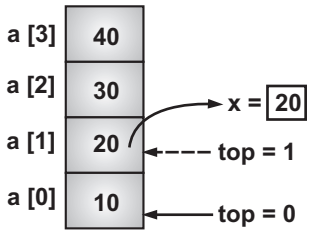


Fig. 1.2 (c) : x = pop( )

If the operation is carried out again, element 10 will be removed and top becomes −1.

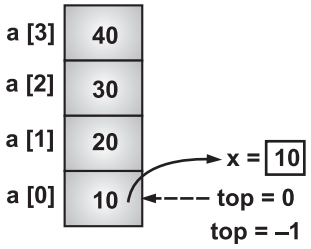


Fig. 1.2 (d) : x = pop( )

Then we can't remove any more elements because stack is empty. The condition for stack empty is top = −1. The program for implementation of stack will require two functions push and pop whose algorithms areas follows :

1. Push (x) :

```
(i)   If top == MAX –1
          print "stack full";
(ii)  else
          top ++;
          a [top] = x;
(iii) Return.
```

Every time we do a push operation, we need to increment top and store the data at the location given by top in the array. But before we do this operation, we must check whether the stack is full or not. Hence, the condition top == MAX −1.

2. x = pop( ) :

```
(i)   If top==–1
          print "stack empty";
          return –9999;
(ii)  else
          x = a [top]
          top --;
          return x;
```

Every time we do a pop operation, we remove an element and then decrement top. But before we do this, we must check whether the stack is empty or not. Hence, the condition top == −1. Note that we are returning −9999 when stack is empty. This is an indication for the calling function, so that it takes appropriate action when the stack is empty. The complete program for stack implementation is as follows :

Program 1.1 : To implement stack using array (Version 1)

```
#define MAX 5
int a [MAX];
int top = –1;
void push (int x)
{
    if (top == MAX – 1)
```

```
            printf("Stack is full");
        else
        {
        top ++;
        a[top]=x;
        }
}
int pop( )
{
        int x;
        if (top == –1)
        {
            print("Stack is empty \n");
            return (–9999);
        }
        else
        {
            x = a[top];
            top ––;
            return (x);
        }
}
main( )
{
        int ch, x;
        do
        {
            clrscr( );
            printf(" 1. Push \n. 2. Pop \n. 3. Exit \n");
            printf("Enter your choice \n");
```

```
            scanf("%d", &ch);
            switch (ch)
            {
                case 1 : printf("Enter a number \n");
                         scanf("% d", &x);
                         push (x);
                         break;
                case 2 :    x = pop( );
                         if (x!= –9999)
                              printf("% d", x);
            }
            getch( );
        } while (ch!=3);
    }
```

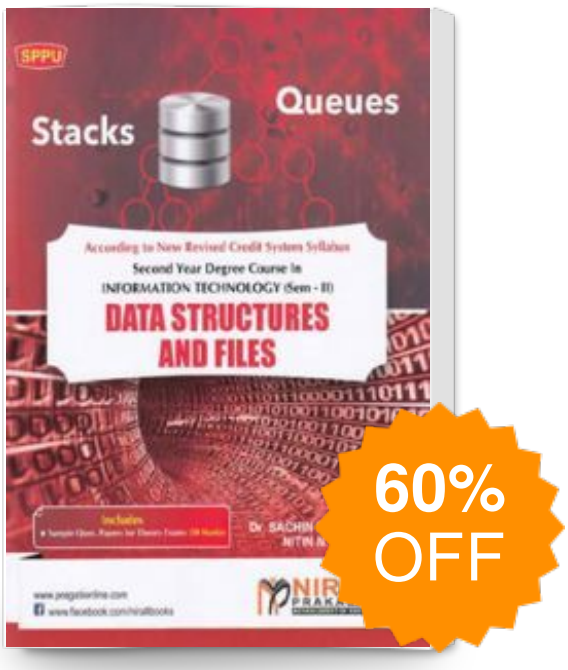
We can implement separate functions for stack full and stack empty conditions as follows :

```
    int stk_full( )
    {
        if (top == MAX –1)
        {
            printf("Stack full");
            return (1);
        }
        else
            return (0);
    }
```

The function returns 1 when stack is full otherwise 0.

```
    int stk_empty( )
    {
        if (top == –1)
        {
```

Data Structure And Files



Publisher : Nirali Prakashan ISBN : 9789386353177

Author : Dr. Sachin R. Sakhare, Nitin N. Sakhare