Laboratory 2: Using dynamic memory, Arrays, Vectors

CSC205A Data structures and Algorithms Laboratory B. Tech. 2017

Vaishali R Kulkarni

Department of Computer Science and Engineering
Faculty of Engineering and Technology
M. S. Ramaiah University of Applied Sciences
Email: vaishali.cs.et@msruas.ac.in
Tel: +91-80-4906-5555 (2212) WWW: www.msruas.ac.in

Introduction and Purpose of Experiment

- Dynamic memory allocation is an important concept in C for allocation and release of the allocated memory dynamically which would help in many applications.
- This experiment includes creating and applying a vector ADT which uses the dynamic memory allocation concept.
- Use the created vector ADT for string pattern matching.

Aim and objectives

Aim:

- To develop vector ADT and to use vector ADT for string pattern matching also demonstrate the concept of dynamic memory allocation

Objectives:

At the end of this lab, the student will be able to

- Illustrate the concept of dynamic memory allocation
- Create vector ADT
- Apply vector ADT for string pattern matching application

Theory Behind the Experiment(1)

Dynamic Arrays

- A dynamic array or growable array is a random access, variable-size list data structure that allows elements to be added or removed.
- In traditional arrays, once an array is declared, the required memory is allocated statically no way that one can alter that afterwards. As a result, the size of an array can not be changed once it is declared.
- There are occasions, however, when we want to declare an array whose size can not be pre-determined. A temporary buffer for variable rate incoming data stream, a list that has variable number of elements are few examples of problems that need array size that needs to be changed dynamically.
- Using a very large array with the assumption that it can hold the largest data (yet whose size is unknown) is neither safe nor efficient. So, the question is how can we declare an array whose size is dynamically alterable?

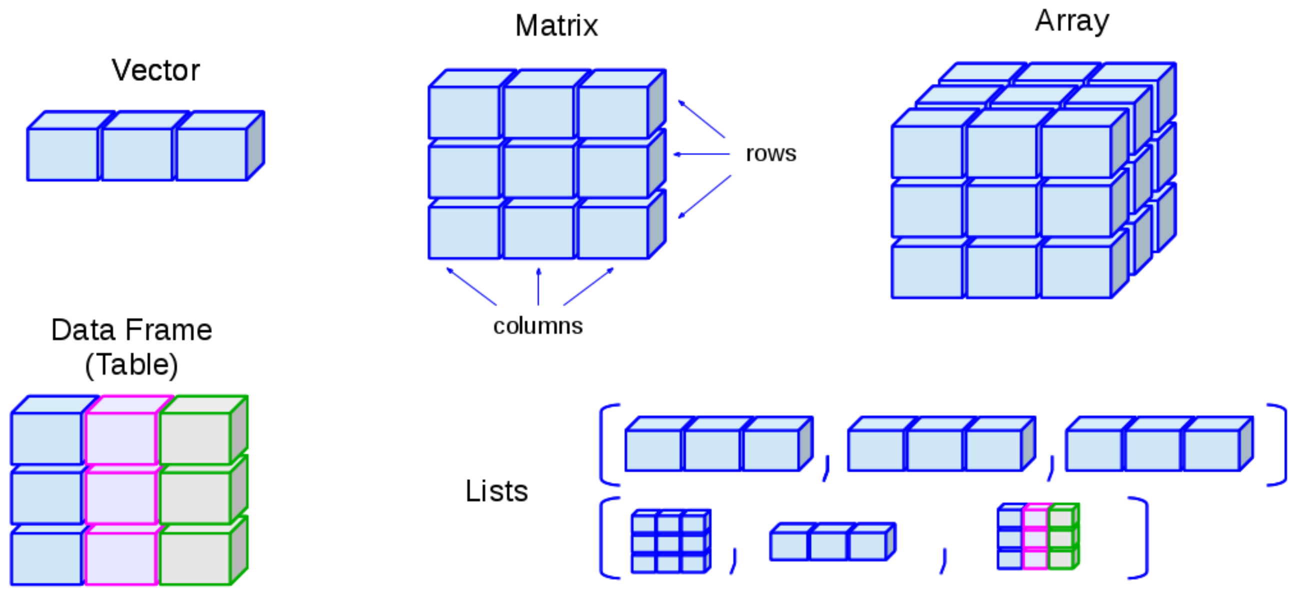
Theory Behind the Experiment(2)

Dynamic Arrays or Vectors

- A dynamic array or growable array is a random access, variable-size list data structure that allows elements to be added or removed.
- In traditional arrays, once an array is declared, the required memory is allocated statically no way that one can alter that afterwards. As a result, the size of an array can not be changed once it is declared.
- There are occasions, however, when we want to declare an array whose size can not be pre-determined. A temporary buffer for variable rate incoming data stream, a list that has variable number of elements are few examples of problems that need array size that needs to be changed dynamically.
- Using a very large array with the assumption that it can hold the largest data (yet whose size is unknown) is neither safe nor efficient. So, the question is how can we declare an array whose size is dynamically alterable?

Theory Behind the Experiment(3)

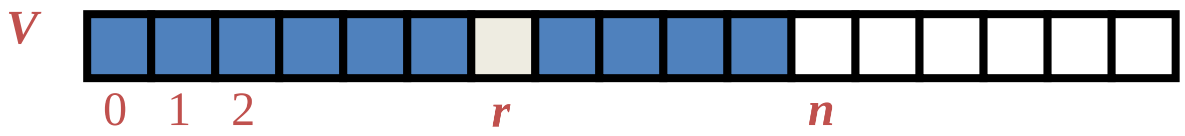
Vectors



Theory Behind the Experiment(4)

Array based implementation

- Use an array V of size N
- A variable n keeps track of the size of the vector (number of elements stored)
- Operation elemAtRank(r) is implemented in O(1) time by returning V[r]



Theory Behind the Experiment(5)

A Simple Array-Based Implementation

Let N be the size of array A and n be the number of elements in the vector.

Algorithms:

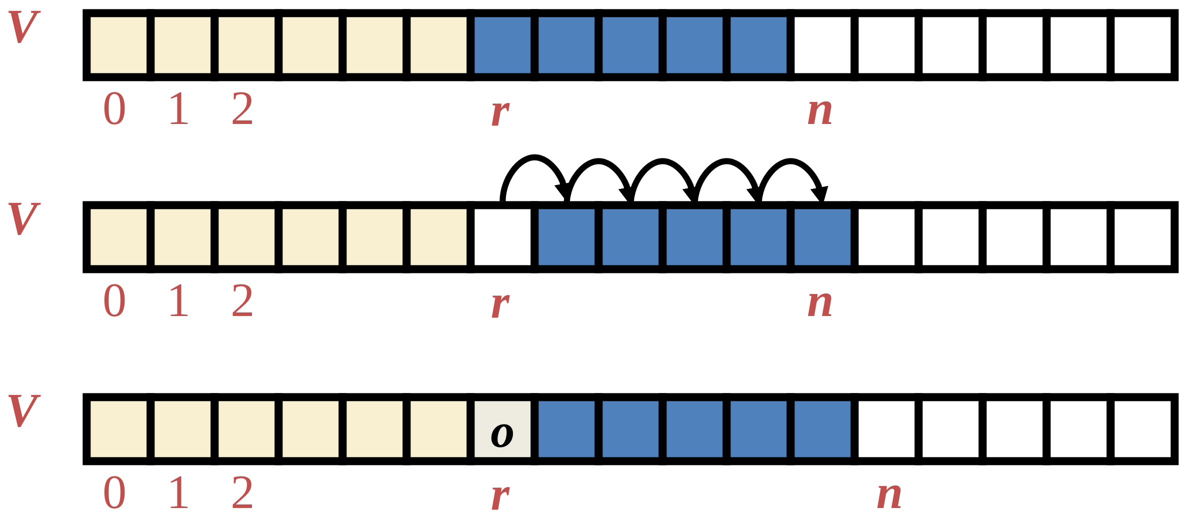
insertAtRank(r,e):

```
for $i = n - 1$ to $r$ do
    assign element $A[\ i\ ]$ to $A[\ i + 1\ ]$
assign $e$ to $A[\ r\ ]$
increment the size $n$
```

Theory Behind the Experiment(6)

Insertion Operation

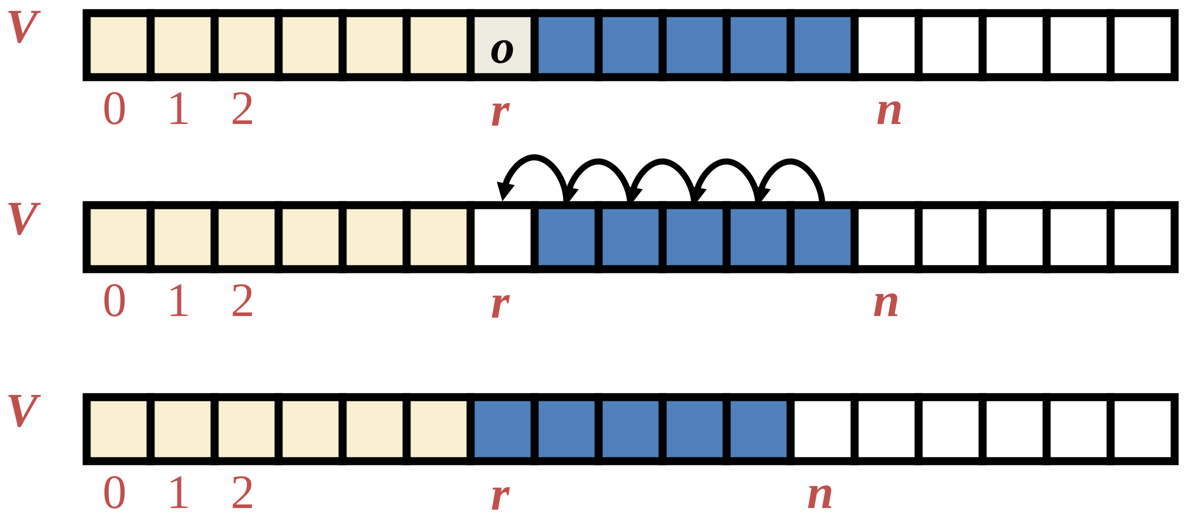
- In operation insertAtRank(r, o), we need to make room for the new element by shifting forward the n - r elements V[r], ..., V[n - 1]



Theory Behind the Experiment(7)

Delete operations

- In operation removeAtRank(r), we need to fill the hole left by the removed element by shifting backward the n - r - 1 elements V[r + 1], …, V[n - 1]
- In the worst case (r = 0), this takes O(n) time

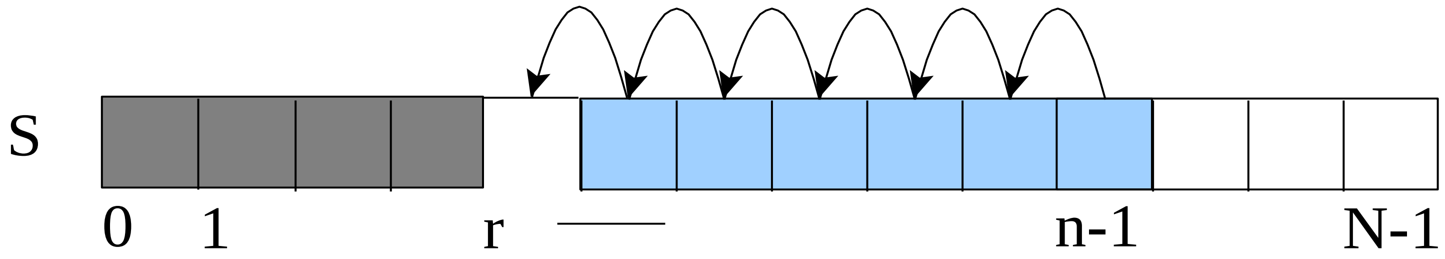


Theory Behind the Experiment(8)

```
removeAtRank(r):

save element $A[\,r\,]$ to a variable $e$
for $i = r$ to $n - 1$ do
      assign $A[\,i\,]$ to $A[\,i - 1\,]$
decrement the size $n$
return e
```

The loop is to remove the element:



Theory Behind the Experiment(9)

Vector ADT

A typical Vector ADT has the following structure:

```
 typedef struct {
  int size;      // slots used so far
  int capacity;  // total available slots
  int *data;     // array of integers we're storing
} Vector;
```

A typical Vector ADT has the following functions:

```
void vector_init(Vector *vector); // to initialize the vector
void vector_append(Vector *vector, int value); // to insert the elements at the end
int vector_get(Vector *vector, int index); // to read element at position
void vector_set(Vector *vector, int index, int value); //to modify the value at position
void vector_double_capacity_if_full(Vector *vector);// to increase the vector size
void vector_free(Vector *vector); //to free the memory allocated to vector
```

Experimental Procedure

- Analyse the problem statement
- Design an algorithm for the given problem statement and develop a flowchart/pseudo-code
- Implement the algorithm in C language
- Compile the C program
- Design test cases and test the implemented program
- Document the Results
- Analyse and discuss the outcomes of your experiment

Exercise

Design the vector ADT and apply vector ADT for string matching problem. Tabulate the output for various inputs and verify against expected values. Analyze the efficiency of the algorithm designed. Describe your learning along with the limitations of overall approach if any. Suggest how these can be overcome.

Key factors and discussion

- Vector should grow when the size is full
  - Program should double the size when the overflow is encountered
  - Before insert the one has to verify for overflow condition if it is true vector contents have to be copied to a new vector of double size
  - The above process an example of dynamically growing array
- Application of vector ADT
  - Vector ADT can be applied for a string matching problem.
  - A set of strings are read from the user and stored in a vector. Then query can be make whether a particular string is existing in vector or not.
  - This is a simple application vector ADT

Results and Presentations

- Calculations/Computations/Algorithms

  The calculations/computations/algorithms involved in each program has to be presented

- Presentation of Results

  The results for all the valid and invalid cases have to be presented

- Analysis and Discussions

  how the data is manipulated or transformed, what are the key operations involved. Errors encounters and how they are resolved.

- Conclusions

  Summary

Comments

- Limitations of Experiments

  Outline the loopholes in the program, data structures or solution approach.

- Limitations of Results

  Present the test cases; justify if the program is tested correctly considering all the outcomes. Mention what is not tested, if any.

- Learning happened

  What is the overall learning happened

- Conclusions

  Summary

References

- Gilberg, R. F., and Forouzan, B. A. (2007): A Pseudocode Approach With C, 2nd edn. Cengage Learning
- The algorithm for recursive level order traversal is taken from:

  http://www.geeksforgeeks.org/level-order-tree-traversal/