PROGRAMMING LANGUAGE OVERVIEW

Advanced Tech. P – SVMC

Contents

Programming Language

- A programming language is a formal constructed language designed to **communicate** instructions to a machine, particularly a computer.
- Programming languages can be used to create programs to control the behavior of a machine or to express **algorithms.**
- The description of a programming language is usually split into the two components of **syntax** (form) and **semantics** (meaning).
- Programming languages usually contain abstractions for defining and manipulating **data structures** or controlling the **flow** of execution.

Programming Language

Simple programs in C, C++ and Java to print out screen “Hello World”

C

```c
#include <stdio.h>
int main(){
    printf("Hello World\n");
    return 0;
}
```

C++

```cpp
#include <iostream>
using namespace std;
int main(){
    cout << "Hello World" << endl;
}
```

Java

```java
class Hello_World{
    public static void main(String args[]) throws Exception{
        System.out.println("Hello World");
    }
}
```

Contents

Variables & Data Types

- *Variable* is a storage to store a value.
- Each variable needs a name (**identifier**) that identifies it and distinguishes it from the others.
- A identifier is *valid* if:
  - Contains one or more letters, digits, or underline characters (_).
  - Always begin with a letter or _.
  - Can not be *keywords* of the programming language.
  - Can not same name with the others.

  Eg: a, Xy, _temp are valid, and

  -z, x y, 123, int are invalid.

Variables & Data Types

□ Fundamental data types

No	Group	C/C++	Java
1	Character	char	char
2	Integer	int long int long long unsigned int unsigned long unsigned long long	int long
3	Floating point	float double long double	float double
4	Boolean	bool	bool

Variables & Data Types

□ Declaration of variables

<data type> <indentifier>;

```c
#include <stdio.h>
int main(){
    int a = 1, b = 2, sum;
    sum = a + b;
    printf("a + b = %d", sum);
}
```

```cpp
#include <iostream>
using namespace std;
int main(){
    int a = 1, b = 2, sum = a + b;
    cout << "a + b = " << sum;
}
```

```java
class Hello_World{
    public static void main(String args[]) throws Exception{
        int a = 1, b = 2, sum;
        sum = a + b;
        System.out.println("a + b = " + sum);
    }
}
```

Contents

Operators & Expression

□ Operator features in C/C++/Java

No	Group	Operator name	Syntax
1	Assignment	Assignment	=
2	Arithmetic	Addition, subtraction, multiplication, division Increment, decrement Modulo	+, -, *, / ++, -- %
3	Relational	Greater than, greater than or equal to Less than, less than or equal to Equal to, not equal to	>, >= <, <= ==, !=
4	**Logical**	Logical AND, OR, negation (NOT)	&&, ||, !
5	**Bitwise**	Bitwise AND, OR, XOR, NOT Bitwise left shift, right shift	&, |, ^, ~ <<, >>

Expression = Identifier + Operator

Operators & Expression

- **Locgical**

A	B	A && B	A | | B	!A
T	T	?	?	?
T	F	?	?	
F	T	?	?	?
F	F	?	?	

Operators & Expression

- **Locgical**

What is the expression for the following condition?

$0 < x < 100$

$x < 0$ and $x > 10$

x not equal *5*

Operators & Expression

- **Bitwise**

p	q	p & q	p | q	p ^ q	~p
1	1	1	1	0	0
1	0	0	1	1	
0	1	0	1	1	1
0	0	0	0	0	

If p = 6, and q = 2, we have p = 110, q = 010, then

110	110	110	~ 0110
& 010	| 010	^ 010	
= 010	= 110	= 100	= 1001
p & q = 2	**p | q = 6**	**p ^ q = 4**	**~p = -7** Why?

Note: ~p = -p - 1

Operators & Expression

- **Bitwise**

Left shift (<<)

3 << 2 = ?

3 = 11 → 2 << 2 = 11**00** = 12

→ $a << k = a*2^k$

Right shift (>>)

21 >> 3 = ?

21 = 10101 → 21 << 2 = 101 = 5

→ $a >> k = a/2^k$

Operators & Expression

What is the value of a, b and c after run the below code?

```
int a, b, c;
a = 5;
b = 3;
c = a & b;
a = a ^ c;
b = b | a;
c = c << a;
```

Contents

Input & Output

□ Standard streams

Below programs read 2 value from keyboard and print out the sum of them.

C

```c
#include <stdio.h>
int main(){
    int a, b;
    scanf("%d %d", &a, &b);
    printf("a + b = %d", a+b);
}
```

C++

```cpp
#include <iostream>
int main(){
    int a, b;
    std::cin >> a >> b;
    std::cout << "a + b = " <<a+b;
}
```

Java

```java
public static void main(String args[]) throws Exception{
    int a, b;
    //System.setIn(new FileInputStream("input.txt"));
    Scanner sc = new Scanner(System.in);
    a = sc.nextInt(); b = sc.nextInt();
    System.out.println("a + b = " + (a + b));
}
```

Contents

Control Flow

- **Simple control flow**

  Choice: `if`, `switch`

  Loops: `for`, `while`, `do while`

```
//even numbers in [1, 10]
for (int i = 1; i <= 10; i++)
    if (i % 2 == 0)
        cout << i << " ";
```

```
//odd numbers in [1, 10]
for (int i = 1; i <= 10; i++)
    if (i % 2)
        cout << i << " ";
```

```
//compute a^b
int a = 2, b = 5, pow = 1;
for (int i = 0; i < b; i++)
    pow = pow*a;
return pow;
```

```
//compute a^b
int a = 2, b = 5, pow = 1;
while (b--)
    pow = pow*a;
return pow;
```

Excersise

- Write programs as follow:

```
int gcd(int a, int b); //return greatest common divisor of a and b
int lcm(int a, int b); //return the least common multiple of a and b
```

Contents

Practice 1

Divisibility

Print all integer numbers a ($1 < a < N$) such that a is divisible by x and not divisible by y.

For example, if $N = 7$, $x = 2$ and $y = 4$, the the answer should be 2 and 6, since 2 and 6 are divisible by 2 and not divisible by 4.

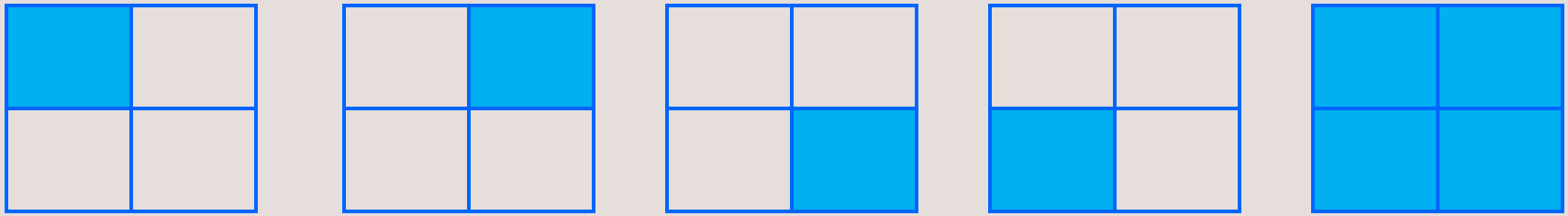
Source: Divisibility (15708) http://www.spoj.com/problems/SMPDIV/

Practice 2

Feynman

Given a integer number *N*, print out the total number of different squares in a grid of *N*x*N* squares.

For example, if *N* = 2, the answer should be 5.



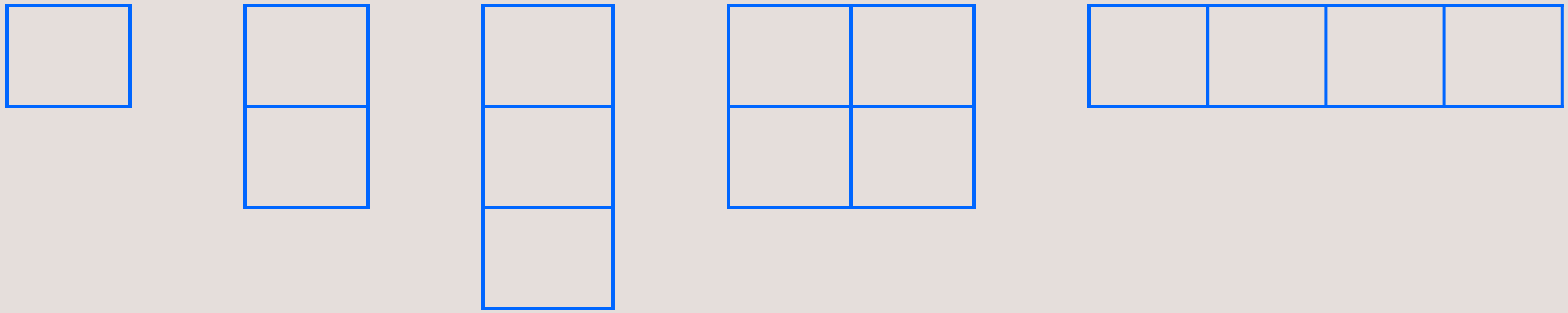
Source: http://www.spoj.com/problems/SAMER08F/

Homework 1

Rectangles

Given N squares of size 1, how many different rectangles can we form using these squares?

For example, if $N = 4$, the answer should be 5.



Source: http://www.spoj.com/problems/AE00/

Homework 2

Prime Generator

A prime is a number such that it is divisible by only 1 and itself (1 is not prime number).

Print out all prime numbers between M and N;

For example, if $M = 1$ and $N = 10$, the answer should be

2, 3, 5, 7.

Source: http://www.spoj.com/problems/PRIME1/

Homework 3

SUM OF PRODUCT

Given a number *N* ($1 < N < 10^9$), find the sum of all products *x***y*, where *x* from 1 to *N*, and *y* is the integer part of *N*/x.

For example, if *N* = 4, then *x* are 1, 2, 3, 4 and *y* are 4, 2, 1, 1 responding. Hence the sum is 1*4 + 2*2 + 3*1 + 4*1 = 15.

Source: http://www.spoj.com/problems/SUMPRO/

Practice

1. **Divisibility (15708)** http://www.spoj.com/problems/SMPDIV/
2. **Feynman (3410)** http://www.spoj.com/problems/SAMER08F/

Homework

1. **Rectangles (4300)** http://www.spoj.com/problems/AE00/
2. **Prime Generator (2)** http://www.spoj.com/problems/PRIME1/
3. **SUM OF PRODUCT (22455)** http://www.spoj.com/problems/SUMPRO/

Reference

- [wiki] Array data structure https://en.wikipedia.org/wiki/Programming_language
- [wiki] Control flow https://en.wikipedia.org/wiki/Control_flow
- [wiki] Variable https://en.wikipedia.org/wiki/Variable_(computer_science)
- [wiki] Operators in C and C++ https://en.wikipedia.org/wiki/Operators_in_C_and_C%2B%2B

ARRAY

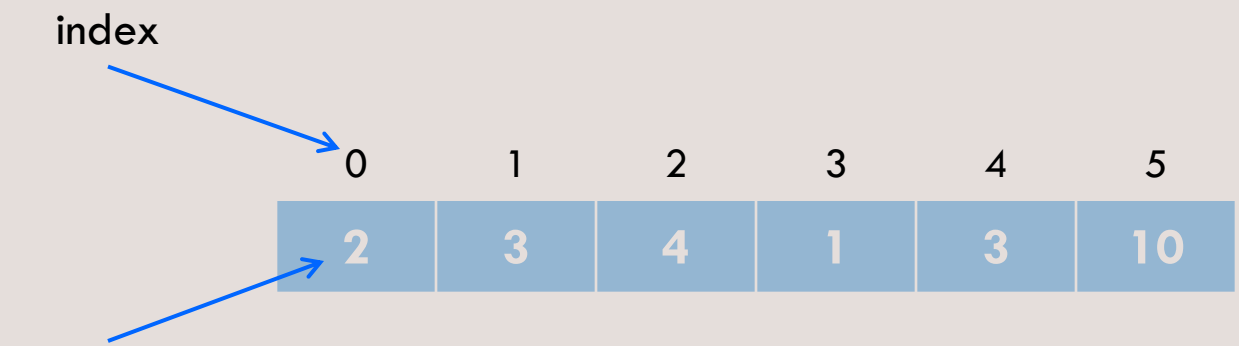
Advanced Tech. P – SVMC

Contents

- Array manipulation
- Searching & sorting algorithms
- Sorting library
- Through 2D Array
- Exhaustive search & Greedy
- Practice

Array

- Array is a data structure consisting of a collection of elements (values or variables).
- Each identified by at least one array index or key.
- An array is stored so that the position of each element can be computed from its index tuple by a mathematical formula.
- The simplest type of data structure is a linear array (1D array).



1D array of integers with 6 elements

Contents

1D Array Declaration

- **C/C++**
  - `type name[size];`
  - `type name[size] = {elem 1, elem 2, …, elem N};`

  Ex:
  - `int A[5]; //an array with maximum 5 values of type int`
  - `int A[5] = {1, 2, 3, 4};`
- ***Java***
  - `type[] name;  //declare`
    `name = new type[size]; //create`
  - `type[] name = {elem 1, elem 2, …, elem N};`

  Ex:
  - `int[] A;`
    `A = new int[5];`
  - `int[] B = new int[10];`

2D Array Declaration

- **C/C++**
  - `type name[row][col];`
  - `type name[row][col] = {{elem 1, elem 2, …, elem N},`
    `…`
    `{elem 1, elem 2, …, elem N}};`

Ex:

- `int A[5][10];`
- `int A[2][3] = {{1, 2, 3},`
  `{4, 5, 6}};`

- ***Java***
  - `type[][] name = new int[row][col];`

Accessing Array Elements

- **1D Array**

```
int A[5] = {1, 2, 3, 4, 5};
A[1] = A[2];
x = A[1];
```

x = ?

- **2D Array**

```
int A[2][3] = {{1, 2, 3},
               {4, 5, 6}};
A[0][1] = A[1][0];
x = A[0][1];
```

x = ?

Contents

Searching in array

- **Is x belongs to array A?**

```
for (int i  = 0; i < N; i++)
     if (x == A[i])
            printf("YES");
```

- **What is the maximum value of array A?**

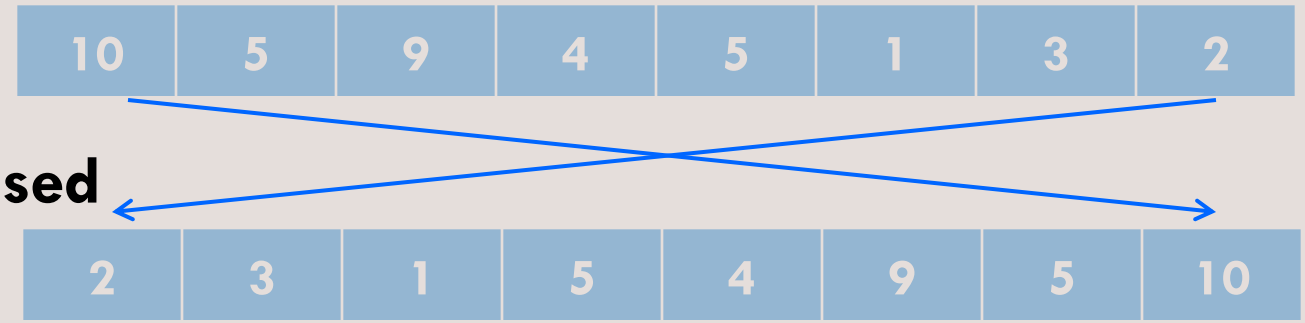
```
int max = A[0];
for (int i  = 1; i < N; i++)
     if (max < A[i])
            max = A[i];
```

- **What is the minimum value of array A?**

```
int min = A[0];
for (int i  = 1; i < N; i++)
     if (min > A[i])
            min = A[i];
```

Reverse an array

- **Original**



- **Reversed**

- **Algorithm?**

```
int tmp;
for (int i = 0; i < N/2; i++){
     tmp = A[i];
     A[i] = A[N-1-i];
     A[N-1-i] = tmp;
}
```

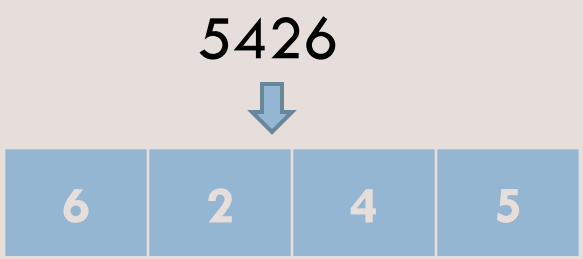
How to reverse a number?

Reverse of **5426** is **6245.**

Algorithm?

1. **Store digits of the number into an array in reverse order**

```
int len = 0;
while (N > 0){
      A[len++] = N%10;
      N = N/10;
}
```



2. **Converse the array of digits to number**

```
N = 0;
for (int i = 0; i < len; i++)
      N = N*10 + A[i];
```

Copy array

How to copy an array to an other?

Is that true? arr1 = arr2

Algorithm

```
for (int i = 0; i < N; i++){
     arr1[i] = arr2[i];
}
```

Sorting in array

- **Unsorted**

10	5	9	4	5	1	3	2

- **Sorted**

Non-decreasing order:

1	2	3	4	5	5	9	10

Non-increasing order:

10	9	5	5	4	3	2	1

How to sort?

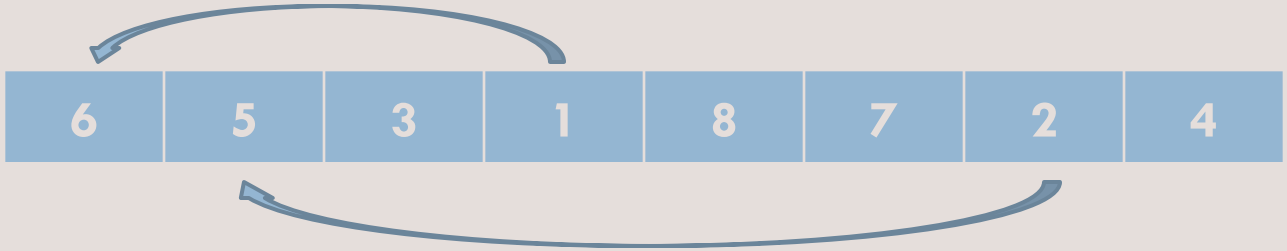
□ Selection sort

1. Find the smallest element in the unsorted part.
2. Swap it with the lefmost unsorted element.
3. Move the boundary of unsort part to the right.
4. Repeat until unsorted part is empty.

O(N^2)

```
int tmp;
for (int i = 0; i < N-1; i++){
    for (int j = i+1; j < N; j++)
        if (A[j] < A[i]){
            //swap(A[j], A[i]);
            tmp = A[i];
            A[i] = A[j];
            A[j] = tmp;
        }
  }
```

What wrong?



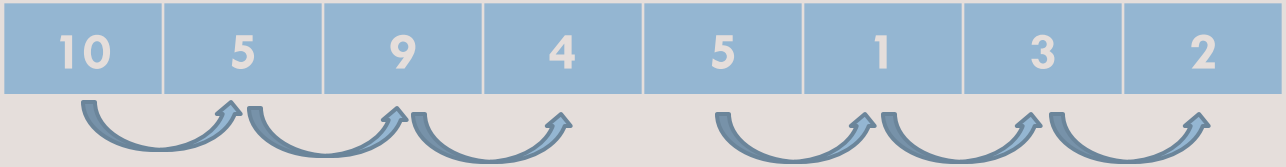
How to sort?

□ Bubble sort

1. Compares each pair of adjacent items
2. Swaps them if they are in wrong oder
3. Repeat until no swaps are needed

$O(N^2)$

```
bool flag = true;
int tmp;
while (flag){
      flag = false;
      for (int i = 1; i < N; i++)
          if (A[i-1] > A[i]){
            swap(A[i-1], A[i]);
            flag = true;
          }
}
```



How to sort?

- **Others algorithm**

Name	Average	Worst	Method
Quick sort	*n*log*n*	N^2	Partitioning
Merge sort	*n*log*n*	*n*log*n*	Merging
Heap sort	*n*log*n*	nlogn	Selection
Counting sort	$n + r$	$n+r$	Counting

Contents

Sorting library

- **Sorting library**
  - **C/C++** (<stdlib.h>, <algorithm>)

```
int compare (const void *a,  const void *b){
   return (*(int*)a - *(int*)b);
}
```

- `qsort (A, N, sizeof(int), compare);`
- `sort(A, A + N);`

  - ***Java***

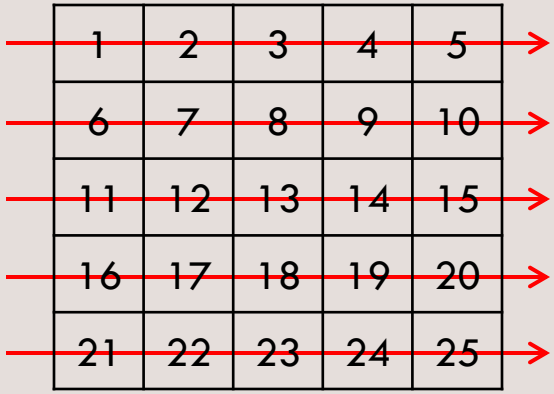
```
Arrays.sort(A);
```

Contents

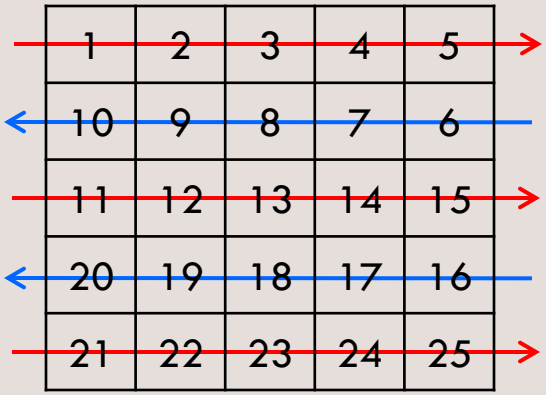
Through 2D array

- **Some ways**

```
int A[5][5];
for (int i = 0; i < 5; i++)
    for (int j = 0; j < 5; j++)
        A[i][j] = 5*i + j + 1;
```



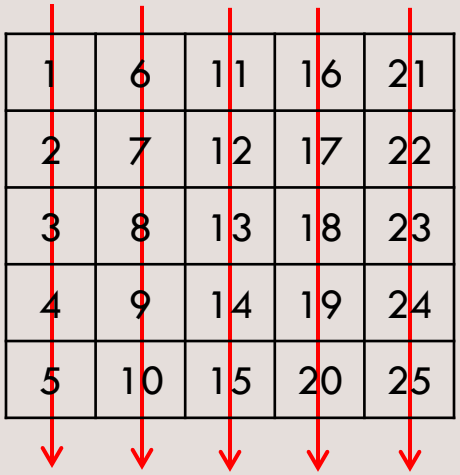

```
int A[5][5];
for (int i = 0; i < 5; i++)
    for (int j = 0; j < 5; j++)
        A[i][j] = 5*i + (5-j)*(i%2) +
                  (j+1)*((i+1)%2);
```



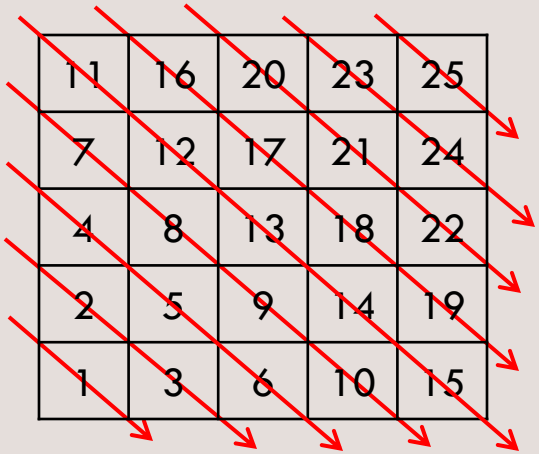
Through 2D array

- **Some ways (cont.)**

```
int A[5][5];
for (int j = 0; j < 5; j++)
    for (int i = 0; i < 5; i++)
        A[i][j] = 5*j + i + 1;
```



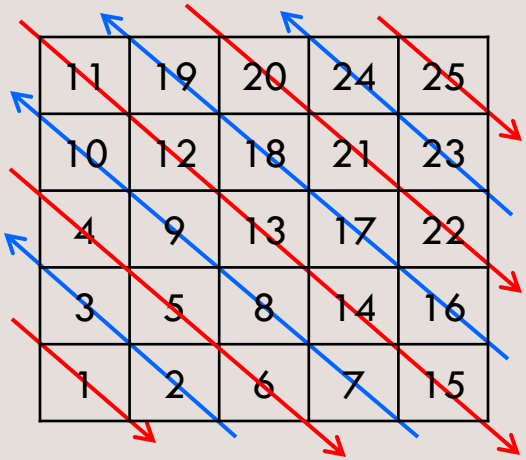

```
int A[5][5], cnt = 1;
for (int i = 4; i >= 0; i--)
    for (int j = 0; j < 5-i; j++)
        A[i+j][j] = cnt++;
for (int j = 1; j < 5; j++)
    for (int i = 0; i < 5-j; i++)
        A[i][i+j] = cnt++;
```

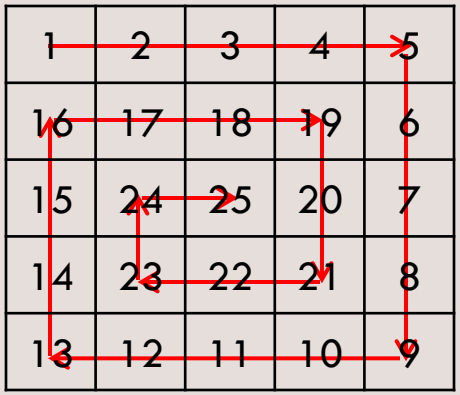


Through 2D array

- **Exercise**

How ?





Contents

Exhaustive search vs. Greedy

- **Example 1**

  Given a sorted array, check is a number *k* in this array or not?

  Eg: Check is 60 in the below array or not?

3	4	7	10	15	29	60	100

  - **Exhaustive approach**

    Search from left to right

3	4	7	10	15	29	60	100

Found

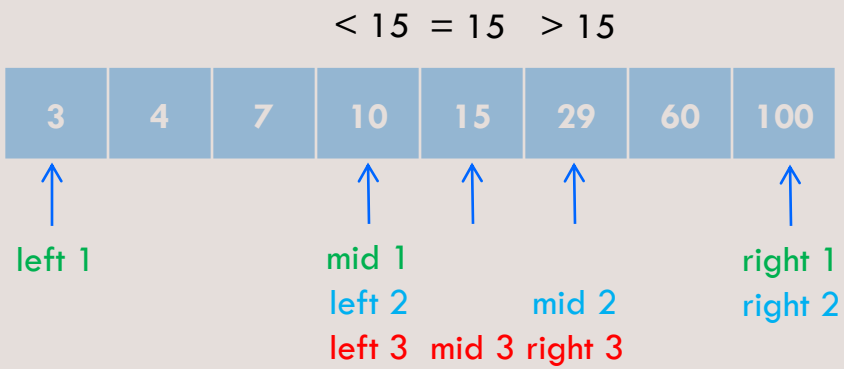
O(*N*)

Exhaustive search vs. Greedy

- **Example 1 (cont.)**
  - **Greedy approach (Binary search)**

```
bool bSearch (int left, int right){
      if (left > right)
         return 0;
      int mid = (left + right)/2;
      if (arr[mid] == key)
         return 1;
      else if (arr[mid] > key)
         bSearch (left, mid-1)
      else
         bSearch (mid+1, right)
      return 0;
}
```

Is **15** in the below array?



O(logN)

Exhaustive search vs. Greedy

- **Example 2**

  How to compute square root of a number with 6 digits after the decimal point without using STL?

  Eg: sqrt(2) = 1.414214

  - **Exhaustive approach**

    Try all real number have 6 digits after the decimal from 1.000000 to 2.000000, if a number x such that $|x^2 - 2| < 10^{-6}$, then it is square root of 2.

  - **Greedy approach**

    Binary search!!!

Exhaustive search vs. Greedy

- **Example 3**

  Given a set of integer numbers, we need to compute the sum of them, the cost to plus 2 number a and b is 0.5*(a+b). How to minimize this cost?

  Eg: We need to find the sum of 4 numbers 6, 10, 2, 4.

  Assume we will compute as following: (6 + 10 + 2 + 4)

  6 + 10 = 16 with cost 0.5*(6+10) = 8;

  16 + 2 = 18 with cost 9;

  18 + 4 = 22 with cost 11 ;

  So the sum is 21 with total cost is 8 + 9 + 11 = 28.

  However we can compute as following to get minimum cost:

  2 + 4 + 6 + 10

  The minimum cost is: 3 + 6 + 11 = 20.

Exhaustive search vs. Greedy

- **Example 3 (cont.)**
  - **Exhaustive approach** (Check all candidates)

    2 + 4 + 6 + 10 with cost 3 + 6 + 11 = **20**;
    2 + 4 + 10 + 6 with cost 4 + 8 + 11 = 23;
    2 + 6 + 4 + 10 with cost 4 + 6 + 11 = 21;
    2 + 6 + 10 + 4 with cost 4 + 9 + 11 = 24;
    2 + 10 + 4 + 6 with cost 6 + 8 + 11 = 25;
    2 + 10 + 6 + 4 with cost 6 + 9 + 11 = 26;
    4 + 6 + 2 + 10 with cost 5 + 6 + 11 = 22;
    4 + 6 + 10 + 2 with cost 5 + 10 + 11 = 26;
    4 + 10 + 2 + 6 with cost 7 + 8 + 11 = 26;
    4 + 10 + 6 + 2 with cost 7 + 10 + 11 = 28;
    6 + 10 + 2 + 4 with cost 8 + 9 + 11 = 28;
    6 + 10 + 4 + 2 with cost 8 + 10 + 11 = 29.

How many candidates?

N!/2

Exhaustive search vs. Greedy

- **Example 3 (cont.)**
  - **Greedy approach**

How to choose exactly 2 number to plus at step *k*?

Assume we need to compute sum of *N* numbers, and we choose a permutation of the numbers $a_1, a_2, \ldots, a_N$ as a candidate, then the cost is:

$$0.5*(a_1+b_1) + 0.5*(a_2+b_2) + \ldots + 0.5*(a_N+b_N)$$

So, to minimimum this cost, at step *k* the sum (a_k+b_k) must be minimum.

$O(N^2)$???

Contents

Practice

1. **Adding Reversed Numbers (42)** http://www.spoj.com/problems/ADDREV/
2. **Queens, Knights and Pawns (1706)** http://www.spoj.com/problems/QKP/

Homework

1. **Queue (Rookie) (16015)** http://www.spoj.com/problems/QUE1/
2. **Number Steps (1112)** http://www.spoj.com/problems/NSTEPS/
3. **Count on Cantor (302)** http://www.spoj.com/problems/CANTON/

Reference

- [wiki] Array data structure https://en.wikipedia.org/wiki/Array_data_structure
- [wiki] Sorting algorithm https://en.wikipedia.org/wiki/Sorting_algorithm
- [wiki] Selection sort https://en.wikipedia.org/wiki/Selection_sort
- [wiki] Bubble sort https://en.wikipedia.org/wiki/Bubble_sort
- [wiki] Greedy algorithm https://en.wikipedia.org/wiki/Greedy_algorithm

STRING

Advanced Tech. P – SVMC

Contents

- String manipulation
- String library
- Palindrome & pattern matching
- Practice

String

- A character is a unit of information that roughly corresponds to a grapheme, grapheme-like unit, or symbol.

  Eg: letters (a-z, A-Z), numerical digits(0-9), common punctuation marks (“.”, “-”), and whitespace.

- A string is traditionally a sequence of characters.

- **Note:** in C, a string is end at ‘\0’.

String Declaration

- **C/C++**
  - `char name[size];`
  - `char name[size] = "initial string";`

  Ex:
  - `char S[50]; //a string with maximum 50 characters`
  - `char S[50] = "Hello World";`

    `S[4] = 0; //S = ???`

- ***Java***
  - `char[] SS = {'i','n','i','t','i','a','l',' ','s','t','r','i','n','g'};`
    `String S = new String(SS); //new keyword`
  - `String S = "initial string"; //string literal`

  Ex:
  - `char[] SS = {'H','e','l','l','o',' ','W','o','r','l','d','!'};`
    `String S = new String(SS);`
  - `String S = "Hello World!";`

Contents

String Length

- **C/C++**
- **Make function**

```
int my_strlen(char S[]){
    int len = 0;
    while (S[len] != 0) //S[len] != '\0'
          len++;
    return len;
}
```

- **Using STL** (strlen())

```
#include <stdio.h>
#include <string.h>
int main(){
    char S[] = "Hello World";
    printf("Length of string is: %d", strlen(S));
}
```

String Length

- ***Java***
- **Using STL** (length())

```
class String_Length{
    public static void main(String args[]) throws Exception{
        String S = "Hello World";
        System.out.println("Length of string is " + S.length());
    }
}
```

Accessing String Elements

- **C/C++**

```
#include <stdio.h>
int main(){
    char S[] = "Hello World";
    int i = 0;
    while (S[i] != 0){
        if (S[i] != ' ')
            printf("%c", S[i]);
        i++;
    }
}
```

Accessing String Elements

- ***Java***

```
class String_Accessing{
    public static void main(String args[]) throws Exception{
        String S = "Hello World";
        for (int i = 0; i < S.length(); i++)
            if (S.charAt(i) != ' ')
                System.out.print(S.charAt(i));
    }
}
```

Exercise 1

□ Name normalization

Given a string of name, your goal is normalize this name as following:

1. The name must contain only letters and space characters.
2. The fisrt letter of each word must be in UPPER case, the other are in lower case.
3. No space in the begin and end of name.
4. Only one space between two words.

Eg: If you given a string " *mY Na99Me i2s John1* ", you need to output "*My Name Is John*"

Contents

String Copy

- C/C++
- Make function

```
void my_strcpy(char Dest[], char Source[]){
    int i = 0;
    while (Source[i] != 0){
        Dest[i] = Source[i];
        i++;
    }
    Dest[i] = 0;
}
```

```
void my_strcpy2(char* Dest, char* Source){
    while (*Source != 0)
        *Dest++ = *Source++;
    *Dest = 0;
}
```

String Copy

- **C/C++** (cont.)
- **Using STL** (strcpy())

```
#include <stdio.h>
#include <string.h>
int main(){
    char S[] = "Hello World";
    char SS[50];
    strcpy(SS, S);
    printf("%s\n", SS);
}
```

What's the output of the above program if we declare SS as follow?

```
char SS[5];
```

String Copy

- ***Java***

```
String S = "Hello World";
String SS = S;
System.out.print(SS);
```

Very simple!!!

String Compare

- **C/C++**
- **Make function**

```
int my_strcmp(char *S1, char *S2){
    while (*S1 == *S2){
        if (*S1 == 0) break;
        *S1++, *S2++;
    }
    return *S1 - *S2;
}
```

What's the output of the above program?

- **Using STL** strcmp()
- ***Java***

Using methods equal(), compareTo(), compareToIgnoreCase().

String Reverse

- **C/C++**

```
void my_reverse(char *S){
    int len = strlen(S);
    char tmp;
    for (int i = 0; i < len/2; i++){
        tmp = S[i];
        S[i] = S[len-i-1];
        S[len-i-1] = tmp;
    }
}
```

- ***Java***

```
String S = "Hello World";
String SS = new StringBuffer(SS).reverse().toString();
System.out.print(SS);
```

String to number

- **C/C++**
- **Make function**

```
int my_atoi(char *S){
    int i = 0, val = 0;
    while (S[i] >= '0' && S[i] <= '9'){
        val = val*10 + S[i] - '0';
        i++;
    }
    return val;
}
```

What's output if we call `my_atoi("12345abc")` and `my_atoi("-123")`?

- **Using STL** atoi()
- ***Java***

Using methods Integer.*parseInt*(), Integer.*valueOf*().

Number to string

- **C/C++**
- **Make function**

```
void my_itoa(int val, char *S){
    int i = 0;
    while (val > 0){
        S[i++] = val % 10 + '0';
        val = val/10;
    }
    S[i] = 0;
    my_reverse(S);
}
```

What's output if we call `my_itoa(123, S)`, `my_itoa(-123, S)` and `my_itoa(-(-123), S)`

- **Using STL** itoa()
- ***Java***

Using method Integer.*toString()*.

Contents

Exercise 2

- **Palindrome**
- A string is said to be palindrome if we write it from left and right then we get the same result. For example “non” is a palindrome of size 3.
- A substring of S is a string begin at character i^{th} and end at j^{th} of string S. For example, “el” is a subtring of “Hello”.

Give a string, your goal is find the longest palindrome substring of it.

Eg: “aabbaa” is the answer for case “aaabbaac”.

Exercise 3

- **Pattern Matching**

Given two strings S and P, you need to count how many subtring of S is P.

Eg: String "abaabbbabba" have 3 substring is "ab".

Contents

Pratice

1. **To and Fro (400)** http://www.spoj.com/problems/TOANDFRO/
2. **Mirror Strings (12262)** http://www.spoj.com/problems/MSUBSTR/

Homework

1. **Anti-Blot System (2157)** http://www.spoj.com/problems/ABSYS/
2. **Broken Keyboard (2852)** http://www.spoj.com/problems/BROKEN/
3. **Find String Roots (7212)** http://www.spoj.com/problems/FINDSR/

Reference

- [wiki] Character (computing) https://en.wikipedia.org/wiki/Character_(computing)
- [wiki] String (computer science) https://en.wikipedia.org/wiki/String_(computer_science)

STACK & QUEUE

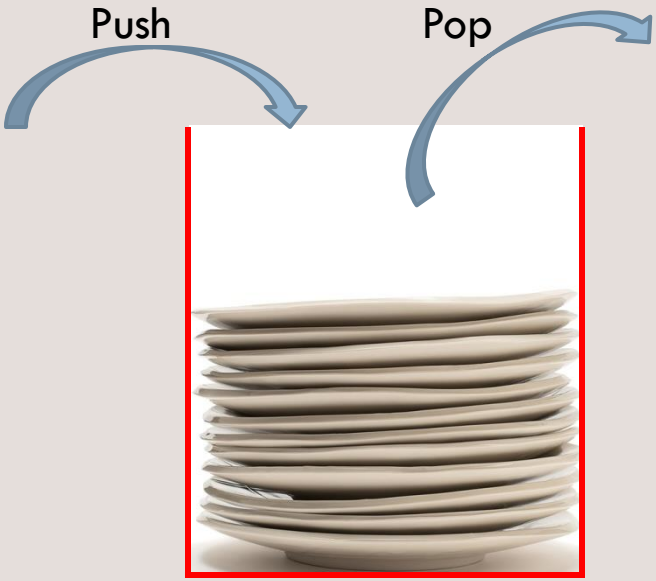
Advanced Tech. P – SVMC

Contents

Stack

- A **stack** is an abstract data type that serves as a collection of elements, with two principal operations:
  - ***push***, which adds an element to the collection, and
  - ***pop***, which removes the most recently added element that was not yet removed.

The order in which elements come off a stack **LIFO** (**last in first out**)

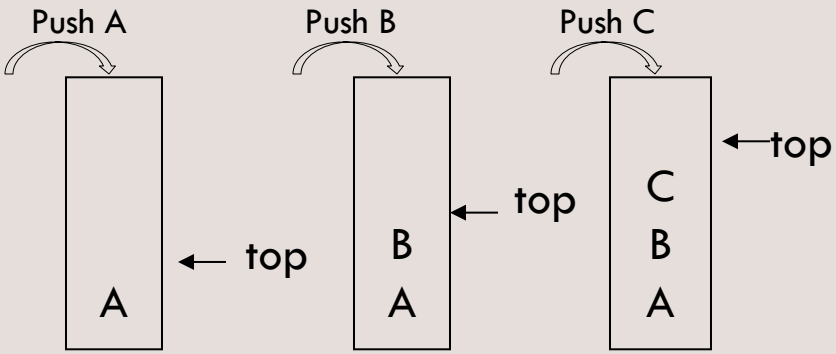


Contents

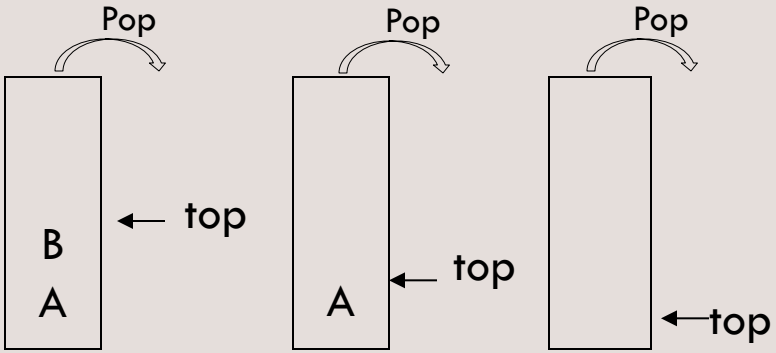
Stack Implementation

- A stack can be easily implemented through an array.
- The push and pop operations occur only at one end of the structure, referred to as the ***top*** of the stack.

```
void push(int S[size], int x){
    if (top < size)
        S[top++] = x;
}
```




```
int pop(int S[size]){
    if (top > 0)
        return S[--top];
    return 0;
}
```



Contents

Stack library

- **C/C++** (#include <stack>)

```
#include <iostream>
#include <stack>

using namespace std;

int main(){
    stack <int> S;

    for (int i = 0; i < 10; i++)
       S.push(2*i + 1);
    for (int i = 0; i < 6; i++)
       S.pop();
    cout << S.size() << " " << S.top();

    return 0;
}
```

What's ouput?

Stack library

- ***Java*** (import java.util.Stack)

```
import java.util.*;

public class Stack_Demo {

    public static void main(String[] args){
        Stack S = new Stack();

        for (int i = 0; i < 5; i++)
           S.push(2*i);
        S.push(S.pop());
        System.out.println(S.peek());
    }

}
```

What's ouput?

Contents

Stack Application

- **Real life**
  - Pile of books
  - Plate trays
- **More applications related to computer science**
  - Expression evaluation and syntax parsing
  - Backtracking
  - Runtime memory management

Stack Application

- **Syntax parsing**

  Given a combine of the folowing syntax parsing, check it is valid or not? (,), {, }, [,].

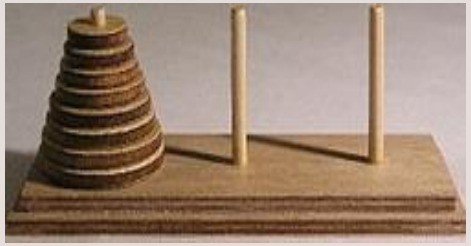
  For example, if the input is “([])” then it is valid, but in case “(([]]”, it is invalid.

  How to check?

Stack Application

- **Tower of Hanoi**

The Tower of Hanoi is a mathematical game or puzzle.



It consists of three rods, and a number of disks of different sizes

The objective of the puzzle is to move the entire stack to another rod with the following rules:

1. Only one disk can be moved at a time.
2. Each move, a disk can only be moved if it is the uppermost disk on a stack.
3. No disk may be placed on top of a smaller disk.

How to move?

Stack Application

- **Tower of Hanoi**

```
void tower(int N, stack A, stack B, stack C){
    if (N > 0){
        tower(N-1, A, C, B);
        int d = A.pop();
        C.push(d);
        tower(N-1, B, A, C);
}
```

Contents

Queue

- A **queue** is an abstract data type that serves as a collection of elements, with two principal operations:
  - ***enqueue***, which adds an element to the *last* of collection, and
  - ***dequeue***, which removes the *first* recently added element that was not yet removed.

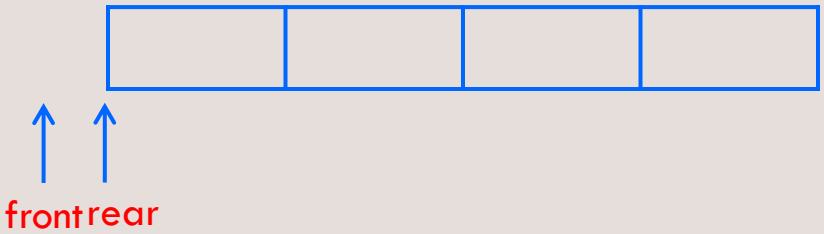
  The order in which elements come off a queue **FIFO** (**first in first out**)

Queue Implementation

- A queue can be easily implemented through an array.
- The enqueue operations occur at the end of the structure (***rear***), while dequeue operations occur at the first (***front***).

```
void enqueue(int Q[size], int x){
    if (rear < size-1)
        Q[++rear] = x;
}
```

```
int dequeue(int Q[size]){
    if (front != rear)
        return Q[++front];
    return 0;
}
```



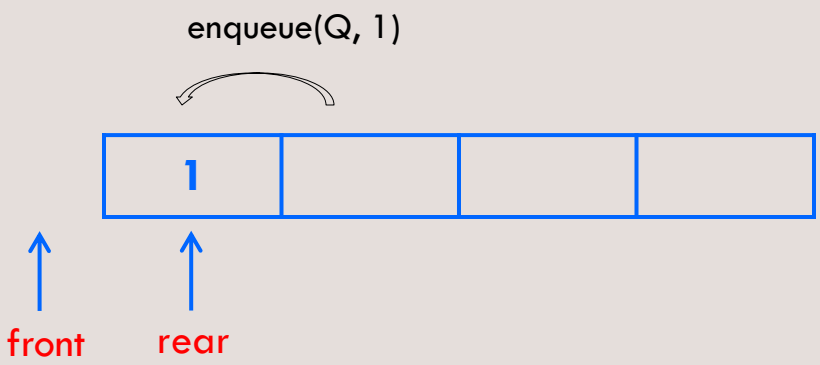
A queue is **empty** if *front* = *rear*.
A queue is **full** if *rear* = size-1;

Queue Implementation

- A queue can be easily implemented through an array.
- The enqueue operations occur at the end of the structure (***rear***), while dequeue operations occur at the first (***front***).

```
void enqueue(int Q[size], int x){
    if (rear < size-1)
        Q[++rear] = x;
}
```

```
int dequeue(int Q[size]){
    if (front != rear)
        return Q[++front];
    return 0;
}
```

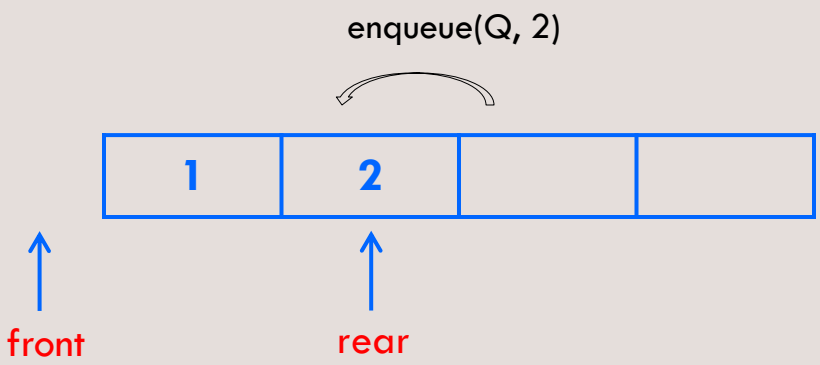


Queue Implementation

- A queue can be easily implemented through an array.
- The enqueue operations occur at the end of the structure (***rear***), while dequeue operations occur at the first (***front***).

```
void enqueue(int Q[size], int x){
    if (rear < size-1)
        Q[++rear] = x;
}
```

```
int dequeue(int Q[size]){
    if (front != rear)
        return Q[++front];
    return 0;
}
```

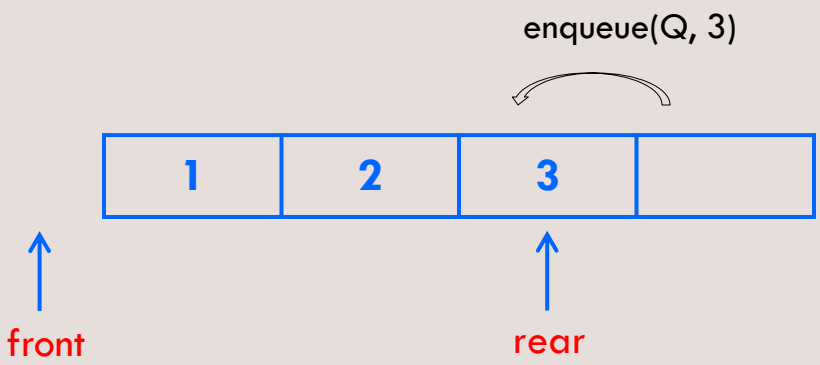


Queue Implementation

- A queue can be easily implemented through an array.
- The enqueue operations occur at the end of the structure (***rear***), while dequeue operations occur at the first (***front***).

```
void enqueue(int Q[size], int x){
    if (rear < size-1)
        Q[++rear] = x;
}
```

```
int dequeue(int Q[size]){
    if (front != rear)
        return Q[++front];
    return 0;
}
```

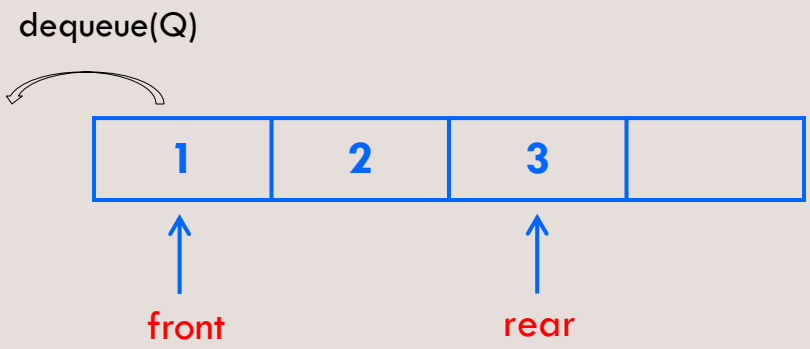


Queue Implementation

- A queue can be easily implemented through an array.
- The enqueue operations occur at the end of the structure (***rear***), while dequeue operations occur at the first (***front***).

```
void enqueue(int Q[size], int x){
    if (rear < size-1)
        Q[++rear] = x;
}
```

```
int dequeue(int Q[size]){
    if (front != rear)
        return Q[++front];
    return 0;
}
```

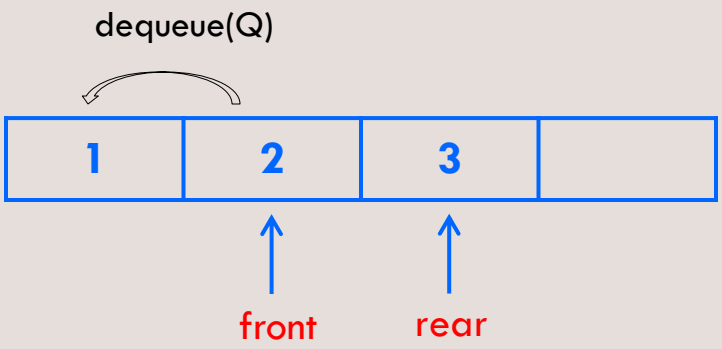


Queue Implementation

- A queue can be easily implemented through an array.
- The enqueue operations occur at the end of the structure (***rear***), while dequeue operations occur at the first (***front***).

```
void enqueue(int Q[size], int x){
    if (rear < size-1)
        Q[++rear] = x;
}
```

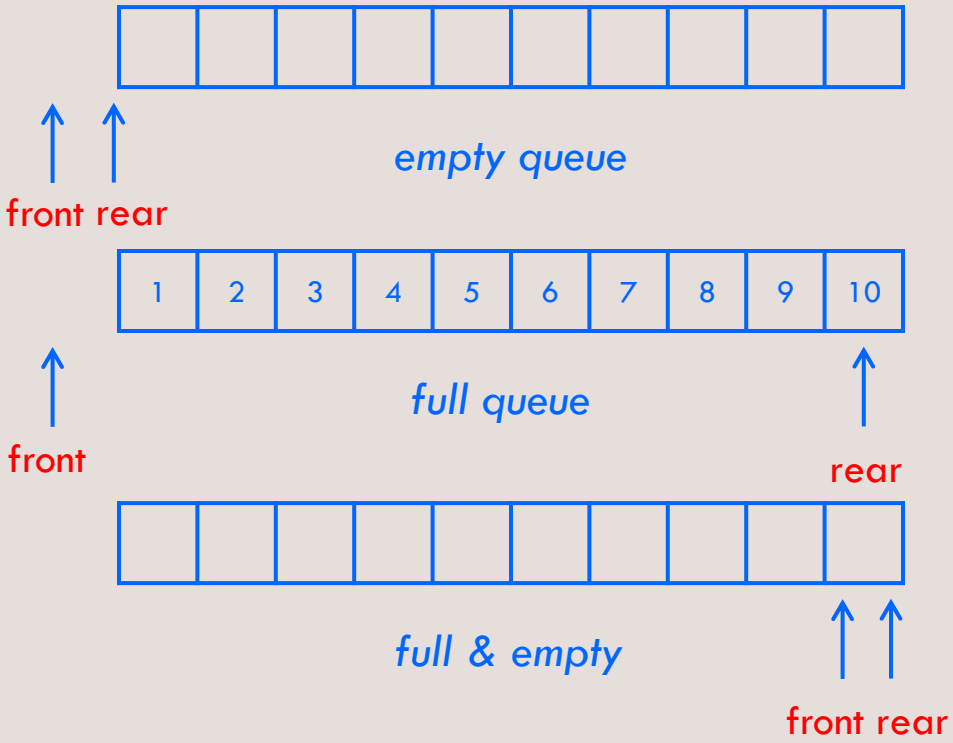
```
int dequeue(int Q[size]){
    if (front != rear)
        return Q[++front];
    return 0;
}
```



Queue Implementation

- What happen after run the following program?

```
int Q[10];
front = rear = -1;
for (int i = 1; i < 11; i++)
    enqueue(Q, i);
for (int i = 0; i < 10; i++)
    dequeue(Q);
```



Queue is empty but we can not *enqueue* since it is also full.

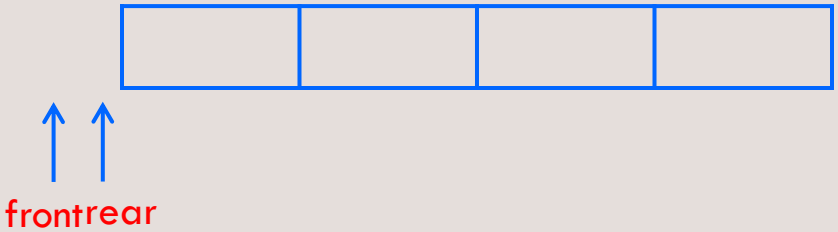
How to solve problem?

Queue Implementation

- **Circular Queue**

```
void enqueue(int Q[size], int x){
    rear = (rear + 1)%size;
    if (rear != front)
        Q[rear] = x;
}
```

```
int dequeue(int Q[size]){
    if (front != rear){
        front = (front + 1)%size;
        return Q[front];
    }
    return 0;
}
```



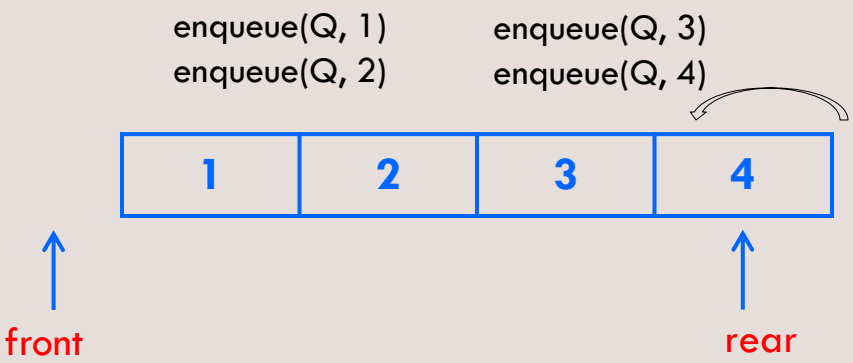
- **When is it empty or full?**

Queue Implementation

- **Circular Queue**

```
void enqueue(int Q[size], int x){
    rear = (rear + 1)%size;
    if (rear != front)
        Q[rear] = x;
}
```

```
int dequeue(int Q[size]){
    if (front != rear){
        front = (front + 1)%size;
        return Q[front];
    }
    return 0;
}
```



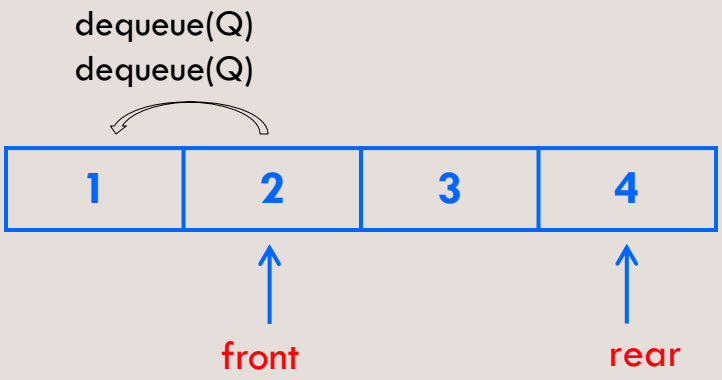
- **When is it empty or full?**

Queue Implementation

- **Circular Queue**

```
void enqueue(int Q[size], int x){
    rear = (rear + 1)%size;
    if (rear != front)
        Q[rear] = x;
}
```

```
int dequeue(int Q[size]){
    if (front != rear){
        front = (front + 1)%size;
        return Q[front];
    }
    return 0;
}
```



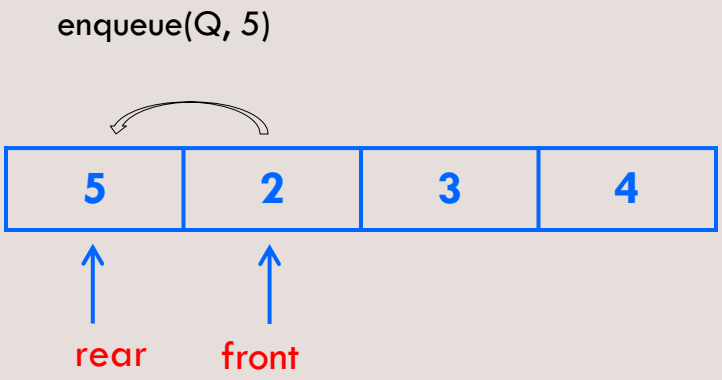
- **When is it empty or full?**

Queue Implementation

- **Circular Queue**

```
void enqueue(int Q[size], int x){
    rear = (rear + 1)%size;
    if (rear != front)
        Q[rear] = x;
}
```

```
int dequeue(int Q[size]){
    if (front != rear){
        front = (front + 1)%size;
        return Q[front];
    }
    return 0;
}
```



- **When is it empty or full?**

Contents

Queue library

- **C/C++** (#include <queue>)

```
#include <iostream>
#include <queue>
using namespace std;

int main(){
    queue <int> Q;

    for (int i = 0; i < 5; i++)
       Q.push(10*i);
    for (int i = 0; i < 3; i++){
       cout << Q.front() << " ";
       Q.pop();
     }

    return 0;
}
```

What's ouput?

Queue library

- ***Java*** (import java.util.*)

```
import java.util.*;

public class Queue_Demo {

    public static void main(String[] args){
        Queue Q = new LinkedList();

        for (int i = 0; i < 5; i++)
           Q.add(2*i+1);
        System.out.println(Q.remove());
        System.out.println(Q.peek());
    }

}
```

What's ouput?

Contents

Queue Application

- **Real life**
  - Waiting in line
  - Waiting on hold for tech support
- **More applications related to computer science**
  - Threads
  - Job scheduling (Round-Robin algorithm for CPU allocation)
  - Breath First Search

Queue Application

- **Job scheduling**

front	rear	Q[0]	Q[1]	Q[2]	Q[3]	Comments
-1	-1					queue is empty
-1	0	J1				Job 1 is added
-1	1	J1	J2			Job 2 is added
-1	2	J1	J2	J3		Job 3 is added
0	2		J2	J3		Job 1 is deleted
1	2			J3		Job 2 is deleted

Contents

Pratice

1. **Transform the Expression (4)** http://www.spoj.com/problems/ONP/
2. **Printer Queue(1840)** http://www.spoj.com/problems/PQUEUE/

Homework

1. **Seinfeld (5449)** http://www.spoj.com/problems/ANARC09A/
2. **Street Parade (95)** http://www.spoj.com/problems/STPAR/

Reference

- [wiki] Stack (abstract data type) https://en.wikipedia.org/wiki/Stack_(abstract_data_type)
- [wiki] Tower of Hanoi https://en.wikipedia.org/wiki/Tower_of_Hanoi