INTRODUCTION

☞ Chapter at a Glance

- **Data** (plural of "**datum**"), refers to qualitative or quantitative attributes of a variable or set of variables. Typically, data are the results of measurements and can be the basis of graphs, images or observations of a set of variables.
- **Metadata** is a data, containing a description of other data. Earlier term for metadata is ancillary data.
- **Data structure:** Data may be organized in many different ways. The logical or mathematical model of a particular organization of data is called data structure.
- **Classification:** Data structure can be classified as ***linear*** and ***non-linear.***
  Linear data structure: The data structure is said to be linear, if its element forms a sequence or linear list. Linear data structures organize their data elements in a linear fashion.
  Non-linear data structure: In nonlinear data structures, data elements are not organized in a sequential fashion. A data item in a nonlinear data structure could be attached to several other data elements to reflect a special relationship among them and all the data items cannot be traversed in a single run.
- Various operations can be performed on data structure. List of some frequently used operations are given below:
  1. Traversing: sometimes also called as visiting, means accessing the required record so as to process the data.
  2. Searching: finding the location of record to be processed.
  3. Inserting: adding a new record to the structure
  4. Deleting: removing the record from the structure.
  5. Sorting: arranging the record in some logical order.
  6. Merging: combining two different sets of records to form a final set of records.
- **Abstract data type (ADT):** Abstract Data Types are a set of data values and associated operations that are precisely independent of any particular implementation. The term abstract signifies that the data type will only set the rule of it's usage but how it will be used depends on the implementation. For example stacks and queues are called abstract data types. The stack data type defines two abstract methods PUSH and POP.
- **The basic properties of an algorithm** are:
  Input: Each algorithm is supplied with a zero or more external quantities.
  Output: Each algorithm must produce atleast one quantity.
  Definiteness: Each instruction must be clear and unambiguous.
  Finiteness: The algorithm must terminate after a finite number of steps within finite time.
  Effectiveness: Each instruction must be sufficiently basic and also be feasible.
- **Analysis of an algorithm** means, to determine the amount of resources (such as time and storage) necessary to execute it. Most algorithms are designed to work with inputs of arbitrary length. Usually the *efficiency or running time* of an algorithm is stated as a function relating the input length to the number of steps (time complexity) or storage locations (space complexity).
- In theoretical analysis of algorithms it is common to estimate their complexity in the *asymptotic* sense, i.e., to estimate the complexity function for arbitrarily large input. Big

notation, **omega n**
are used because
However the effic
related by a consta
- **Complexity:** Ther
complexity.
The ***time complex***
the algorithm.
The ***space comple***
completing the exe

1. Each element of
arr is 2000. The loca
a) 2820
Answer: (b)

2. In C language, ma
a) integer pointe
c) null pointer
Answer: (c)

3. Which of the follo
a) $O(n)$
Answer: (b)

4. Four algo.s do th
values of n?
a) $O(n^2)$
Answer: (d)

5. In C language, arr
a) Column major
Answer: (b)

6. Which of the follo
N?
a) O (N)
Answer: (b)

7. Which is better co
a) $O(n)$
Answer: (c)

notation, **omega** notation and **theta** notation are used for this. Usually asymptotic estimates are used because different implementations of the same algorithm may differ in *efficiency*. However the efficiencies of any two "reasonable" implementations of a given algorithm are related by a constant multiplicative factor called a *hidden constant*.

- **Complexity:** There are two types of complexities of an algorithm, time complexity and space complexity.

The ***time complexity*** of an algorithm is the amount of time the computer requires to execute the algorithm.

The ***space complexity*** of an algorithm is the amount of memory space the computer requires, completing the execution of the algorithm.

Multiple Choice Type Questions

1. Each element of an array arr[20][50] requires 4 bytes of storage. Base address of arr is 2000. The location of arr[10][10] when the array is stored as column major is
a) 2820 b) 2840 c) 4048 d) 4840 [WBUT 2008]
Answer: (b)

2. In C language, malloc( ) returns [WBUT 2009]
a) integer pointer b) structure pointer
c) null pointer d) void pointer
Answer: (c)

3. Which of the following is the best time for an algorithm? [WBUT 2010]
a) $O(n)$ b) $O(\log_2 n)$ c) $O(2^n)$ d) $O(n\log_2 n)$
Answer: (b)

4. Four algo.s do the same task. Which algo. should execute the slowest for large values of n? [WBUT 2010]
a) $O(n^2)$ b) $O(n)$ c) $O(\log_2 n)$ d) $O(2^n)$
Answer: (d)

5. In C language, arrays are stored in which representation? [WBUT 2011]
a) Column major b) Row major c) Layer major d) None of these
Answer: (b)

6. Which of the following algorithm should execute the slowest for large values of N? [WBUT 2012]
a) O (N) b) O (N^2) c) O($\log_2$N) d) None of these
Answer: (b)

7. Which is better computing time (in analysis of algorithm)? [WBUT 2013]
a) $O(n)$ b) $O(2n)$ c) $O(\log 2n)$ d) None of these
Answer: (c)

8. A dynamic data structure where we can search for desired records in $O(\log n)$ time is [WBUT 2018]
a) heap b) binary search tree
c) circularly linked list d) array
Answer: (b)

9. For Column Major, what is the address of [3, 2] th element of a 3×4 Matrix (contains integer number)? It is given that the 'Base Address is 2000'. [WBUT 2017]
a) 2010 b) 2012 c) 2016 d) 2018
Answer: should be 2020

10. Which of the following expressions access the (i, j)th entry of a (m × n) matrix stored in column major order? [WBUT 2015]
a) n × (i – 1) + j b) m × (j – 1) + i
c) m × (n – j) + j d) n × (m – i) + j
Answer: (b)

11. Which of the following is non-linear data structure? [WBUT 2017]
a) Stacks b) List c) Strings d) Trees
Answer: (d)

Short Answer Type Questions

1. Let the size of the elements stored in an 8 × 3 matrix be 4 bytes each. If the base address of the matrix is 3500, then find the address of A [4, 2] for both row major and column major cases. [WBUT 2007, 2015, 2020]
What is sparse matrix?
OR,
What are sparse matrices? How such a matrix is represented in memory? What are the types of sparse matrices? [WBUT 2018]
Answer:
1st Part:
In row major order the address will be

$$\text{Loc}(A(i,j)) = \text{Lo} + ((i-1) * n + (j-i)) * c$$

Therefore

$$\text{Loc}(A(4,2)) = 3500 + ((4-1) * 3 + (2-1)) * 4 = 3500 + (10) * 4$$
$$= 3500 + 40 = 3540$$

In column-major order the address will be

$$\text{Loc}(A(i,j)) = \text{Lo} + ((j-i) * m + (i-1)) * c$$

Therefore

$$\text{Loc}(A(4,2)) = 3500 + ((2-1) * 8 + (4-1)) * 4 = 3500 + (11) * 4$$
$$= 3500 + 44 = 3544$$

2nd Part:
A matrix is represented i
column.
If there are m rows and n
x n elements. This m x n
A matrix, that is filled wi
sparse matrix.
Now, if there is a sparse r
store this matrix, then the
Now, a sparse matrix can
In this way we can store o

Let us consider a matrix M
Here the number of non-ze
The array A [0...t, 1...3] c

	1	2	3
A[0]	5	6	5
A[1]	1	5	1
A[2]	2	1	2
A[3]	3	2	–3
A[4]	4	4	7
A[5]	5	6	30

2. Show that the function
$f(1) = 1$
$f(n) = f(n$
has the co
Define Big – O, Ω, θ nota
three representatives.
And, Define Big O notation

2nd Part:

A matrix is represented in memory as 2D array, like A [i, j], where i is a row and j is a column.

If there are m rows and n columns, then total elements in a matrix or size of a matrix is m x n elements. This m x n elements requires m x n units of storage.

A matrix, that is filled with mostly zeroes (more than 2/3 elements are zero), is called as a sparse matrix.

Now, if there is a sparse matrix of order m × n, and if we use a 2D array of order m × n to store this matrix, then there is basically wastage of large amount of memory.

Now, a sparse matrix can be stored as a list of three tuples of the form (i, j, value).

In this way we can store only non-zero elements.

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & -3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 7 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 30 \end{bmatrix}$$

Let us consider a matrix M(5×6)

Here the number of non-zero elements t = 5.

The array A [0…t, 1…3] can store all the non zero elements as given below:

	1	2	3
A[0]	5	6	5
A[1]	1	5	1
A[2]	2	1	2
A[3]	3	2	–3
A[4]	4	4	7
A[5]	5	6	30

The element A [0, 1] and A [0, 2] contain the number of rows and columns of the matrix. A[0, 3] contains total number of non-zero items.

2. Show that the function $f(n)$ defined by: **[WBUT 2007, 2010, 2016, 2018]**

$$f(1) = 1$$

$$f(n) = f(n-1) + \frac{1}{n} \text{ for } n > 1$$

has the complexity O (log n).

Define Big – O, Ω, θ notations. Explain the conceptual differences among these three representatives. **[WBUT 2007]**

And, Define Big O notation. **[WBUT 2010, 2012]**

Answer:

1st Part:

$f(n) = f(n-1)+1/n$

$= f(n-2)+1/(n-1)+1/n$

$= f(n-3)+1/(n-2)+1/(n-1)+1/n$

.....

.....

$= f(2-1)+1/2+1/3+.....+1/(n-1)+1/n$

$= 1/1 + 1/2 +1/3 +.....+ 1/n$

These types of numbers are called Harmonic numbers.

We can evaluate this type of series just by integrating 1/x from ½ to n + ½

After integrating, we get the result as $\ln(n + ½) - \ln ½$

$\approx \ln n - 0.7$

Hence, we can write the complexity as O(log n).

Big O notation:

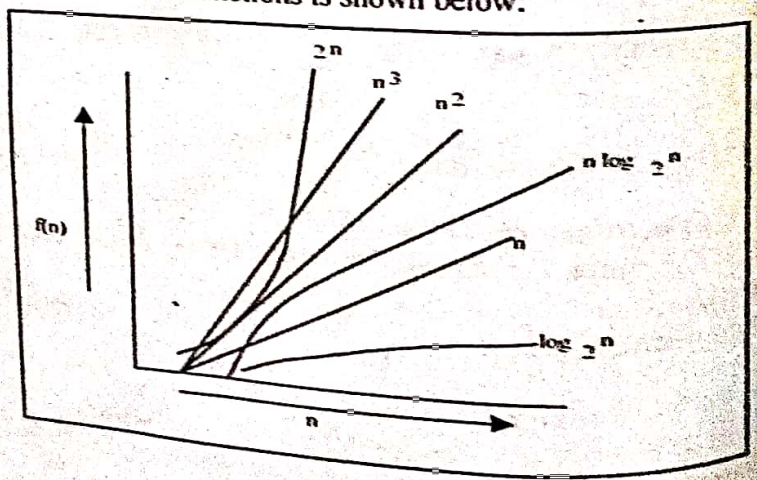
The Big Oh (the "O" stands for "order of") notation is used to classify functions by asymptotic growth function and hence finding the time complexity of an algorithm.

There are two types of complexities of an algorithm, time complexity and space complexity. The time complexity of an algorithm is the amount of time the computer requires to execute the algorithm. The space complexity of an algorithm is the amount of memory the computer needs to run to complete the execution of the algorithm. The algorithms are compared based on their performances. The performance is measured in terms of time complexity & space complexity. Based on the nature of the input, the complexity can be of three different types: best case, average case and worst case.

The different computing functions are measured as:

$$n, n^2, n^3, \log_2 n, n\log_2 n, 2^n.$$

The rate of growth of all these functions is shown below:



Ω:

Let $f(n)$ and $g(n)$ be fu

and only if $g(n) = O(f(n$

θ:

Let $f(n)$ and $g(n)$ be fu

and only if $g(n) = O(f(n$

3. Derive values relate

Sort algorithm. Also, c

Write an algorithm for

technique and find its t

Answer:

The algorithm for bubble

```
Procedure: BubbleSo
 Inputs: List[] - A
 Locals: i, j - int
 Begin:
    For i = 0 to Li
       For j = i+1
          If List[i
             Swap L
          End If
       Next j
    Next i
```

Analysis:

The size of the sorting pr

expect the execution time t

The innermost instruction i

this always takes the sam

respect to N. Surrounding

body, or does not; in the w

O(1).

The if statement is executed

executes as many times as

inner loop executes $N - 1$ t

on.

$$O\left(\sum_{i=0}^{N-1}(\right.$$

numb

So the complexity is O(N2) i

The best case would be if the

there will be comparisons as

But if we keep a track of ex

so there are no exchanges, th

Ω:
Let $f(n)$ and $g(n)$ be functions, where n is a positive integer. We write $f(n) = \Omega(g(n))$ if and only if $g(n) = O(f(n))$.

θ:
Let $f(n)$ and $g(n)$ be functions, where n is a positive integer. We write $f(n) = \theta(g(n))$ if and only if $g(n) = O(f(n))$. and $g(n) = \Omega(f(n))$.

3. Derive values related to the average case and worst case behaviour of Bubble Sort algorithm. Also, confirm that the best case behavior is O(n). [WBUT 2007]

OR,

Write an algorithm for sorting a list numbers in ascending order using bubble sort technique and find its time complexity. [WBUT 2017]

Answer:

The algorithm for bubble sort is given as follows:

```
Procedure: BubbleSort(List[])
 Inputs: List[] - A list of numbers
 Locals: i, j - integers
 Begin:
    For i = 0 to List.Size-1
       For j = i+1 to List.Size-1
          If List[i] > List[j], Then
             Swap List[i] and List[j]
          End If
       Next j
    Next i
```

Analysis:

The size of the sorting problem is related to the size N of the list. As N increases, we expect the execution time to increase as well.

The innermost instruction is to swap two list elements. Regardless of how long the list is, this always takes the same amount of time, and so the swap instruction is O(1) with respect to N. Surrounding that instruction is an if statement, which either executes its body, or does not; in the worst case, it will execute its body. Again, the if statement is O(1).

The if statement is executed as many times as the inner loop iterates, and the inner loop executes as many times as the outer loop iterates. During the first iteration ($i = 0$), the inner loop executes $N - 1$ times, during the second iteration ($i = 1$), $N - 2$ times, and so on.

$$O\Big(\underbrace{\sum_{i=0}^{N-1}(N-i-1)}_{\text{number-of-loops}} \times \Big(\underbrace{\frac{1}{if}}_{\text{time-per-loop}} + \underbrace{\frac{1}{swap}}_{}\Big)\Big) = O(N^2 - N).$$

So the complexity is O(N2) in the average case as well worst case.

The best case would be if the list were already sorted.

There will be comparisons as it is but no exchanges and execution time is in O(N2).

But if we keep a track of exchanges in each pass and terminate the program checking if no there are no exchanges, then the program would require only one pass and max. (N-1)

comparisons are required in that single pass and we can say that the complexity is order of O(N).

4. What is linear data structure? [WBUT

Answer:
A data structure is said to be *linear* if its elements form a sequence or a linear list.
Examples:

- Arrays
- Linked Lists
- Stacks, Queues

The Operations that can be done on linear structures are:

- *Traversal*: Travel through the data structure
- *Search*: Traversal through the data structure for a given element
- *Insertion*: Adding new elements to the data structure
- *Deletion*: Removing an element from the data structure
- *Sorting*: Arranging the elements in some type of order
- *Merging*: Combining two similar data structures into one.

5. Prove that, $O(f(x)) + O(g(x)) = O(max(f(x),g(x))$ [WBUT

Answer:
Suppose q(x) is in O(f(x) + g(x)) then q(x) is in O(max(f(x), g(x)) because two times greatest of f(x) and g(x) is greater than or equal to the sum. More precisely:

$q(x) <= c(f(x) + g(x)) <= 2c(max(f(x), g(x))$.

Next, suppose q(x) is in O(max(f(x), g(x)) then q(x) is in O(f(x) + g(x)) because greatest of f(x) and g(x) is less than or equal the sum of the two (assuming the functions are positive valued). In formulas:

$q(x) <= c \max(f(x),g(x),h(x)) <= c(f(x) + g(x))$.

Hence proved.

6. What is an Abstract Data Type? What do you mean by a Dynamic Structure? [WBUT

Answer:
An **abstract data type (ADT)** is an object with a generic description independent of implementation details. If for a particular collection of data only the structure of the functions to be performed on the data is defined but the implementation is defined, then such a collection of data is called Abstract data type.
In dynamic Data Structure memory allocation for the data structure takes place at time and only required amount of memory is allocated.
E.g.: Link lists, Stacks, Queues, Trees

7. $T(n) = 4n^2 + 3n \log n$, express T(n) in Big (O) notations. [WBUT

Answer: In Big (O) notation the complexity is $O(n^2)$.

8. Write the differenc

Answer:
The main difference b
refers to an array of pa
the fact that multidim
number of elements
necessarily present in a

9. Do a comparison a

Answer:
Data type consists of t
or character data types
operations that can be
for addition, subtracti
perform these arithmet
part of the integer data
A data type can be co
and its implementation
another for, e.g., effic
program). Thus, speal
considering it irrelevant
A data structure is an a
An example of several
binary trees, and hash t
can be implemented as t

10. What are the differ

Answer:
Data structure can be cla
Linear data structure:
sequence or linear list.
memory. Here the eleme
(1) Linear relationship
memory locations.
(2) Linear relationship
links.
Non-linear data structu
relationship between ele
they are traversed in a n
start from the root but w
both.
Ex: Graphs and Trees.

8. Write the difference between a[ ] [ ] and **a. [WBUT 2012]**

Answer:

The main difference between multidimensional arrays (a[][]) and arrays of pointer (**a) refers to an array of particular datatype arrays (a pointer to pointers to that datatatype) is the fact that multidimensional arrays are rectangular data structures, with the same number of elements allocated for each specified dimension. This constraint is not necessarily present in an array of pointers.

9. Do a comparison among Data Type, Abstract Data Type and Data structure.
[WBUT 2013]

Answer:

Data type consists of the values it represents and the operations defined upon it.. Integer or character data types are found in nearly all computers, and they have well-defined operations that can be done with them. For example, the integer data type has operations for addition, subtraction, multiplication, and division. The computer knows how to perform these arithmetic operations with any two integers because these operations are part of the integer data type.

A data type can be considered **abstract** when it is defined in terms of operations on it, and its implementation is hidden (so that we can always replace one implementation with another for, e.g., efficiency reasons, and this will not interfere with anything in the program). Thus, speaking about such a type, we leave its implementation aside considering it irrelevant to the topic, unless we directly discuss the implementation.

A data structure is an arrangement of data in a computer's memory or even disk storage. An example of several common data structures are arrays, linked lists, queues, stacks, binary trees, and hash tables. **Data Structure** is an implementation of ADT. Many ADT can be implemented as the same Data Structure.

10. What are the differences between linear and non-linear data structures?
[WBUT 2013, 2014]

Answer:

Data structure can be classified as linear and non-linear.

Linear data structure: The data structure is said to be linear, if its element forms a sequence or linear list. There are two basic ways of representing such structure in memory. Here the elements are traversed sequentially starting from the beginning.

(1) Linear relationship between data elements is represented by means of sequential memory locations. Ex: Arrays.

(2) Linear relationship between data elements is represented by means of pointers and links. Ex: Linked list.

Non-linear data structure: They are used to represent the data having a hierarchical relationship between elements. Here the elements are not traversed sequentially, rather they are traversed in a non linear fashion. For example, in case of the tree, we have to start from the root but we have to traverse either left subtree or right subtree, but not the both.

Ex: Graphs and Trees.

11. What are the characteristics of algorithm? [WBUT

Answer:

- **Precision** – the steps are precisely stated(defined).
- **Uniqueness** – results of each step are uniquely defined and only depend on the and the result of the preceding steps.
- **Finiteness** – the algorithm stops after a finite number of instructions are execute
- **Input** – the algorithm receives input.
- **Output** – the algorithm produces output.
- **Generality** – the algorithm applies to a set of inputs.

12. a) Suppose one 2-D array is initialized as int a [5][7]; Base address is 4000 the location of element a [2][4] in row major form and column major form.
b) Define Sparse Matrix. [WBUT

Answer:

a) Assume that the size of each element is 4 bytes.

In row major order the address will be

$$\text{Loc}(A(i,j)) = Lo + ((i-1) * n + (j-1)) * c$$

Therefore,

$$\text{Loc}(A(2,4)) = 4000 + ((2-1) * 7 + (4-1)) * 4 = 4000 + (7+3) * 4$$
$$= 4000 + 40 = 4040$$

b) *Refer to Question No. 1(2nd Part) of Short Answer Type Questions.*

13. What is the default return type of malloc( )? Why do we need to typecast it [WBUT

Answer:

`malloc` returns a void pointer (`void *`), which indicates that it is a pointer region of unknown data type.

The (**char***) or (int*) is an explicit type conversion, converting the p returned by malloc from a pointer to anything, to a pointer to char or integer is unnecessary in C, since it is done implicitly, and it is recommended not this, since it can hide some errors.

14. If $T(n) = a_0 + a_1x + a_2x^2 + a_3x^3 + ... + a_nx^n$, then prove $T(n) = \Theta(x^n)$. [WBUT

Answer:

To prove that $f(n) = \Theta(g(n))$ we have to find $c1, c2 > 0$ and n_0 such that $0 \le c_1g(n) \le f(n) \le c_2g(n)$.

Now c_1 and c_2 are two valid constants for $n \Rightarrow n_0$:

$$c_1 = \frac{1}{2^k}a_k \qquad c_2 = \left(2 - \frac{1}{2^k}\right)a_k$$

where

$$n_0 = 2 \cdot \max\left\{\sqrt[i]{\frac{|a_i|}{a_k}}\right.$$

Hence proved.

1. Write short note on Abs
Answer: *Refer to Questio*

2. Define pseudo-code.
Answer:
Actual purpose of using ps conventional programming independent description of t Pseudo-code is an artificia algorithms. It is a "text-base and informal high-level des structural conventions of a rather than machine reading human understanding of th code and subroutines.

Examples:

```
1. If student's grade
Print "passed"
else
Print "failed"

2. Set total to zero
Set grade counter to
While grade counter i
Input the next grade
Add the grade into th
Set the class average
Print the class averag

3. Initialize total t
Initialize counter to
Input the first grade
While the user has not
Add this grade into th
Add one to the grade c
```

where

$$n_0 = 2 \cdot \max\left\{ \sqrt[i]{\frac{|a_i|}{a_k}} \;\middle|\; 0 \le i \le k-1 \right\}$$

Hence proved.

Long Answer Type Questions

1. Write short note on Abstract Data Type. **[WBUT 2012, 2015]**

Answer: *Refer to Question No. 6 of Short Answer Type Questions.*

2. Define pseudo-code. **[MODEL QUESTION]**

Answer:

Actual purpose of using pseudo-code is that it is easier for humans to understand than conventional programming language code, and that it is an efficient and environment-independent description of the key principles of an algorithm.

Pseudo-code is an artificial and informal language that helps programmers develop algorithms. It is a "text-based" detail (algorithmic) design tool.Pseudo-code is a compact and informal high-level description of a computer programming algorithm that uses the structural conventions of a programming language, but is intended for human reading rather than machine reading. Pseudo-code typically omits details that are not essential for human understanding of the algorithm, such as variable declarations, system-specific code and subroutines.

Examples:

```
1. If student's grade is greater than or equal to 80
Print "passed"
else
Print "failed"

2. Set total to zero
Set grade counter to one
While grade counter is less than or equal to ten
Input the next grade
Add the grade into the total
Set the class average to the total divided by ten
Print the class average.

3. Initialize total to zero
Initialize counter to zero
Input the first grade
While the user has not as yet entered the sentinel
Add this grade into the running total
Add one to the grade counter
```

```
input the next grade (possibly the sentinel)
if the counter is not equal to zero
set the average to the total divided by the counter
print the average
else
print 'no grades were entered'

4. initialize passes to zero
initialize failures to zero
initialize student to one
while student counter is less than or equal to ten
input the next exam result
if the student passed
add one to passes
else
add one to failures
add one to student counter
print the number of passes
print the number of failures
if eight or more students passed
print "raise tuition"
```


Chapter a

- **Stacks:** A stack is
  are implemented
- The **PUSH** operat
  The **POP** operatio
  A **STACK** is call
- A stack may be
  element of the sta
  linked stack. The

```
typede
{
    int
    str
} lsta
lstack
```

- **Various types of**
  operations (operat
  Infix notatio
  Prefix notati
  Postfix notat
- **Conversion from**
  corresponding pos
  - Fully parenth
  - Move all ope
  - Delete all par

  The priorities of di

- **Priority Queue:**
  associated with it
  speak of one item
  which is the highe
  queue in any, arbit
  their priorities sta
  priority are process
- **De-queue:** De-que
  from either end of