WEEK 6 : BACKTRACKING, SCOPE, DATA STRUCTURES: STACK, QUEUES AND HEAPS.

Lecture 2 : Global Scope, Nested Functions :

Variable declared as global anywhere, can be accessed and changed anywhere.

Function b and c, nested inside a function a, are accessible only in function a.

Lecture 3 : Generating Permutations :

1. From the right, identify the first decreasing position :

D c h b a e g l k o n m j i

1. Swap that value with its next larger letter to its right.

* D c h b a e g l k o n m j i
* D c h b a e g l m o n k j i

1. Reverse the increasing suffix:

D c h b a e g l m i j k n o

Example :

Ques : f j a d b I h g e c

1. Find : f j a d b I h g e c
2. Find value greater than it : f j a d b I h g e c
3. Swap : f j a d c I h g e b
4. Reverse : f j a d c b e g h i

Lecture 4 : Sets, stacks, queues.

Sets :

Even = {2,4,6} odd = {1,3,5}

union : | : odd | even = {1,2,3,4,5,6}

Intersection : & : odd & even = {}

Exclusive or : ^ : elements which are not common to both sets. (vice-versa of intersection)

Set difference : odd – even = {}

Stack(list) : LIFO

Push(s,x) : add x to s = append x

Pop(s) : return most recently added element = s.pop()

QUEUE(list) : FIFO

Add(q,x) : adds x to rear of q

Remove(q) : removes head of q

Left is rear, right is front.

Add(q,x) is q.insert(0,x)

q.insert(j,x) is insert x before position j

(QUEUE in Python is reverse of what queue is in real life)

Remove(q) is q.pop()

LECTURE 5 : Priority queues and Heaps :

Priority queue doesn’t have FIFO, the element with high priority leaves the queue first.

Operations :

1. Delete\_max():

Identify and remove job/element with highest priority,

Need not be unique

1. Insert()

Add new element to list

* Unsorted list :

Insert() takes O(1) time

Delete\_max takes O(n) time

* Sorted List :

Delete\_max() takes O(1) time

Insert() takes O(n) time.

Processing a sequence of n jobs requires O(n2) time.

Binary tree : 2D structure

Heap : Special Kind of binary tree

* Balanced : N node tree has height logN
* Insert : delete\_max : O(logN)
* Processing n jobs takes time O(NlogN)
* Truly flexible, need not fix upper bound for N in advance.

Heap : Binary tree filled level by level, left to right.

* At each node, value stored is bigger than both children.

No holes are allowed in heap.

Build heap : heapify()

Heapify() takes O(NlogN) time.

Application of Heap is Heap Sort.

WEEK 6 Quiz :

1 . Suppose u and v both denote sets in Python. What is the most general condition that guarantees that u - (v - u) == u?

Ans : *This is true for any u and v.*

* Suppose u and v both denote sets in Python. What is the most general condition that guarantees that u|v == u^v?

Ans : *The sets u and v should be disjoint.*

Suppose we insert 19 into the min heap [17,25,42,67,38,89,54,98,89]. What is the resulting heap?



* Ans : original heap :
* *17*
* */ \*
* *25 42*
* */ \ / \*
* *67 38 89 54*
* */ \*
* *98 89*

*After inserting 19, the new heap is*

*17*

*/ \*

*19 42*

*/ \ / \*

*67 25 89 54*

*/ \ /*

*98 89 38*

Suppose we execute delete-min twice on the min-heap [13,29,24,67,52,89,45,98.79,58]. What is the resulting heap?



No, the answer is incorrect.  
Score: 0

Feedback:

*The original heap is:*

*13*

*/ \*

*29 24*

*/ \ / \*

*67 52 89 45*

*/ \ /*

*98 79 58*

*After one delete-min, we have:*

*24*

*/ \*

*29 45*

*/ \ / \*

*67 52 89 58*

*/ \*

*98 79*

*After the second delete-min, we have:*

*29*

*/ \*

*52 45*

*/ \ / \*

*67 79 89 58*

*/*

*98*