**Lecture one**

1. 6046 after this series. About algorithm design and implementation.
2. Peak finding: One dimensional  
   The straight forward algorithm is of O(n) complexity (linear). Note: The O represents theta in these notes.  
   Better way: Divide and conquer. Something like the binary search. (29.35) The base case is T(1) = O(1) but the complexity is O(log n base 2).
3. Peak finding: Two dimensional  
   The straight forward algorithm is known as the Greedy Ascent Algorithm. The complexity is O(nm) or O(n^2). Attempt #2 (49.50) is: pick the middle column, find the maximum in that column, compare with elements to the right and left, move your column based on this.

**Lecture two (Hella informative maaan)**

1. Al-Khwarizimi was the first one to use an algorithm in a textbook of his. The word was coined from his name. Algorithm is a computational procedure for solving a problem. (5.55)
2. Random Access Machine (RAM): Load constant words. Do constant computations, store constant words.   
   Pointer Machine: Dynamically allocated objects, has constant fields, fields with words (eg. Int) or pointers.  
   Python model: 22.21
3. Document distance problem: Find the distance (difference) between two documents  
   Document is a sequence of words and a word is a string of alphanumeric chars  
   Split a document into words, compute word frequencies, compute dot product, divide

**Lecture three**

1. Applications of sorting
2. Insertion sort: Swap pair wise to get to the required position. To improve this algo, use binary search to find the position to insert into. (Read about vanilla sort, In place merge sort)
3. Merge sort requires auxiliary space of O(o), Times computed (45.56)
4. Recurrences: Two types, all the work done in the top level or all the work done in the leaves.

**Lecture four**

1. Priority queue: Using a heap. Then move onto a heap sort. The functions associated are; Insert, Max, ExtractMax, IncreaseKey.
2. A heap is an array visualized as an almost complete binary tree
3. Max Heap Property: The key of a node is greater than or equal to the keys of its children. Simillarly, Min Heap Property
4. Heap operations: Build max heap, Max heapify (The condition for this to work is that the subtrees have to already be max heaps, what the function does is, it exchanges the sub root with the highest child)
5. Build a max heap, pop the root and then repeat. Build a max heap from n/2 to the root.
6. A better analysis (40.04)

**Lecture five**

1. Binary Search Trees: Problem – Runway reservation problem – (3.24)
2. Discrediting other data structures for this problem
3. BST- The invariant says that all children to the left are lower in value than the parent and all children on the right are higher in value than the parent.
4. While inserting into a BST, you can perform tests. (k-min check for this problem)
5. New req. – Calculate the rank of the BST.  
   Add a new data member called size. It counts the number of nodes below it (including itself)  
   While adding/deleting, traverse and add one to the size. To find rank, walk down the BST, add in the nodes that are smaller and add in the subtrees to the left.

**Lecture six (Tough Theory)**

1. Inorder traversal is used to print the elements in order.
2. Height of a tree and height of a node
3. Height of the BST = max(height of left child, height of right child) + 1
4. AVL Trees- Heights of every node to be only a difference on 1. (subtrees are of similar height)
5. Rotate a BST node- 40.47-**IMPORTANT**

**Lecture seven**

1. Comparison model – Only allowed operation is a compare, time cost is considered to be the no. of comparisons
2. Analogy of a decision tree and algorithm (14.48)
3. Proving that any search can be done in logn time (best case) - Decision tree would have to have at-least n answers and therefore the height would be at-least logn height which is the worst case
4. Proving that any sort can be done in nlogn time – Decision tree is binary and the no. of leaves is at-least the number of possible answers. That is n! This means that the height is at-least logn!
5. Linear time sort or Integer sorting – Using counting sort, have an array with all the keys and increment the value for every occurrence. Print the array in order with the position values. O(n+k) is the order of the counting sort. Radix Sort. Uses counting time as a sub-routine. Sort each integer by the least sig. figure and then move upto the most significant digit. Use the base as the number of elements to be sorted to get close to linear time.

**Lecture eight**

1. Dictionary – Abstract data structure, if there is a key clash, it overwrites the existing one
2. Uses – Databases, compliers and interpreters, network routers, servers, login details
3. Problems – Keys may not be integers, gigantic memory hog
4. Pre-hash function in Py – Key’s should not change over time
5. Idea – The size of the dictionary should be around the number of the keys
6. Chaining – Dealing with collisions, by defining the hash function correctly, we can ensure that the hash table is evenly filled, n is the number of keys, m is the number of slots in the hash table. Constant time if m is roughly equal to n. This can be achieved if m/n is around 1. This would mean that there are roughly 1 elements for each m.
7. Hashing functions(6.046) – Division method(k mod m), good if m is prime, not close to power of 2 or 10, Multiplication method(((a.k)mod2^w)>>(w-r)), where word size is w, where m=2^r, it’s good if a is odd and not close to a power of 2, Universal Hashing(((ak+b)mod p))mod m, where a and b are random numbers b/w 0 and p-1, where p is a prime number > size of the key universe

**Lecture nine (First part unclear)**

1. n/m gives you the load factor or the number of elements mapped to the same key.
2. Choosing m is the key to get a constant time function: Use table doubling.
3. Amortization: The average run time for a batch of operations
4. String matching: Simple algorithm is to find all the substrings of the same length as s in t and check each one. Time taken is O(len(t)\*len(s)).
5. For this, use rolling hashes to get constant time. It’s called Karp-Rabin hashing algorithm.
6. Use the division hashing method by choosing m as a prime number greater than the length of s (51.2)

**Lecture ten**

1. Implement a hash table using an array – Open Addressing
2. Another way to deal with collisions, no. of slots has to be greater than no. of elements.
3. Hash function takes key and no. of trials as arguments. Insert and search is the same but Delete function has a subtle difference. Use a DeleteMe flag instead of None after deleting an element.
4. Fundamental principle of Open addressing is that the same probing function gives the same hash values for the same key. (Essentially same order of probing.)
5. Probing strategies (A hash applicable to open addressing): Double hashing.

**Lecture eleven**

1. Big number computations – Irrationals, Catalan numbers – Set P of balanced parenthesis strings (Cardinality of this set, C0 = 1)
2. Catalan Numbers – Cn+1 = Sumation(Ck x Cn-k),k from 0 to n, 42 is a Catalan Number too!
3. Newton’s method – Using derivatives, quadratic convergence (Precision doubles)