Notes Algoritms and Data Sctructues

## Lecture 1

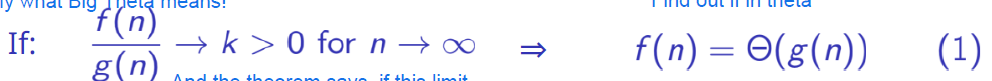
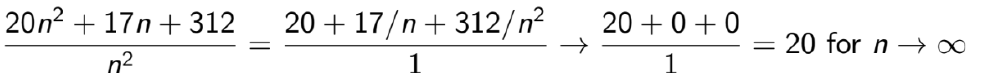
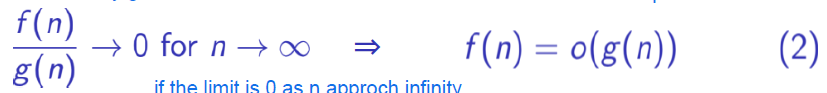
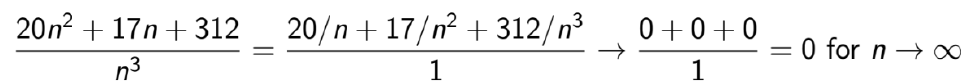
* What is algoritms
  + Algoritms is a set of step by step to solve a problem
* What is data structure?
  + A way to store data
* What is primitive data?
  + Provided by programming languages, fx int, floating
* What is abstract data?
  + Higher level, fx que, stack, list, arrays
* What is time complexity?
  + How fast It runs, on the amount of data it is working on
* Space complexity?
  + Amount of memory depending on the data it is working on
* What is divide and conque?
  + Break the problem into smaller problems, and combine the solutions
* What is Brute force?
  + Trying all possible solutions and selecting the best.
* What are the minimum requirements for an algorithm?
  + Correctness:
    - It is never an infinity loop
    - It gives an output when it stops(provide and answer to out problem).
* What does quality mean, regarding algorithms?
  + Time
* What is time for an algorithm?
  + Number of basic operations performed
* What is space for an algorithm?
  + Maximum occupied memory cells.
* What is a time function?
  + A function that tells how the algorithm performs when data increases
* What is the greedy algorithm?
  + Take a step by step, by choosing the locally optimal choose. Without consider the future.
* What is a permutation?
  + A list of numbers
* What is a cycle?
  + You decide what to move first, and move that element to it’s correct position. The one you need to pick up to make space for the first element, you now place at it’s correct position and so on
  + Imagine we have a permutation: Permutation: [5, 1, 3, 4, 2 And the solved arrangement is: Solved: [1, 2, 3, 4, 5]
    - Start with position 0 (value 5):
    - The number 5 belongs at position 4.
    - Move 5 to position 4.
    - Pick up the number currently at position 4 (value 2):
    - The number 2 belongs at position 1.
    - Move 2 to position 1.
    - Pick up the number currently at position 1 (value 1):
    - The number 1 belongs at position 0.
    - Move 1 to position 0.
    - The cycle is now complete:
    - All numbers (5 → 2 → 1 → 5) are in their correct positions.
  + If cycle is length one, then the piece of the puzzle is in the correct position
* What is the formula for the expected cycles:
  + Et billede, der indeholder tekst, ur, Font/skrifttype, nummer/tal

    Indhold genereret af kunstig intelligens kan være forkert.

## Lecture 2

* What is the perks of sorting?
  + Finding values faster
* What is the difference between array and list in python?
  + Can use mathematical operations on arrays
  + Arrays can also be 2D, list can not
  + Arrays only take one datatype, list take multiple different datatypes.
* What is a sorting key?
* What is insertion sort?
  + You simply shift the array to the right, and move the element to the left, based on the value. (just like a hand of cards)
  + So you always compare the element to the value to the left of it, until the left value is no longer bigger.
* What is the formula for insertion sort?
  + C1n2
    - Where c1 is a constant that does not depend on n
* What is best case for insertion sort?
* What is worst case for insertion sort?
* What is selection sort?
  + Find the smallest move it to the left
  + Find the next smallest move it to the left after the one you started with
* Is there a difference of merge algorithm and merge sort?
  + Yes
* What is merge algorithm
  + Divide the unsorted array into two sub-arrays, half the size of the original.
  + Continue to divide the sub-arrays as long as the current piece of the array has more than one element.
  + Merge two sub-arrays together by always putting the lowest value first.
  + Keep merging until there are no sub-arrays left.
* What is the formula for merge sort algorithm?
  + C2\*n\*log2(n)
* Where comes the 2 from in log2 in merge sort algorithm?
  + Because we divide each array with 2, to split it in half
* What is best for low inputs, and what is best for high inputs insertion sort or merge sort?
  + Et billede, der indeholder tekst, skærmbillede, Kurve, linje/række

    Indhold genereret af kunstig intelligens kan være forkert.
    - Insertion sort for low input and merge sort for high inputs
* what is asymptotic analysis?
  + analysis on how the algorithm perform without worring about hardware etc.
  + How does an algorth perfom when the input size grows?
* How do we find the average runtime ?
  + By finding the best case
  + and the worst case
  + Often they will be the same fx O(n2) and (n2).
  + If they are not the same, then the average runtime don’t exist.
* what is the RAM model?
  + Random Access machine model
  + A theoretic model that computer scientist use to calculate the effiency of an algorithm
    - sequential execution – every thing is execute on thing after another, no parrallism.
    - Unilimited memory and it take constant time to acees different things in memory
    - Basic operations takes the same time, fx add, subtract, multiply etc.
* Does the multiplicative constants matter when growth rates are different?
  + No’
* If we have to algorithms, where on is fastest on low inputs and one is fastes on high inputs, what should we ussullay use?
  + The one that is fastest with hight input almost always
* what does it mean to scale a function?
  + That we look at them as c \* f(n)
    - a multiplicative constant multiplied by the function
* what is a class of a function?
  + In plain English, the class of f(n) is the collection of *every* function you can get by multiplying f(n) by any positive constant number
* What is big o notation ( O)?
  + It is a way to understand worst case scenario for our algorithms, so how does the runtime grow as the input size (n) grows
  + The algortithm wont take longer than this
    - The run time is to O.
    - f(n)
* What is omega in runtime?
  + This is the best case
  + The run time is bigger then or equal to omega
* What is Thetha notation ?
  + Right in between the worst and best case, so this is how long if often will take an algorithm, f(n):
  + Et billede, der indeholder diagram, linje/række, Kurve

    Indhold genereret af kunstig intelligens kan være forkert.
  + The average case
  + Run time is often f(n) =
  + If an only if
    - Meaning it should be between big O and omega
* what is little o (o)
  + This is the strictly upper bound, the growth rate of a function will always be less
* little omega
  + The strictly lower bound, the growth rate will be strictly bigger than this¨
* How can I check if it is thetha in practice?
  + 
  + 
  + Should be a constant bigger then zero then thetha
* How can I check if it is little o in practice? (strictly greater)
  + 
  + 
  + If it is equal to zero then little o.
* what is the hierarchy of growth rate of functions?
  + Et billede, der indeholder tekst, skærmbillede, Font/skrifttype, linje/række

    Indhold genereret af kunstig intelligens kan være forkert.
    - Every one grows strictly slower then the one then the next one
      * 1 slower then log n, log n slower then etc.
  + Et billede, der indeholder tekst, Font/skrifttype, linje/række, nummer/tal

    Indhold genereret af kunstig intelligens kan være forkert.
* What term is the dominant term, when we have multiple terms (with +)
  + The one with the highest growth rate
  + dictates the **overall asymptotic growth rate** of the entire function.
  + 
    - f(n) grows faster then h(n), because 700 > 600.
    - And the other terms in h(n) are less important
* What is a linear search?
  + Checking all the elements step by step from 1-n
* What runtime does a nested loop:   
  for n  
   for n
  + O(n2), because each loop takes n time, because we iterate over n elements.
    - So n\*n = n2
* How do we write a constant runtime with big O notation?
  + O(1), because we don’t care about specific value.
* What does it mean to have an inversion?
  + **a larger number appears before a smaller number** in the array (i.e., they are out of order).
  + Et billede, der indeholder Font/skrifttype, tekst, typografi, Grafik

    Indhold genereret af kunstig intelligens kan være forkert.
* What can we use inversion to?
  + Calculate runtime, because we can see how (un)sorted an array/list is.
* What is this:  
  Et billede, der indeholder tekst, Font/skrifttype, håndskrift, hvid

  Indhold genereret af kunstig intelligens kan være forkert.
  + The run time for Insertions sort, which equals n2  
    Et billede, der indeholder tekst, Font/skrifttype, skærmbillede, nummer/tal

    Indhold genereret af kunstig intelligens kan være forkert.
* Where does the runtime speed of merge-sort come form?
  + By dividing it into halfs, thereby less comparing
* What three steps do we usullay go through when making a algorithm?
  + Et billede, der indeholder tekst, Font/skrifttype, skærmbillede, hvid

    Indhold genereret af kunstig intelligens kan være forkert.
* If we divide to functions to check it asymptotic and it goes to zero what can we say?
  + Et billede, der indeholder tekst, skærmbillede, Font/skrifttype

    Indhold genereret af kunstig intelligens kan være forkert.

Et billede, der indeholder tekst, Font/skrifttype, skærmbillede, linje/række

Indhold genereret af kunstig intelligens kan være forkert.

* Left algorithm 3
  + N2+n, but know n is just a constant, so
* Right algorithm 4.
  + n\*n\*(n-1) =

## Lecture 4

* What is the maximum subarray sum problem?
  + Finding the maximum sum of elements in a subarray, where the length of the array doesn’t matter.
* How can we transform products into sums? Fx 1.2\*4\*1 <= 5\*4\*9?
  + By taking the logarithm, because it is a growing function the relationship (<=) holds.
    - We know that log(a\*b)=log(a)+log(b)
  + Log(1.2)+log(4)+log(1) <= log(5)+log(4)+log(9)
* How can we transform the maximum subarray product, to maximum subarray sum?
  + By taking the logarithm to all values.
* What does kanes algorithm do?
  + For each subarray it takes max ending at the index, and compare it to the old max, and if the new max is biggest, it gets saved, and that is being compared next time.
* What is the divide and conque algorithm?
  + A recursive method, that divides the problem into subproblems.
  + Solve the small subproblems
  + Has a base case for the smallest subproblems
* What is a recursion tree?
  + A way to visualize global flow control of resuive relations.
  + Each call of the function is a new node, and branching down
  + Solving subproblems and return a solution is braching up
  + Et billede, der indeholder skitse, taburet, design, illustration/afbildning

    Indhold genereret af kunstig intelligens kan være forkert.
* What principle does the stack follow?
  + Last in first out, LIFO. Just like a stack of dirty plates.
* What is quicksort ?
  + A sortng algorithm, that works recursively. By selecting a pivot point (often the end n-1.
  + Comparing each element from the left to the right with the pivot point.
  + it's **swapped** to the "left side" of the subarray being partitioned.
  + if higher, stays, since the part between lower part and unsorted part is for the elements higher then the pivot point.
  + Move to the next unknown/sorted element in the array
  + When you reach the pivot point, insert it to the right of the lower part of the array.
  + Now do the same thing on the lower part of the array recursively.
  + After sorting the "lower part", Quicksort *also* recursively calls itself on the "bigger part" (the subarray to the right of the pivot)
* What is the runtime of quick sort for worst case?
  + O(n2), when we are unlucky with out pivot point.
    - The biggest or smallest value, so we partition n+n-1+n-2..2+1
  + This is for sorted array
* What is the best case run time of quick sort?
  + O(n log n), when we divides the array at the middle :⌈(n − 1)/2⌉ and ⌊(n − 1)/2⌋
* How many recursive calls do we have in quick sort?
  + Two:  
    Et billede, der indeholder tekst, Font/skrifttype, skærmbillede, algebra

    Indhold genereret af kunstig intelligens kan være forkert.
* How do we ensure that we often get the best run time for quick sort?
  + Choose the middle element instead of the end
* What is the most used sorting algorithm?
  + Quick sort
* What is the difference between mergesort and quick sort in memory regards?
  + Mergesort is not “in place”. Merge Sort typically uses a **temporary array** to store the merged result before copying it back to the original array.
  + Quick sort is in place, since it is just swapping elements, with no placeholders.  
    So the most of the memory is from the initial input.

## Lecture 5

* What is a binary tree?
  + A tree that branches out in at most two nodes, connecting by edges(lines).
* What is the depth of a binary tree?
  + Edges from the root:  
    Et billede, der indeholder diagram, hvid, cirkel, linje/række

    Indhold genereret af kunstig intelligens kan være forkert.
* What is the height of a binary tree?
  + The longest path (max number of edges) from the root down to a leaf:  
    Et billede, der indeholder hvid, diagram, cirkel, linje/række

    Indhold genereret af kunstig intelligens kan være forkert.
* What is The height of a node?
  + is the *maximum* number of *edges* on the longest path from that node *down to a leaf* node.
* What is a full complete binary tree?
  + A tree where all nodes are filled with leaves, so:  
    Et billede, der indeholder cirkel, hvid, skitse, linje/række

    Indhold genereret af kunstig intelligens kan være forkert.
  + NOT: Et billede, der indeholder skitse, hvid, cirkel, kunst

    Indhold genereret af kunstig intelligens kan være forkert.
* How can i calculate how many nodes a binary tree can have at maximum (full)?
  + Et billede, der indeholder tekst, Font/skrifttype, linje/række, hvid

    Indhold genereret af kunstig intelligens kan være forkert.
* What is a max-heap order?
  + When the root is bigger then or equal to the children individually:  
    Et billede, der indeholder cirkel, diagram, skærmbillede, hvid

    Indhold genereret af kunstig intelligens kan være forkert.
* What is a min-heap order?
  + Reverse of max-heap, so the root is <= to the children.
* When does a binary tree have I heap shape?
  + If all the nodes are full with leaves, except the last level, where all the leaves are as far to the left as possible  
    Et billede, der indeholder diagram, cirkel, skitse, hvid

    Indhold genereret af kunstig intelligens kan være forkert.
* What Is the maximum heigh of a binary tree with heap structure?
* How do place elements from a binary tree with heap structure, in an array?
  + Start from the root.
  + Than go down and LEFT, after this go right on the same level.
  + Than go down and start from the LEFT again. And go to the right on the same level.
  + repeat  
    Et billede, der indeholder skitse, tegning, diagram, illustration/afbildning

    Indhold genereret af kunstig intelligens kan være forkert.
* how can I calculate, where in the array a parent node is?
  + Idex = i
  + Parrent node : ⌊i/2⌋
  + Fx if we take the element at index 7, with the value 3:  
    Et billede, der indeholder tekst, Font/skrifttype, linje/række, hvid

    Indhold genereret af kunstig intelligens kan være forkert.
    - ⌊7/2⌋ = 3, the values at index 3 is 10. Which is correct:  
      Et billede, der indeholder cirkel, diagram, skitse, tegning

      Indhold genereret af kunstig intelligens kan være forkert.
* How can I find the index of children to a node in a array representation of the bineary tree with heap structure?
  + Et billede, der indeholder Font/skrifttype, Grafik, tekst, typografi

    Indhold genereret af kunstig intelligens kan være forkert.
  + Left and right children
* What is max-heapify?
  + An algorithm to sort the binary tree I a max heapify structure
  + If the node has a bigger child, swap place with the biggest of it’s children.
    - Continue this  
      Et billede, der indeholder Modetilbehør, Smykker

      Indhold genereret af kunstig intelligens kan være forkert.
* What is build heap?
  + Build-Heap efficiently converts an unordered array into a heap.
  + It uses a bottom-up approach, starting from the last parent nodes and moving up to the root.
  + For each parent node, it applies Max-Heapify to ensure the max-heap property is satisfied for the subtree rooted at that node.
  + The overall time complexity is linear, O(n), which is very efficient for heap construction.
* What do you need to know about an array to use counting sort?
  + The number of elements of the array,
* How does counting sort work?

1. We count the number of unique elements in the input array.  
   Et billede, der indeholder Font/skrifttype, nummer/tal, tekst, typografi

   Indhold genereret af kunstig intelligens kan være forkert.
2. Put the counts into an array, where the value of the input corresponds to the index of the count array.   
   Et billede, der indeholder Font/skrifttype, nummer/tal, tekst, typografi

   Indhold genereret af kunstig intelligens kan være forkert.  
   So there there is two 0’s in the input array A, and so on
3. Make the counting array into an accoumatetive array, where you sum up the elements before:  
   Et billede, der indeholder nummer/tal, Font/skrifttype, tekst

   Indhold genereret af kunstig intelligens kan være forkert.
4. Select the last element from the input array A, see what it’s count is in the accumulative array, that number is the index for the element in the new array:  
   Et billede, der indeholder skærmbillede, tekst, Font/skrifttype, nummer/tal

   Indhold genereret af kunstig intelligens kan være forkert.  
   3, has a count of 7 in the accumulative array, so input it at index 7.
5. Decrese the value in the accumulative array:
6. Et billede, der indeholder nummer/tal

   Indhold genereret af kunstig intelligens kan være forkert.  
   Here index 3 goes from a value of 7, to a value of 6.
7. Now reprease from the end of the array.

* What does it mean that an algorithm is stable (like counting sort)?
  + **Reverse Iteration (Processing from the end of the input array):**

When we iterate through the input array **backwards**, we are essentially processing elements in reverse of their original order. This is important when dealing with duplicates.

* + **Decrementing the Count Array:**

After placing an element in the output array, we **decrement** the count in the C array. This is the crucial step that ensures stability. The count array C is used to determine the correct position in the sorted output. By decrementing after each placement, we ensure that the *next* element with the same value will be placed at the *preceding* available position.

* What is radix sort?
  + An algorithm to sort longer numbers, where you start by sort from the right, and move to the left:  
    Et billede, der indeholder tekst, Font/skrifttype, hvid, typografi

    Indhold genereret af kunstig intelligens kan være forkert.
  + If the number is the same they maintain their original order (stable)
* When is radix sort good?
  + When we have numbers with a small base, fx binary with a base of 2. Or a base of 10
* What is the run time of radix sort?
  + O(d(n+k), where d= base, n = number of elements, k = base (here 10).
  + Simplifies to O(n)
* What is a priority que?
  + A data structure
* What is data structure?
  + Data with operations/algorithms. So algorithms that gets and saves data.
* What is the goal with operation?
  + Flexibility and efficiency (often they can not be both)
* For radix sort we sort it stable, so when two values are equal you look at the original array, and the ones that comes first, will go first in our new sorted array.
  + Fx insertion-sort, caus only insert if bigger
  + Merge-sort, always the left when merging if equal
    - Et billede, der indeholder håndskrift, tekst, tavle, Font/skrifttype

      Indhold genereret af kunstig intelligens kan være forkert.
  + Quick-sort NOT stable
  + Heap-sort NOT stable
    - Et billede, der indeholder håndskrift, Font/skrifttype, tavle, tekst

      Indhold genereret af kunstig intelligens kan være forkert.
* If we order it stable, when if we have two equal values, than the one that came first in the original array, comes first in the sorted array
* What operation do an unordered dictionary support?
  + Insert, delete, search
* What operation do a ordered distionary support?
  + The same as an unordered AND
    - Predecessor: gives the largest key that is less than my provided key
    - Successor: gives the smallest key that is bigger than my provided key
    - Ordered transversal: print the all elements in sorted order based on keys
* What is a binary search tree
  + Each node have at most 2 children
  + For each node v:
    - The left child (subnode) key is less then or equal to the parent node v key.
    - The right child (subnode) key is bigger than or equal to the parent node v.
* In a binary earch tree what is the inorder property?
  + Left node is less than or equal to the parent, and right node is bigger or equal.s
* What is the key difference from binary heap tree to binary search tree?
  + Et billede, der indeholder tekst, skærmbillede, Font/skrifttype, nummer/tal

    Indhold genereret af kunstig intelligens kan være forkert.
* Where do we save binary trees that have heap order?
  + In arrays
* What does a variable holds, that we save an object to?
  + The object memory address, where there is located memory for the object
* How do we get the keys from a binary search tree out in sorted order?
  + By using inorder transversal
  + Et billede, der indeholder tekst, Font/skrifttype, skærmbillede, algebra

    Indhold genereret af kunstig intelligens kan være forkert.
  + Run time is O(n), (and O(1) for each node)
* How do we find the tree minimum and maximum in a binary search tree?
  + Follow the outer left or outer right path:
  + Et billede, der indeholder tekst, skærmbillede, cirkel, hvid

    Indhold genereret af kunstig intelligens kan være forkert.
* What is the time for operations (that is NOT inorder transversal) in binary search trees?
  + O(n), simply the height of the tree.
    - From root to node or node to root.
* How many nodes can vi maximally have in a binary search tree?
  + Et billede, der indeholder diagram, hvid, linje/række, cirkel

    Indhold genereret af kunstig intelligens kan være forkert.
* When do we have the maximal amount of nodes in a tree?
  + When each node has 2 children.
* What is the best possible height of a binary search tree?
* What is the structural requirements for a red/black binary search tree?
  + Top and bottom (empty leaves) are black.
  + Equal amount of black on left and right side
  + No 2 red in row
* What datatype do we use for binary search tree?
  + Objects (node objects)
* What is the longest possible combination of red and black in binart search tree?
  + That they alters, red,black, red, black etc.
  + Shortest is black, black, black
* What is the height of a red/black tree?
  + 2 log n, at most
* How do we build an array from a binary search tree?
  + The root is the first element in the array, then the next element is the left child, and then the right child. Then the left most grand child etc:  
    A = |root, left child, right child, left child of the left child, right child of the left child, etc|