Test Questions for info 2

1. The complexity of an algorithm can be reduced by clever code optimization.
2. Complexity is a property of the algorithm, not of the program code.
3. Dynamic programming means trading time for space.
4. Dynamic programming means storing intermediate data for reuse.
5. Dynamic programming cannot reduce the complexity of an algorithm.
6. Logarithmic complexity is usually achieved by successive partition of the problem space.
7. Exponential complexity usually results from the enumeration of all potential solutions.
8. Full backtracking has exponential complexity.
9. Using iterators to enumerate potential solutions can cause exponential complexity.

10. Sorted trees have an overall search complexity of log n.
11. Sorted trees are binary trees by definition.
12. Only balanced tree provide a logarithmic search complexity.
13. The advantage of splay trees is that recently added information is found near the root.
14. Sorted Trees are linearized into sorted lists through postfix traversal.
15. The following operation, called right rotation, exchanges the root of a subtree without modifying its order.
16. The predecessor of a node in a sorted tree is the rightmost leaf of its left subtree.
17. Sorted trees allow efficient access, addition and removal of information, without moving blocks of data.
18. Sorted trees are space efficient, because their space demand grows and shrinks with the amout of data stored.
19. Inserting data from a sorted file into a sorted tree leads to frequent rebalancing.

20. Searching a linear list has a complexity of (n log n) at best.
21. Sorting a linear list can be done in (n log n) time.
22. Sorting a linked list cannot be done in (n log n) time, because there is no random access.

23. Searching a linear list naively has a complexity of n^2.

24. Sorting a linear list naively has a complexity of n^2.

25. Sorting a partially sorted list gives advantage to less efficient algorithms like insertion sort.

26. (n log n) sorting algorithms are based on the principle of successive partitions, called "divide and conquer".

27. Hash tables allow more efficient searching than sorted lists.

28. A good hash code yoeld an even distribution of the number of rehashes required over the problem space.

29. In a hash table implementation, external chaining requires more space than open addressing.

30. Hash maps can be used even if the keys are not ordered.

31. Hash maps can only be used for keys which are not ordered, like fingerprints.

32. An index table can be used to store an order of a list without re-ordering the list.

33. Index tables allow to store multiple orders of a list.

34. The semantics of a recursive definition is determined by rewriting.

35. In a recursive call, the control information is kept implicitly in the call stack.

36. Every recursive algorithm can be transformed into an interative one.

37. Recursive definitions must terminate.

38. A recursive method should make visible progress towards the base condition.

39. A recursive method needs a parameter which steadily decreases.

40. Recursion makes the algorithm implicit.

41. Recursion always means exponential complexity.

42. Tail recursion can easily be transformed into iteration.

43. List implementations as linked lists are good for space efficiency.
44. List implementations as linked lists allow easy insertion and deletion of information.
45. List implementations as linked lists are not optimal for sorting and searching of information.
46. List implementations as linked lists can easily be kept sorted.
47. List implementations as arrays are very good for sorting and searching.
48. List implementations as arrays require moving blocks of data for insertion and deletion.
49. List implementations asarrays should be doubled in size when grown to the limit.
50. List implementations as sorted trees are efficient for all operations.
51. List implementations as hash tables are efficient for adding and removing data.
52. List implementations as hash tables are hard to increase in size.

53. Graphs represent arbitrary structures of nodes and edges.
54. Graphs are usually stored as linked structures.
55. Graphs are usually stored as lists or matrices.
56. A disconnected graph cannot be cyclical.
57. A directed graph cannot be disconnected.
58. Nodes in graphs are usually encoded as numbers.
59. Each graph has a "natural" order of traversal along edges.

Older theory questions for inf2

1. The advantage of true random numbers is that they are reproducable.
2. Dynamic programming reduces the backtracking complexity by storing intermediate results for re-use.
3. Dynamic programming reduces the computation time by distributing the computation over several processors.
4. Static task distribution has a better load balnce than dynamic distribution.
5. The bag-of-task paradigm is a technique for dynamically distributing tasks over processes.
6. The shunting yard algorithm has linear complexity.
7. An RPN formula consists of a list of operands followed by a list of operators.

8. An RPN formula contains no brackets.
9. Balancing is only relevant on sorted trees.
10. The purpose of tree balancing is to reduce the time required for adding new entries.
11. The complexity of an iterative algorithm depends on the number of nested loops.
12. Logarithmic complexity is even better than linear complexity.
13. Logarithmic complexity is usually achieved by using the divide and conquer pattern.
14. JUnit testing is a good alternative to debugging.
15. A dpeth-first-search can be used to detct cycles in a graph.
16. Task scheduling can be handled using Prim's algorithm for minimal spannig trees.
17. A vertex list is always shoreter than an edge list.
18. The main advantage of Hash maps is their flexible length.
19. Trees are always connected graphs.
20. Floyd's algorithm is efficient, but does not always compute the optimum.