Algorithms and Data Structures
Exam Overview

Poznań University of Technology
Artificial Intelligence, 2nd semester

Transcription by Arthur Intel based on a private recording.
Some bits taken from the Google Doc by Garbarkozaurus.
Last update: June 17, 2021

General Information

Date/Time	Jun 30th 2021, 11AM
Duration	about 2 hours
No. Questions	5
Anti-cheat	Camera + Microphone
Minimum to Pass	13/25 points

Exam Requirements

1. Computability of the problem. Decision and optimization problems - dependencies.

Finite set of solutions. Decision (yes/no) vs. optimization (min/max of goal function).

decision problem is difficult	⇒	optimization is difficult.
decision problem is easy	⇒	optimization is easy.

2. Problem coding. Language and alphabet. Reasonable encoding rule.

- Use a reasonable encoding
- General rule: don't use unnecessary elements

3. The length of the coding string.

- Strictly depends on the complexity function.
- Depends on the number of elementary steps, which depends on the input size, e.g. the length

4. DTM and NDTM.

Deterministic and Nondeterministic Turing Machines.

> "This is not necessary to know, because i will send you these after the lecture"
> — It seems he will send us additional materials to study from.

UPDATE He sent us a chapter from Johnsons book, which is available here.

5. Computational complexity function.

Provide explanation on whether a function is polynomial or exponential.
If we know that a model is deterministic for a DTM, then it is also deterministic for other deterministic models.

6. Polynomial transformation.

- Demonstrating that a problem is P/NP by polynomially transforming a problem of the given class.

7. Class P and NP problems. Complexity class diagram of decision problems.

- Knowledge of the P/NP/NP-hard/NP-complete sets, their relations and the diagram.
- **NP-Complete**: no polynomial algorithms known
  - ⇒ **Ordinarly NPC**: can construct a pseudo-polynomial algorithm
  - ⇒ **Strongly NPC**: not even a pseudo-polynomial algorithm exists
- If a problem is NP-hard, and it is not "numerical" (no independent numerical parameters), then it is immediately **strongly** NP-hard (example: Hamiltonian Circuit Problem).

8. Formulation and representative examples of 5 NP-complete problems.

The "Big 5" problems:

- 3-Dimensional Matching
- Graph Vertex Cover
- Clique
- Hamiltonian Circuit
- Set Partition

Out of these 5, only Set Partition is (weakly) NP-Complete. The rest is strongly NP-Complete.

9. Knapsack Problem

- Formulation
- Examples of a problem
- Algorithms

10. NP-completeness proof — scheme

- How to, in general, prove the NP-Completeness of some problem?
  1. Construct an algorithm for NDTM.
  2. Check if a solution can be verified in polynomial time.
- Oracle machine

11. Problem $P_2 \,||\, C_{max}$

- How to assign jobs to 2 independent processors so that execution time is minimal?
- How to prove NP-Completeness of this problem?

12. Problem $1|r_j, d_j|$

Mr Pawlak gave a pretty vague explanation of this problem, and I was unable to find it online, but here is what I understood from the recording:

- Given a single processor and a set of tasks, each task having a completion time and a deadline, is it **feasible** to perform all tasks within their deadlines?
- How to prove that this problem is NP-hard? (transforms to Set Partition)

13. Sorting algorithms

Very precise analysis, you must know *exactly* how these algorithms are working.

a. Quick sort QS
b. Heap sort HS
c. Merge sort MS
d. Insertion sort IS
e. Selection sort SS
f. Bubble sort BS
g. Shell sort ShS
h. Counting sort CS

14. Dynamic data structures

- ordered lists
- queues
- trees

15. Compound data structures

Knowledge of these item operations on a list, tree and BST:

- adding
- deleting
- searching

16. Tree searching orders

- pre-order
- in-order
- post-order

17. Stack, binary tree, BST, BBST (balanced and AVL balanced tree)

Definitions.

18. Directed and undirected graphs. Graph connectivity.

Use of DFS and BFS methods.

19. Graph representations in a digital machine.

4 basic representations, their strengths and weaknesses:

- Adjacency Matrix
- Incidence Matrix
- Adjacency List
- Edge List

Example questions (*"Which representation is best for...?"*):

- ...checking if 2 vertices are connected?
- ...finding all vertices adjacent to some given one?

Provide time complexity of each representation for the given problem.

20. Searching a graph in depth and breadth (BFS and BFS).

Both for directed and undirected graphs.

21. Topological order - algorithms and examples.

Stick with the one from the book (p. 612).

22. Bi-connectivity graphs and algorithm.

- Articulation point
- Finding biconnected subgraphs (+complexity)

23. Euler cycle and path.

Definitions, algorithms for checking existence and their complexity.

24. Hamiltonian cycle and path.

Definitions, algorithms for checking existence and their complexity.

25. Backtracking algorithms

An example of how the algorithm works.

26. Minimum Spanning Tree

Definition, examples, algorithms and their complexities:

- Kruskal
- Prim

27. Dynamic programming algorithms example.

- Knapsack problem

28. Brute Force methods for solving problems, exhaustive search.

- When is it applicable to use a brute force algorithm?

29. Approximate and greedy-like algorithms.

- When are we justified to use a greedy algorithm?
- When does a greedy algorithm give the optimal solution? (matroids; connected to 26., since both Kruskal and Prim are greedy and optimal)

30. Branch and Bound method (B&B).

Basic implementation and description.

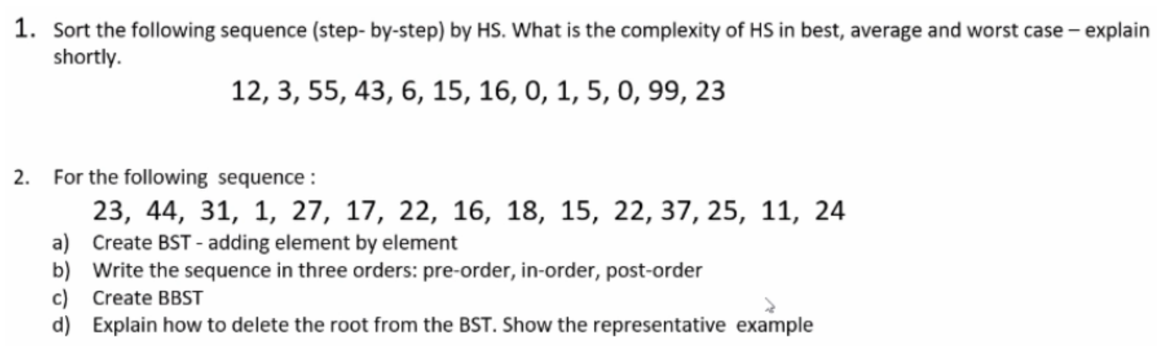
31. Traveling salesman problem.

Lower/upper bounds, examples how to solve the problem.

32. Methodology for solving open problems.

I. Try to find a polynomial-time optimal algorithm.
II. Prove the problem belongs to class P.
   1. If successful, that's it.
   2. If unsuccessful, try to find an NP-Complete problem that transforms into your problem, thus proving your problem is NP-Complete.
      a. If successful, you are justified to apply an approximate algorithm, as it's unlikely that a polynomial solution exists.
      b. If unsuccessful, the problem remains open.

Past Examination Sheets



1. Sort the following sequence (step- by-step) by HS. What is the complexity of HS in best, average and worst case – explain shortly.

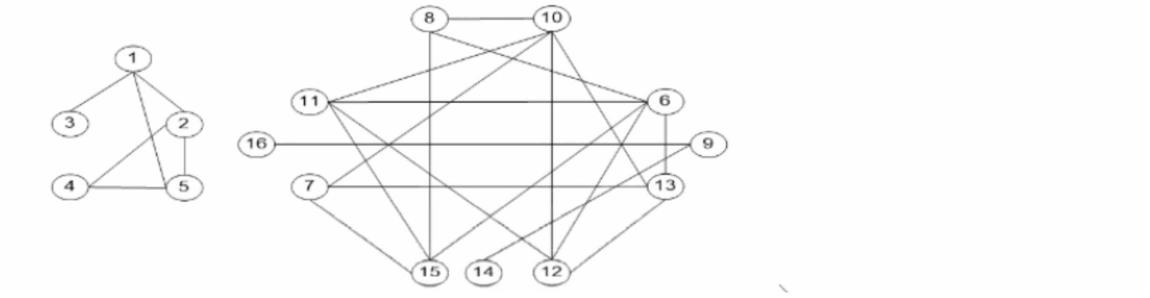
   12, 3, 55, 43, 6, 15, 16, 0, 1, 5, 0, 99, 23

2. For the following sequence :

   23, 44, 31, 1, 27, 17, 22, 16, 18, 15, 22, 37, 25, 11, 24

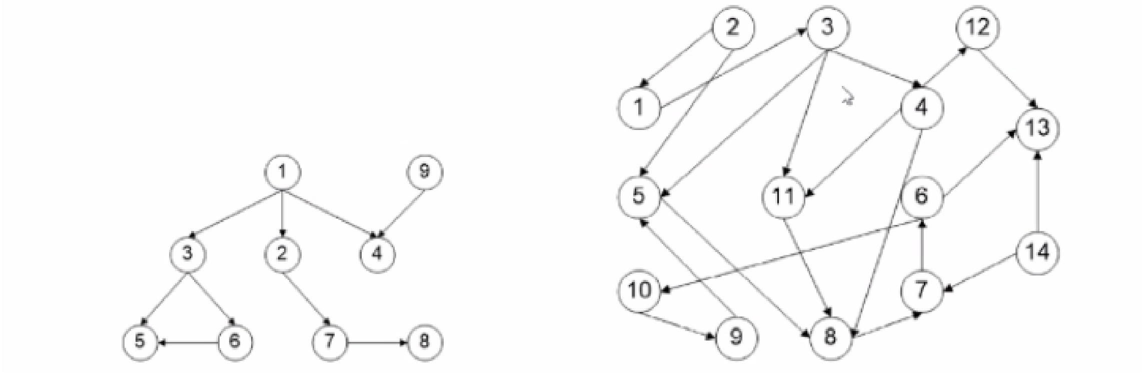
   a) Create BST - adding element by element
   b) Write the sequence in three orders: pre-order, in-order, post-order
   c) Create BBST
   d) Explain how to delete the root from the BST. Show the representative example

3. a) For the following graph writhe the sequence of vertices in DFS and BFS orders. Check the connectivity of the graph.



b) Write the topological order of the following graphs showing the steps of the algorithm you apply.

A. B.



4. Solve the Knapsack problem for the following example employing the dynamic programming algorithm method: knapsack capacity $b=7$, number of elements $n=6$, for i-th element the size s_i and the weight w_i shown in the table:

element i	1	2	3	4	5	6
s_i	1	3	3	1	2	14
w_i	3	2	6	2	14	44

a) Write the recursive function used in DP,
b) Write the chosen elements for the optimal solution
c) Write the optimal value for : b=5 and for only 4 elements? Explain shortly.
d) Write the complexity of DP? Explain shortly.

5. Write the schema of the NP-complete class of decision problems. Give an example of the problems belonging to each class.