Revision: definitions

2. In this question you are asked to define technical terms introduced in the lecture. Try to give exact and concise answers.

   (a) (2 marks)
   Give the definition of a tree.

"A tree does not contain any circle and each vertex is connected."

student answer

Incorrect technical term: "circle" (actual name: "cycle)

Nature of object missing: a tree is a graph.

"A tree is a graph that is connected and acyclic."
"A tree is a graph that is connected and does not contain cycles."

reference answer

Revision: definitions

2. In this question you are asked to define technical terms introduced in the lecture. Try to give exact and concise answers.

   (a) (2 marks)
   Define the notion of a combinatorial optimisation problem.

"A combinatorial optimisation problem is a problem where there are many combinations of different things (eg coins) where a selection of them must be taken to maximise or minimise some criteria (eg amount of coins)."

student answer

Characteristic missing: "feasibility constraint".

"A combinatorial optimisation problem is a computational problem consisting of a set of possible solutions, a feasibility constraint, and a cost/value function. The task is to find a feasible solution of optimal value/cost."

reference answer

Revision: definitions

When asked to define a concept:

1. Start with the nature/category of the concept you are defining: *"A tree is a graph…"*
2. List defining characteristics (only the essentials): *"…that is connected…"*
3. Use correct technical terms: *"…and acyclic."*

Define the following…

- Binary Tree
- Cycle (Graph)
- BST
- Min-Heap
- Max-Heap
- Clique
- Independent Set
- Minimum Spanning Tree
- Hamiltonian Cycle
- Travelling Salesman Problem
- n-Queen Problem
- Coin Denomination Problem
- Complexity Classes P and NP

- Other key definitions uploaded in week 13 Moodle

Revision: Computational complexities

Definition

The computational (time) complexity of an algorithm is the *number of elementary steps* $T(n)$ needed for computing its output for an input *of a size* n.

Usually we are interested in order of growth of complexity only (Big-O notation)

Can you perform complexity analysis and find the Big-O notations of simple algorithms?

Big-O Notation

- The following are in order of increasing time complexity:
  - Constant $O(1)$ (fixed time steps)
  - Logarithmic $O(\log N)$
  - Linear $O(N)$
  - Linearithmic $O(N \log N)$
  - Quadratic $O(N^2)$
  - Exponential $O(2^N)$
  - Factorial $O(N!)$

Big-O Notation

Can you explain the computational complexity / Find the best case and worst case for

- Selection Sort
- Insertion Sort
- Mergesort
- Heapsort
- Extracting min/max from heap
- Inserting a new element into a heap
- Inserting a new element into a BST
- Binary Search
- Linear Search

Revision: Sorting

[0,-3,2,4,3,5,1]

Write the states of the input list after each iteration of the main loop when applying the iterative version of:

- Merge sort
- Heap sort
- Insertion sort
- Selection sort

Invariants

A loop invariant is an assertion INV with:

- The loop initialisation will yield a state in which INV holds
- Every execution of loop body yields state in which INV holds again

(started in any state in which INV holds and exit condition does not hold)

We are interested in loop invariants that together with loop exit condition “turn into” desired post-condition.

Invariants

What are the loop invariants of the following algorithms?

- Selection Sort
- Insertion Sort
- Minimum Spanning Tree
- Binary Search
- Breadth First Search
- Dijkstra

Exam preparation checklist...

Review lecture materials and codes provided, re-implement workshop codes, and solve tutorial questions ...

- Definitions of key concepts
- Time complexities, invariants, and codes of sorting, searching, shortest path, binary tree algorithms
- Attempt practice exam (discuss solution/questions in forums)