MATEMATISKA INSTITUTIONEN
STOCKHOLMS UNIVERSITET
Avd. Beräkningsmatematik
Examinator: Marc Hellmuth

Tentamensskrivning i
Datastrukturer och algoritmer
7.5 hp
2024-05-23

General Information

- Time 8:00 - 12:00
- No books, internet, smartphones, notebooks or any extra material are allowed to be used.
- In total, you can get 64 points and you need to get at least 32 points to pass the exam.
- Many problems just require simple answer like 'yes' or 'no'; or $T(n) \in \Theta(f(n))$; or something similar. To answer these problems, **do not** waste your time by writing more than required to save time for answering other exercises!
- Do not provide ambiguous (or multiple) solutions. Do not change the problem statement to fit your solution.
- Make sure not to overlook any of the **16** problems.
- On average, you have 15 minutes for each of the 16 problems. However, many of them can be solved much faster. The level of difficulty or time needed to solve the respective problem is indicated using 1, 2, or 3 stars "$\star$":

  easy / quick = ($\star$)
  medium / moderate time = ($\star\star$)
  difficult / more time-consuming = ($\star\star\star$)

Problem 1 (*Basic Questions* ($\star$)) 1+1+1=3p

Answer the following questions.

(a) When is an algorithm said to be correct with respect to a given computational problem?

An algorithm is said to be correct if, for every input instance, it halts (=terminates) with the correct output w.r.t. the given computational problem. (minors 0.5p) 1p

(b) How is the worst-case time complexity of an algorithm defined?

The worst-case time complexity of an algorithm is the function defined by the maximum number of steps/instructions taken in any instance I of size $|I|$. (minors 0.5p) 1p

(c) How many bits are 8 byte?

8 byte $= 8 * 8$ bits $= 64$ bits (minors 0.5p) 1p

Problem 2 (*Algorithm Design* ($\star\star\star$)) 4p

Provide pseudocode for an algorithm that takes as input a binary number B of length n and has as output the translation of B as an integer (base_10 number). You can assume that B is given as an array $B[0..n-1]$, where $B[i]$ is the i-th entry of B that corresponds to the bit encoding 2^i. Moreover, you can assume that the length n of B is part of the input. You are allowed to use any type of loops, recursions, and variable assignments as well as the return or print operations. However, for basic arithmetic operation use only multiplication and addition.

Exemplify how your algorithm works step-by-step by using the array $B = [1, 1, 0, 1, 0]$ representing the binary number

$B[4]$	$B[3]$	$B[2]$	$B[1]$	$B[0]$
0	1	0	1	1

Provide the value of each of your used variables in each step of your executed algorithm. A correctness proof is *not* required.

Solution:

```
function binaryToDecimal(B,n):
   decimal = 0
   power = 1
   for i from 0 to n - 1:
      decimal = decimal + B[i] * power
      power = power * 2
   return decimal
```

For $B = [1, 1, 0, 1, 0]$:

iteration	decimal	power
Initially	decimal = 0	power = 1.
$i = 0$	decimal = 0 + 1 * 1 = 1	power = 1 * 2 = 2.
$i = 1$	decimal = 1 + 1 * 2 = 3	power = 2 * 2 = 4.
$i = 2$	decimal = 3 + 0 * 4 = 3	power = 4 * 2 = 8.
$i = 3$	decimal = 3 + 1 * 8 = 11	power = 8 * 2 = 16.
$i = 4$	decimal = 11 + 0 * 16 = 11	power = 16 * 2 = 32.

Loop ends because we've gone through all bits.
Return 11

POINTS:
correct pseudocode: 2p. minor mistake 1p. else 0p
exemplify on $B = [1, 1, 0, 1, 0]$: 2 (correct), 1 (minor mistake), 0 (else).

Problem 3 (*O-, Ω- and Θ-Notation* (⋆⋆)) 1+2+6=9p

(a) Define the set $O(g(n))$.

(b) Prove in detail that $T(n) = (n+2)^3 \in \Theta(n^3)$.

In detail means that arguments "as shown in the lecture" are not sufficient and that a rigorous mathematical proof is required.

(c) For two given functions f and g, indicate whether $f \in O(g)$, whether $f \in \Omega(g)$, and whether $f \in \Theta(g)$.

Here *yes/no* answers are sufficient. Provide your solutions in a table as indicated right-below and where the entries are filled with either *yes* or *no*.

Nr	$f(n)$	$g(n)$
(1)	$\frac{1}{3}10^n$	$3n^2 2^n$
(2)	$5n + \sin^2(3n^4) + 2n^2$	$(\frac{3}{n^3} + n)^2$
(3)	$\frac{\frac{1}{2}n + \frac{1}{6}n^6 - 8 + 5n^3}{2n^3+10}$	$5\sqrt{n^8+1}+1$
(4)	$n \log_3(n^5)$	$n \log_{10}(\frac{n}{3} + 2)$

Nr	$f \in O(g)$	$f \in \Omega(g)$	$f \in \Theta(g)$
(1)	no	yes	no
(2)	yes	yes	yes
(3)	yes	no	no
(4)	yes	yes	yes

Solution:

(a) $O(g(n)) := \{f(n)$: there are positive constants c and n_0 such that $0 \le f(n) \le cg(n)$ for all $n \ge n_0\}$

(b) $T(n) \in O(n^3)$:

$T(n) = (n+2)^3 = n^3 + 6n^2 + 12n + 8 \le n^3 + 6n^3 + 12n^3 + 8n^3 = 27n^3$

$\Rightarrow$ for $c = 27$ and $n_0 \ge 1$ we have $T(n) \le cn^3$ for all $n \ge n_0$

$\Rightarrow T(n) \in O(n^3)$

$T(n) \in \Omega(n^3)$:

$T(n) = (n+2)^3 = n^3 + 6n^2 + 12n + 8 > n^3$

$\Rightarrow$ for $c = 1$ and $n_0 \ge 1$ we have $T(n) \ge cn^3$ for all $n \ge n_0$

$\Rightarrow T(n) \in \Omega(n^3)$

$T(n) \in O(n^3)$ and $T(n) \in \Omega(n^3) \implies T(n) \in \Theta(n^2)$

(c) see tabular above.

POINTS:

(a) correct 1p. else 0p

(b) Correct $T(n) \in O(n^3)$ 0.75p (minor 0.25p else 0p)

Correct $T(n) \in \Omega(n^3)$ 0.75p (minor 0.25p else 0p)

Argument: for $T(n) \in \Theta(n^2)$ we must show that $T(n) \in O(n^3)$ and $T(n) \in \Omega(n^3)$ 0.5p

X = summed up point. Then total points: $\lceil X \rceil$

(c) per correct answer 0.5p = 6p.

Problem 4 (*Time and Space Complexity* (⋆⋆)) 1.5+1.5=3p

Provide the exact bounds for the time complexity $T(n)$ and space complexity $S(n)$ of the following program `policeThief(...)` in Θ-notation, assuming a unit-cost model.

Here is some more information about used commands.

- The the command `length` returns the length of a given linked list.
- The input size is determined by $n := \texttt{length}(L)$ of the input list L.
- The function `append` adds the given element to the end of the list, thus extending it by 1 element.
- $list[i]$ returns the key of the element corresponding to the i-th entry of the list $list$

Remark: You do **not** need to understand what the program computes in order to answer this question.

```
policeThief(list L, int n, int k)
 1  int res := 0, i := 0
 2  init empty list pol
 3  init empty list thi
 4  WHILE (i < n) DO
 5      i := i + 1
 6      IF (L[i] = P) THEN pol.append(i)
 7      ELSE IF (L[i] = T) THEN thi.append(i)
 8  int l := 0, r := 0
 9  WHILE (l < length(thi) and r < length(pol)) DO
10      IF (|thi[l] − pol[r]| ≤ k) THEN
11          res := res + 1
12          l := l + 1
13          r := r + 1
14      ELSE IF (thi[l] < pol[r]) THEN l := l + 1
15      ELSE r := r + 1
16  return res
```

Answers of the form $T(n) \in \Theta(..)$ and $S(n) \in \Theta(..)$ are sufficient and proofs are not needed. In other words, you only need to answer:

(a) Time complexity: Provide $f(n)$ such that $T(n) \in \Theta(f(n))$.

(b) Space complexity: Provide $g(n)$ such that $S(n) \in \Theta(g(n))$.

Solution:

Time complexity: $T(n) \in \Theta(n)$
Space complexity: $S(n) \in \Theta(n)$

[For us: this function simulates a scenario where police officers and thieves are positioned at different indices in an array, and it counts the number of thieves caught by police officers within a maximum distance k.]

POINTS:
correct 1.5p. else 0p (maybe 0.5p if some explanations are given that still lead to some wrong result)

Problem 5 (*Master Theorem* ($\star\star\star$)) 1+2+1+4 = 8p

Recall, the Master Theorem specifies the runtime $T(n) \in \Theta(..)$ for certain algorithms and is based on the input size n and

a = number of subproblems in the recursion

n/b = size of a single subproblem

d for the overhead $g(n) \in \Theta(n^d)$

(a) Provide the values a, b and d for the algorithm `BinarySearch` for searching an element in an array of length n as provided in the lecture.

(b) What is the solution $T(n) \in \Theta(..)$ according to the Master theorem for **general values** a, b and d that satisfy $a = b^d$.

Which bound $T_{\texttt{BinarySearch}}(n) \in \Theta(..)$ does the latter result imply for `BinarySearch`?

(c) What complexity $T(n) \in \Theta(..)$ would result from the Master theorem for the recurrence equation

$$T(n) = 8T(n/2) + \Theta(n^4)?$$

(d) Given is the following pseudocode of a divide-and-conquer approach to compute $base^{exponent}$.

```
power(int base, int exponent)
 1  IF (exponent = 0)
 2     print(exponent)
 3     return 1
 4  ELSE IF (exponent = 1)
 5     print(exponent)
 6     return base
 7  half_exponent := ⌊exponent / 2⌋
 8  result := power(base, half_exponent)
 9  IF (exponent mod 2 = 0)
10     new_result := result · result
10     print(new_result, exponent)
11     return new_result
12  ELSE
13     new_result := result · result · base
13     print(new_result, exponent)
14     return new_result
```

(i) Call `power`(2,9) and provide the output of the `print` command in each of the single recursive calls in the order they appear.

(ii) Use the Master Theorem to determine the runtime $T(n)$ of `power` that takes as input an arbitrary integer $base$ and $n := exponent$, assuming a unit-cost model. In particular, specify a, b, and d as well as the function $f(n)$ such that $T(n) \in \Theta(f(n))$.

Solution:

(a) $a = 1, b = 2, d = 0$ 1p

(b) general: $T(n) \in \Theta(n^d log_2(n))$

`BinarySearch`: $T(n) \in \Theta(n^0 log_2(n)) = \Theta(log_2(n))$ 1+1 =2p

(c) $\Theta(n^4)$ (Remark: $a = 8$, $b = 2$, $d = 4$) 1p

(d) (i) 1
4 2
16 4
512 9 each correct print 0.5p = 2p

(ii) $a = 1, b = 2, d = 0$ 1p

$\implies 1 = 2^0 \implies T(n) \in \Theta(n^0 log_2(n)) = \Theta(log_2(n))$ 1p

total or (ii) = 2p

total for (d) = 4p

Problem 6 (*Heap Sort* ($\star\star$)) 1+3+2 = 6p

Given is the array $A = [2, 3, 1, 5, 4]$

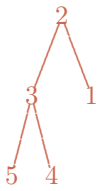
(a) Draw the array A as a binary heap tree.

(b) Use `Build-Max-Heap(`A`)` according to the lecture to transform A into a max-heap. For each index i for which the max-heap property is violated, specify the key $A[i]$ and also the key $A[largest]$ with which $A[i]$ changes its position in the heap. Draw the heap A after each such "exchange" as a binary tree.

(c) Starting from the max-heap produced in (b), use `Heapsort`(A) and consider the *first* iteration of the `Heapsort` algorithm (i.e., the iteration that brings the largest element to its correct position in A). Provide a drawing of the max-heap after this iteration.

Solution:

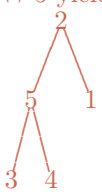
(a)



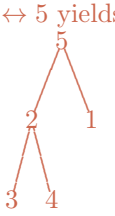
1p

(b)

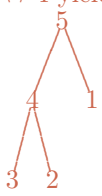
$3 \leftrightarrow 5$ yields



$2 \leftrightarrow 5$ yields

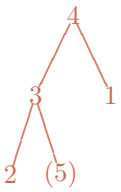


$2 \leftrightarrow 4$ yields



0.5p for correct $A(i) \leftrightarrow A[largest]$ + 0.5p for correct tree 3p

(c)



(OK! if leaf (5) not drawn) 2p

Problem 7 (*Insertion Sort* ($\star$)) 2p

Given is the array $A = [2, 1, 3, 5, 4]$. Sort A using `Insertion_Sort` in ascending order as provided in the lecture. Provide the array after *all* steps in which the array differs from the array in the previous step.
Hint: there are more than two arrays to provide.

Solution:

2	2	3	5	4
1	2	3	5	4
1	2	3	5	5
1	2	3	4	5

POINTS: for each correct line 0.5 point

Problem 8 (*Searching* (⋆⋆)) 2.5+3.5 = 6p

Given is the following sequence of numbers stored in an array L, sorted in ascending order:

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$L[i]$	19	31	44	52	91	108	111	119	127	130	150	175	196	201	202	203

i	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
$L[i]$	207	211	215	218	221	227	231	235	239	243	250	252	267	269	277	278

(a) Use `binary_search` as provided in the lecture and provide the search steps for searching for element 150. Specify for each step, in which part of the array you are searching and shortly state how this decision has been derived in each step.

(b) Use `Exponential_Search` as provided in the lecture and provide the search steps for searching for element 150. Specify for each step, in which part of the array you are searching and shortly state how this decision has been derived in each step.

Solution:

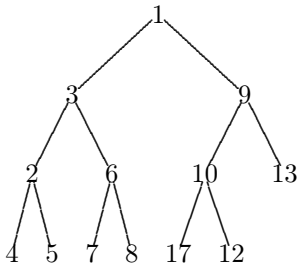
(a)
- Search in 1 .. 32. $mid = \lfloor (1+32)/2 \rfloor = 16$; $L[16] = 203 > 150$
- Search in 1 .. 15. $mid = \lfloor (1+15)/2 \rfloor = 8$; $L[8] = 119 < 150$
- Search in 9 .. 15. $mid = \lfloor (9+15)/2 \rfloor = 12$; $L[12] = 175 > 150$
- Search in 9 .. 11. $mid = \lfloor (9+11)/2 \rfloor = 10$; $L[10] = 130 < 150$
- Search in 11 .. 11. $mid = \lfloor (11+11)/2 \rfloor = 11$: $L[11] = 150 = 150$ (found)

(b)
- Check $L[1] = 19 < 150$.
- Search in 2 .. 32; $L[2] = 31 < 150$
- Search in 4 .. 32; $L[4] = 52 < 150$
- Search in 8 .. 32; $L[8] = 119 < 150$
- Search in 16 .. 32; $L[16] = 203 > 150$
- Linear Search in 9 .. 16;
- L[9]=127
  L[10]=130
  L[11]=150 (found)

POINTS: for each correct line 0.5 point

Problem 9 (*Properties of Trees* ($\star$)) 3+1 = 4p

(a) Given is the following rooted tree T.



Specify:

- the height of T 3
- the depth of vertex 9 1

Indicate (yes/no) whether T has the following properties:

- binary search tree no
- fully-binary yes
- nearly-complete yes
- complete no

(b) Sketch a rooted tree with $n > 1$ vertices with a minimum number of leaves.

Solution:

(a) see answers above.

(b) Path $1 - 2 - \ldots - n$ ($1 = root$)

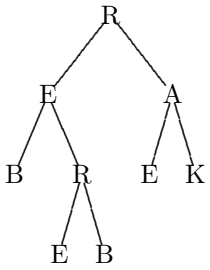
POINTS:

(a) each correct answer: 0.5p

(b) 1p (an example for specific n only, e.g. $n = 5$ = -0.5p)

Problem 10 (*Traversing Trees* ($\star$)) 1+1=2p

Given is the following rooted tree T.



Write the keys in the order they are visited in T using

(a) in-order B E E R B R E A K

(b) post-order B E B R E E K A R

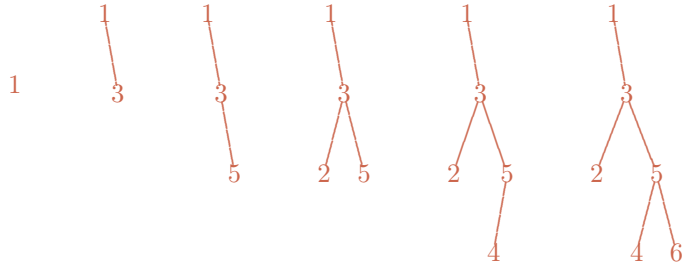
Solution:

POINTS: each correct (a) / (b) = 1p // total: 2p

Problem 11 (*Binary Search Trees* ($\star$)) 3p

For the given sequence of integers $1, 3, 5, 2, 4, 6$ build the binary search tree, that is, insert the integers one after another as keys into an initially empty binary tree and draw the tree after each insertion.

Solution:



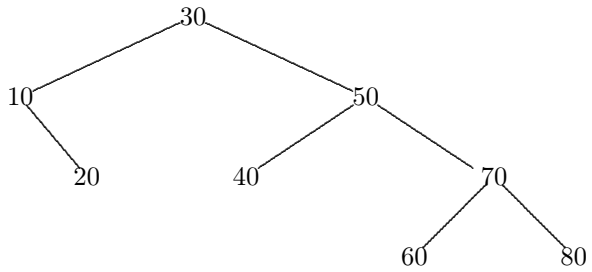
POINTS: 0.5p for each correct tree = 3p

Problem 12 (*AVL tree* (⋆⋆)) 2+2 = 4p

For the given AVL tree, perform the operation as specified in (a) and (b).

If necessary, perform the deletion operation in the right subtree.
For required rotations, specify whether it is a left rotation (LR) or right rotation (RR), and specify the vertex where the rotation takes place. Represent double rotations as two single rotations.

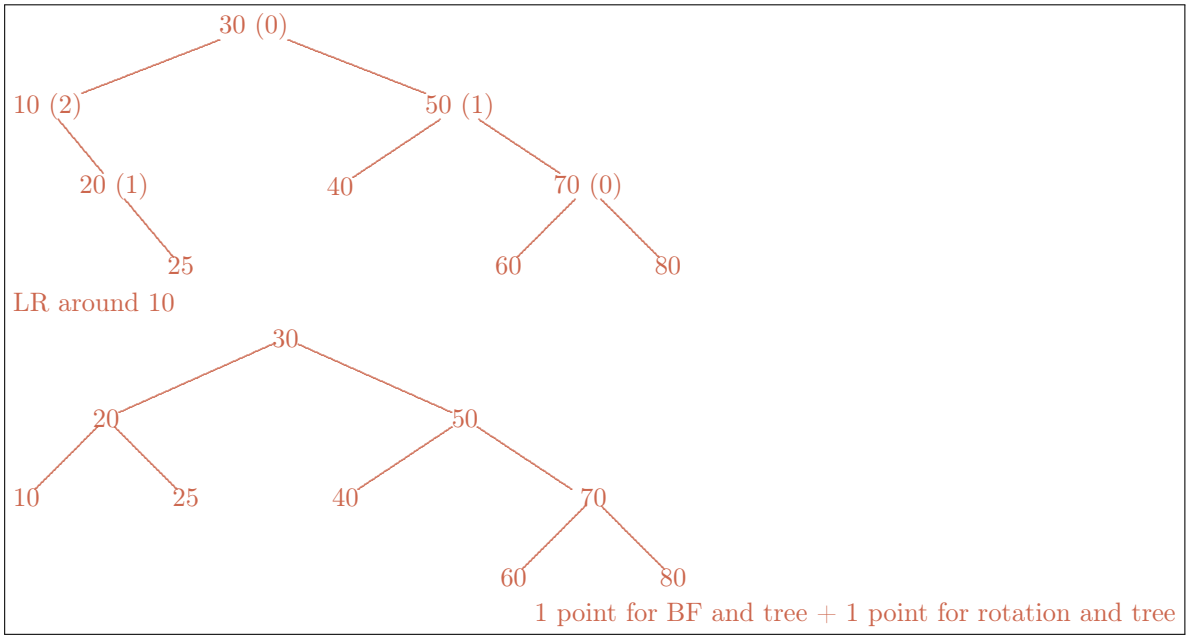
The AVL tree T that is used as starting tree for each of the following subtasks is as follows:



The tree T that serves as initial tree for the tasks in (a) and (b).

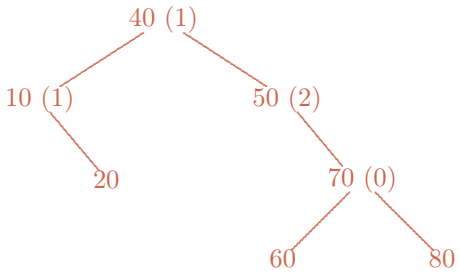
(a) Insert 25 into the tree T and draw the resulting tree "$T + 25$" including the balance factors next to each inner vertex. If "$T + 25$" is not an AVL tree, rebalance it. Specify the rotation(s) for rebalancing and draw the resulting tree after rebalancing it.

Solution:

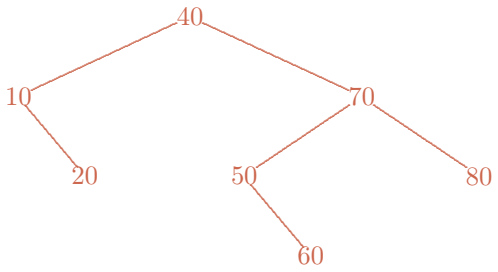


(b) Delete 30 from the tree T and draw the resulting tree "$T - 30$" including the balance factors next to each inner vertex. If "$T - 30$" is not an AVL tree, rebalance it. Specify the rotation(s) for rebalancing and draw the resulting tree after rebalancing it.

Solution:



LR around 50



1 point for BF and tree + 1 point for rotation and tree

Problem 13 (*Hashing* ($\star$)) 2.5p

Given is a hash table H of size $m = 7$. Access to H is done using Linear Probing.
The hash functions for keys k is here defined as follows:

$$h(k, i) = (k + 2 \cdot i) \bmod m \qquad (i = 0, 1, 2, \ldots) \ .$$

Insert the keys `16,26,5,3,7` in this order into the initially empty hash table H; indicate the table and the respective value of i after each single insertion of a key into the hash table H.

Solution:

0	1	2	3	4	5	6	i
-	-	16	-	-	-	-	0
-	-	16	-	-	26	-	0
5	-	16	-	-	26	-	1
5	-	16	3	-	26	-	0
5	-	16	3	7	26	-	2

POINTS: for each correct line (H and i): 0.5p // total 2.5p

Problem 14 (*Bloom Filter* ($\star$)) 2p

Given is the following bloom filter $(B, \mathcal{H})$ where the array $B = [0..8]$ consists of $m = 9$ bits, all initially set to 0 and $\mathcal{H} = \{h_1, h_2, h_3\}$ is the set of the hash functions

$$h_i(k) = (k + i) \bmod m; \text{ where } i \in \{1, 2, 3\}.$$

Based on this, provide the bloom filter that represents the set $S = \{11\}$ and the set $S = \{11, 19\}$.

Solution:

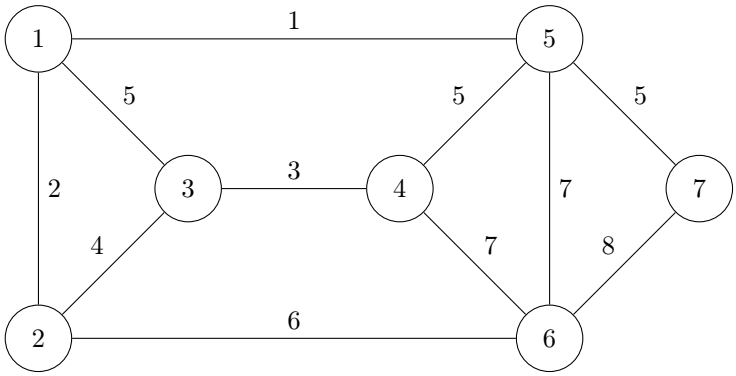
For $S = \{11\}$:
$B = [\ 0\ \ 0\ \ 0\ \ 1\ \ 1\ \ 1\ \ 0\ \ 0\ \ 0\ ]$

For $S = \{11, 19\}$:
$B = [\ 0\ \ 0\ \ 1\ \ 1\ \ 1\ \ 1\ \ 0\ \ 0\ \ 0\ ]$

POINTS: each correct B = 1p // total 2p

Problem 15 (*Elementary Graph Algorithms* ($\star$)) 3p

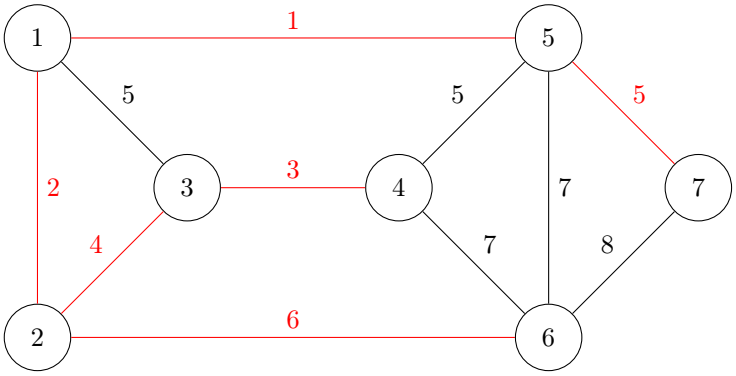
Given is the following undirected weighted graph G.



Find a minimum spanning tree of G using `Kruskal`'s algorithm. Provide the edges in the order in which they are inserted into the spanning tree. Edges that are **not** included in the spanning tree do not need to be specified.

Solution:

Spanning tree in the order of insertion:
$\{1,5\}, \{1,2\}, \{3,4\}, \{2,3\}, \{5,7\}, \{2,6\}$



POINTS: 0.5 point for each edge (incl order)

Problem 16 (*A final puzzle* ($\star\star\star$)) 2.5p

The puzzle keeper stands as the guardian at the border, ready to challenge any who seek passage to the land of happiness and completed exams. The puzzle keeper speaks to you:

"I have lined up 5 *coins* $C_1, C_2, \ldots, C_5$*. Each coin has two sides: a head (*`H`*) or a tail (*`T`*). Up to some number* $i \in \{1, \ldots, 4\}$*, all coins* $C_1, \ldots, C_i$ *show head, while all coins* $C_{i+1} \ldots C_5$ *show tail. You cannot see the coins, but you can ask me two questions that can be answered with either 'yes' or 'no'. If you are able to tell me the exact number of coins showing tail after asking at most two questions, you can pass."*

You desperately want to finish the exam. How can you find the correct number of coins showing tail using at most two questions? Note that, by assumption, at least one coin shows head and at least one coin shows tail. Listed are all possible coin arrangements:

C_1	C_2	C_3	C_4	C_5
H	H	H	H	T
H	H	H	T	T
H	H	T	T	T
H	T	T	T	T

Explain in detail the first question you are going to ask and what the resulting question will be based on the answer to your first question. Explain how you determine the number of coins showing tail after you receive the answers to your questions.

Solution:

Several ways to solve it:
One way is binary-search:

Q1: Is $C_3 = T$?
If Q1 yes:

- Q2: Is $C_2 = T$?
  If Q2 yes: 4 coins show tail (since $C_1 = H$)
  If Q2 no: 3 coins show tail

If Q1 no:

- Q2: Is $C_4 = T$?
  If Q2 yes: 2 coins show tail
  If Q2 no: 1 coins show tail (since $C_5 = T$)

***
Alternative way:
Q1: Are there more than 2 coins showing tail.
If Q1 yes:

- Q2: Are there 3 coins showing tail.
  If Q2 yes: 3 coins show tail
  If Q2 no: 4 coins show tail (since $C_1 = H$)

If Q1 no:

- Q2: Are there 2 coins showing tail.
  If Q2 yes: 2 coins show tail
  If Q2 no: 1 coins show tail (since $C_5 = T$)

POINTS: 1 point for correct Q1 and 1.5p for Q2+explanation. (1p for good attemps, 0p else)