Exam, DAT038/TDA417 (with solutions)

Data structures and algorithms

Sample exam 2, 2019-12-18

There will be **6 basic questions**, and **3 advanced questions**, and **two points per question**. So, the highest possible mark is 18 in total, of which 12 on the basic questions and 6 on the advanced questions. Here is what you need to do to get each grade:

- To pass the course, you must get 8 out of 12 on the basic questions.
- To get a four, you must get 9 out of 12 on the basic questions, and also get 2 out of 6 on the advanced questions.
- To get a five, you must get 10 out of 12 on the basic questions, and also get 4 out of 6 on the advanced questions.

Grade	Total points	Basic points	Advanced
3	≥ 8	≥ 8	—
4	≥ 11	≥ 9	≥ 2
5	≥ 14	≥ 10	≥ 4

Good luck!

Write your anonymous code (*not* your name) here: ___________________

Basic question 1 (complexity)

[complexity quiz, 2019-11-19]

1A) Arrange the following functions according to their *order of growth*

A) $34x + x\log_2 x$	$\Theta(x \log x)$
B) $1x + 2x + 3x + 4x + 5x + 6x + 7x$	$\Theta(x)$
C) $104000x + 0.005x^2 + 103/x^4$	$\Theta(x^2)$
D) $10\log_2 x + 2\log_{10} x$	$\Theta(\log x)$
E) $2^x + x^2$	$\Theta(2^x)$
F) $3(x-2)(x-1)x$	$\Theta(x^3)$

Write A–F in the cells below:

smallest growth rate *largest growth rate*

D	B	A	C	F	E

1B) What is the order of growth of the following code?

```
int result = 0;
for (int i = 1; i <= n; i *= 2) {
    for (int j = 0; j <= n; j++) {
        result++;
    }
}
```

Order of growth (in terms of *n*): ***n* log *n***

(because the outer loop has logarithmic order of growth, and the inner is linear)

Write your anonymous code (*not* your name) here: ___________________

Basic question 2 (quicksort)

[DIT960, 2012-05-29, q6]

Perform a quicksort partitioning of the following array:

0	1	2	3	4	5	6	7	8
53	88	32	44	23	90	67	71	42

Show how the partitioned array looks like with the following choices of pivot. Remember to swap the pivot with the first element before partitioning.

Also note which sub-arrays that need further sorting in a recursive call. (E.g., by drawing curly braces under each sub-array, or by crossing over the cells that do not need sorting).

2A) Pivot = the first element

0	1	2	3	4	5	6	7	8
23	42	32	44	~~53~~	90	67	71	88
	sub-array			pivot		sub-array		

2B) Pivot = median-of-three

0	1	2	3	4	5	6	7	8
32	23	~~42~~	44	88	90	67	71	53
sub-array		pivot			sub-array			

Write your anonymous code (*not* your name) here: ____________________

Basic question 3 (queues)

[DIT960, 2012-05-29, q3]

We have a class Queue that is implemented as a circular array. Internally it has two integer variables `front` and `rear`, which points to the front and the rear of the array (which is called `data`). It has two public methods, `offer` and `poll`:

```
public boolean offer(E item);
public E poll();
```

If we create a queue q, add the four elements a, b, c, and d, and then remove one, the internal representation looks like this:

front = 1   rear = 3

	0	1	2	3	4
data =	a	b	c	d	

Now assume that we execute the following sequence:

```
q.offer("e"); q.offer("f"); q.poll(); q.poll(); q.offer("g"); q.poll();
```

3A) How does the internal state of q look like after this?

front = 4   rear = 1

	0	1	2	3	4
data =	f	g	c	d	e

Note: it's ok if you don't include `c` and `d` (because they're not in the queue), but also if you include them (because they're in the array).

3B) Resize the queue to size 8. How will the internal state look like?

front = 0   rear = 2

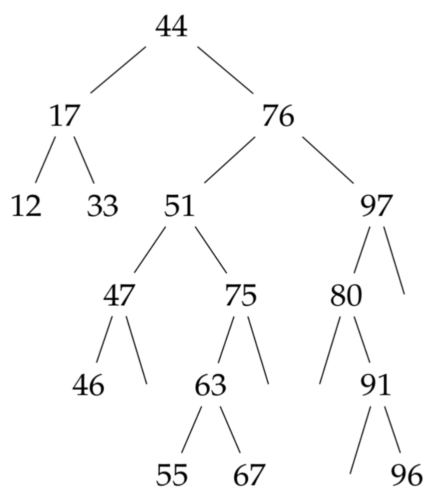
	0	1	2	3	4	5	6	7
data =	e	f	g					

Write your anonymous code (*not* your name) here: ___________________

Basic question 4 (search trees)

[DIT961, 2018-10-12, q2]

Assume the binary search tree to the right (note that it's not a balanced tree).



4A) In which order could the elements 17, 33, 47, 63, 67, 76 have been added to the tree?

Draw a circle around the letters A–F that denote a possible order. (Hint: there are more than one and less than five).
Assume that no elements have been deleted yet.

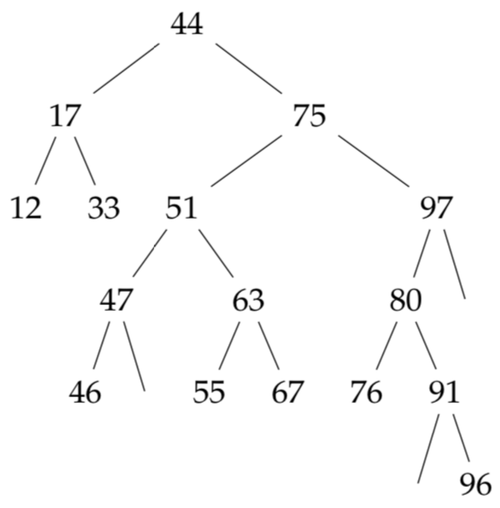
A) 17, 33, 47, 63, 67, 76
B) 17, 76, 33, 63, 47, 67
C) 17, 76, 47, 33, 67, 63
D) 76, 63, 67, 47, 17, 33
E) 76, 67, 17, 33, 47, 63

The answers are B and D.

The idea is: if node x is an ancestor of node y, then node x must have been added before y. In A, 67 is added before 76, but 76 is an ancestor of 67. In C and E, 67 is added before 63, but 63 is an ancestor of 67.

4B) Remove 76 from the tree, then insert it again. How does the tree look like now?

See the tree to the right (assuming you moved the greatest element from the left subtree (75) to the position where 76 was before).

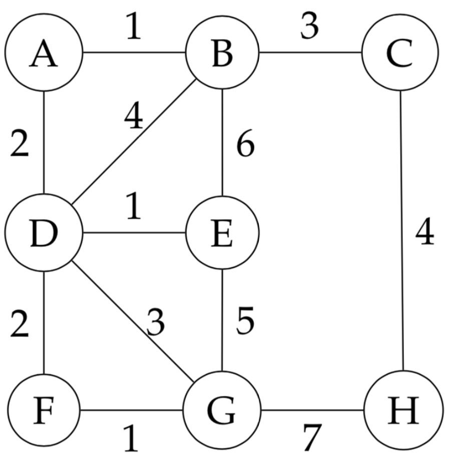


Write your anonymous code (*not* your name) here: ________________

Basic question 5 (graphs)

[DIT961, 2019-10-11, q6]

You are given the undirected weighted graph to the right.



5A) Compute a minimal spanning tree for the graph by manually performing Prim's algorithm using H as starting node.

Your answer should be the set of edges which are members of the spanning tree you have computed. The edges should be listed in the order they are added as Prim's algorithm is executed. As a hint, the first added edge is already printed. Refer to each edge by the labels of the two nodes that it connects.

	first added						*last added*
edge	HC	CB	BA	AD	DE	DF	FG

5B) Perform Dijkstra's algorithm starting from node A.

In which order does the algorithm visit the nodes, and what is the computed distance to each of them?

	first visited							*last visited*
node	A	B	D	E	F	C	G	H
distance	0	1	2	3	4	4	5	8

Write your anonymous code (*not* your name) here: ____________________

Basic question 6 (tries, counting)

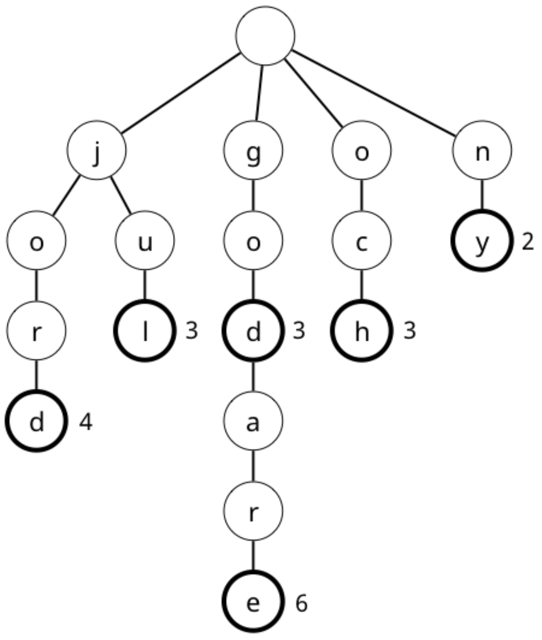
Consider the following table of words and integers (the numbers are the length of the words, but that's not relevant for this exercise):

Word	Length
`god`	3
`jord`	4
`och`	3
`godare`	6
`ny`	2
`jul`	3

6A) Draw the (r-way) trie that results when the words are inserted into an initially empty trie.

Insert them in the top-down order of the table, and use the numbers as values. Do not draw null links, and draw values right next to the corresponding trie node.

See the trie to the right



Write your anonymous code (*not* your name) here: ___________________

6B) Perform key-indexed counting, using the lengths as sorting key.

Recall that key-indexed counting uses auxiliary arrays: one for counting the frequencies, one for calculating starting indices, and one for the resulting sorted array. Fill in the resulting values of these two arrays:

	original	frequencies	indices	result
0	god	0	0	ny
1	jord	0	0	god
2	och	0	0	och
3	godare	1	1	jul
4	ny	3	4	jord
5	jul	1	5	godare
6		0	5	
7		1	6	

Note that the frequencies and indices are offset by one (because that's how key-indexed counting works), but it's ok if you forgot to do that.

(Side note: usually the same array is used to first store the frequencies and then convert them to indices, but you can treat them as different arrays here).

Write your anonymous code (*not* your name) here: ___________________

Advanced question 7 (binary search)

[DIT960, 2013-08-27, q8]

Consider the following implementation of binary search.

```
 1   int search(Object[] array, Object key) {
 2       int low = 0;
 3       int high = array.length - 1;
 4       while (low <= high) {
 5           int mid = (low+high)/2;
 6           if (array[mid].compare(key) < 0) {
 7               low = mid+1;
 8           } else if (array[mid].compare(key) > 0) {
 9               high = mid-1;
10           } else {
11               return mid;
12           }
13       }
14       return -1;
15   }
```

Suppose we change the code in the following ways. For each change, say whether the code will still work as intended. If not, explain what can go wrong.

7A) Change the test in line 4 to `low < high`.

The code does not work. If `low == high` then the function will return `-1`, even though `array[low]` might be the correct element.

7B) Change line 5 to `int mid = low;`

The code gives the correct answer still. However, in the worst case, it degenerates to linear search instead of binary search. This happens when the key is the last element of the array.

7C) Change line 7 to `low = mid;`

The code goes into an infinite loop if `mid == low`.

Write your anonymous code (*not* your name) here: ________________

Advanced question 8 (counting votes)

[DAT035, 2006-12-22, q7]

Your task is to design efficient algorithms for counting votes in an election. Assume that there are ***N*** voters and ***C*** candidates, and that each voter votes for exactly one candidate. Given the list of votes, your algorithm should calculate which candidate got the most votes.

The best algorithm depends on the relationship between ***N*** and ***C***.

8A) In ordinary elections, *C* is much smaller than *N*.

Which algorithm and data structure is suitable in this case? Answer by giving pseudocode, or by giving a careful explanation of the different steps in your algorithm. You should also give the order-of-growth of your algorithm in terms of ***N*** and ***C***.

Answer: Assign every candidate a number 0, ..., $C-1$. Use an integer array with C cells to count the votes. Iterate through all votes and for every candidate i, increase the value of the cell i. Finally loop through the array once to find the candidate with the maximum number of votes.

The first loop takes $\Theta(N)$ and the second $\Theta(C)$, so in total $\Theta(N+C)$.

8B) If *C* is much larger than *N* it is better to use another method.

Suggest a suitable algorithm and data structure, and give the order-of-growth of your algorithm in terms of ***N*** and ***C***.

Answer: Use a map from candidates to counts, implemented as a hash table. Iterate through all votes: if the candidate is not in the hash table, insert it with value 1; otherwise increase the value of the candidate. Finally iterate through the items in the hash table to find the candidate with the maximum number of votes.

The first loop takes $\Theta(N)$, and the second $\Theta(C_0)$ where $C_0 < N$ are the number of candidates that got a vote. In total we get order of growth $\Theta(N)$.

Write your anonymous code (*not* your name) here: ___________________

Advanced question 9 (set of integers)

[DIT961, 2018-08-24, q6]

Design a data structure for storing a set of integers. It should support the following operations:

- `new`: create a new, empty set
- `add`: add an integer to the set
- `member`: test if a given integer is in the set
- `delete`: delete a given integer from the set
- `increaseBy`: add a given integer to all the integers in the set

For example, calling `increaseBy(2)` on a set containing the values 1, 2, 3, 4, 5 should give a set containing the values 3, 4, 5, 6, 7.

You may use existing standard data structures as part of your solution – you don't have to start from scratch.

Write down the data structure or design you have chosen, plus pseudocode showing how the operations would be implemented. The operations must have the following order of growth:

- `new`: constant time
- `add`/`member`/`delete`: logarithmic time
- `increaseBy`: constant time

You should also give an explanation why your operations have that order of growth.

Answer: Use a hash table (or a balanced BST) plus an integer *k* which represents the total amount added in all calls to increaseBy. The operations work as follows:

- new: initialise the hash table to be empty, set *k* to 0
- increaseBy(n): set *k* = *k*+*n*
- add(n): add *n*-*k* to the hash table
- member(n): check if *n*-*k* is present in the hash table
- delete(n): delete *n*-*k* from the hash table

(The idea is that when the number *n* is present in the hash table, that "really" represents the number *n*+*k* being present in the set. Conversely, if the number *n* is present in the set, then *n*-*k* should be present in the hash table.)

All operations take (expected) constant time, if you use a hash table. If you use a balanced BST, add/member/delete take logarithmic time.