An instance about Test 1

Some students kept writing even after the TAs announced the end of the test.

To be fair:
Whoever did stop writing, please drop an email to both TAs and we will add 5 marks to your test.

While I understand students want to get higher marks, we should follow the instructions from the TAs. So, do NOT do it in Test 2!

Announcements

Deadline for Assignment 3: TODAY

Deadline for Programming assignment: 30 Nov, 2016

Class on 29 Nov (next Tue) will be cancelled.

Test 2:
Date: Dec 01 (Thur)
Time: 2:30 – 4:30pm
Venue: KB223
Topics: Everything up to and including sorting by comparison; focus on topics not yet tested in Test 1

Examination:
Dec 09 (9:30am – 12:30pm)
Topics: everything taught in the lectures

Selected exercises:

Q1. Sorting these numbers in decreasing order using (i) bubble sort; (2) selection sort; (3) insertion sort; (4) merge sort; (5) heap sort; (6) quicksort

35, 70, 5, 100, 78, 36, 49, 88, 21

Q2. The range of the following integers is from 0 to 10, sort the integers in decreasing order using counting sort.

4, 6, 0, 6, 2, 8, 9, 0, 1, 4, 3

Q3. Sort the following integers in decreasing order using radix sort.

444, 445, 604, 705, 235, 144, 45, 4, 605

Q4. Can we sort n real numbers with at most 2 decimal places in the range of [0..100] in o(n log n) time.

If yes, show how you sort them and why the lower bound of O(n log n) does not apply?
If no, argue why it is not possible.

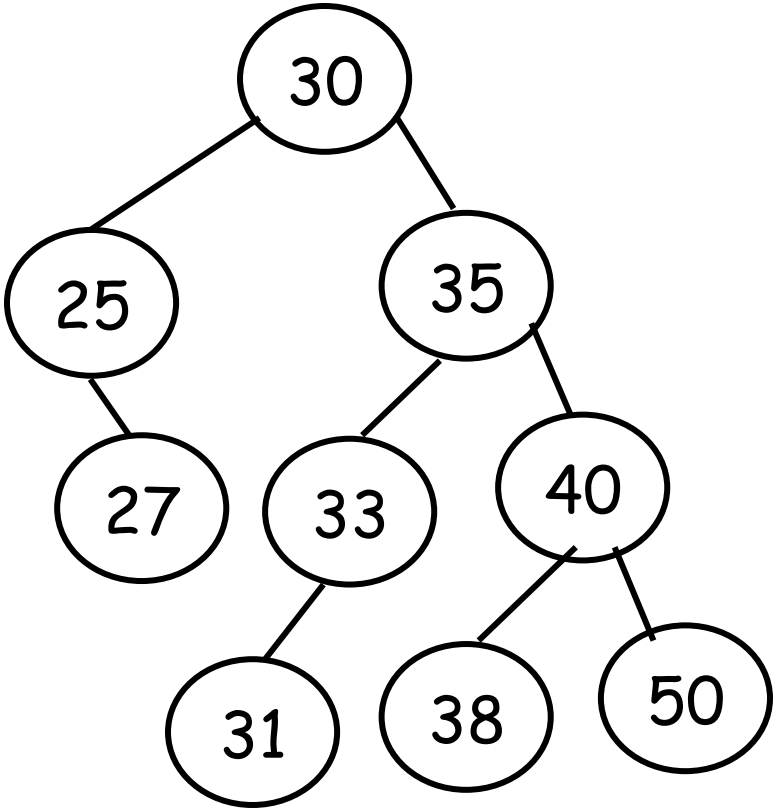
Q5. Create a (min)-heap for the following numbers.
200, 90, 35, 80, 100, 67, 21, 9

Q6. Enhance a min-heap to report the maximum element. Provide the additional data structure and also present the procedures for (i) Max() – return the maximum element; (ii) Insert(); and (iii) Deletemin(). What are the time complexities of your procedures.

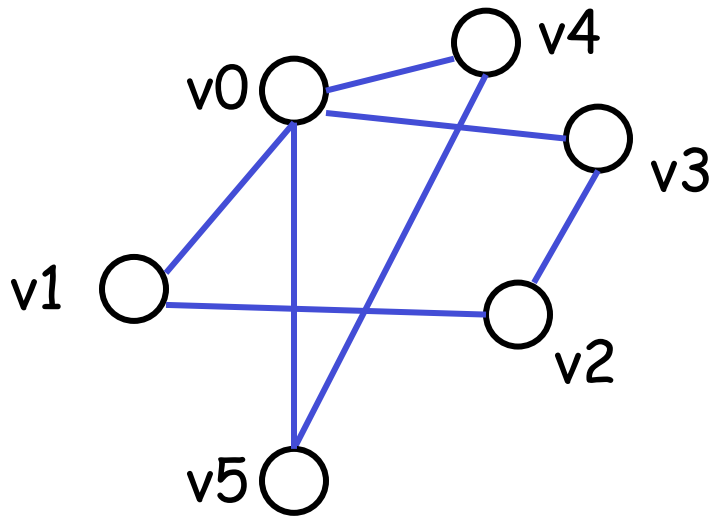
Q7. We insert the following into an initially empty AVL tree one by one: 30 50 80 100 130 90 150 180. Then, delete 100, 80, 130, 30, 50, 90 one by one.

Q8. Can you give an example such that deleting a node from an AVL tree will make the grandparent not satisfying the AVL tree property while the parent does satisfy the AVL tree property?

Q9. Consider the following AVL tree. Delete 27, what rotation we can perform to restore the balance of the tree?



Q10. Consider the following graph.



(a) Which of the followings are possible list of vertices for BFS?

(i) v0, v3, v2, v1, v4, v5

(ii) v5, v4, v0, v3, v1, v2

(iii) v1, v0, v2, v4, v3, v5

(b) List the vertices in order of BFS starting from v3.

(c) Which of the followings are possible list of vertices for DFS?

(i) v0, v5, v4, v3, v1, v2

(ii) v5, v0, v4, v1, v3, v2

(iii) v1, v0, v4, v5, v3, v2

(d) List the vertices in order of DFS starting from v3.

Q11. What is wrong with the following proof?

To show that $n^2 \neq O(n)$:

Let $c = 0.5$, $n_0 = 1$, then $\forall n \geq n_0$,
$n^2 > 0.5n$.
So, $n^2 \neq O(n)$

Generating Combinatorial Objects

Q12. Design a recursive algorithm to general all n-bit strings.

e.g. If n = 3, the algorithm should output all the followings:
000, 001, 010, 011, 100, 101, 110, 111

Hint: Try to generate the strings recursively by fixing the last bit.

Generating Combinatorial Objects

6) Design a recursive algorithm to general all n-bit strings.

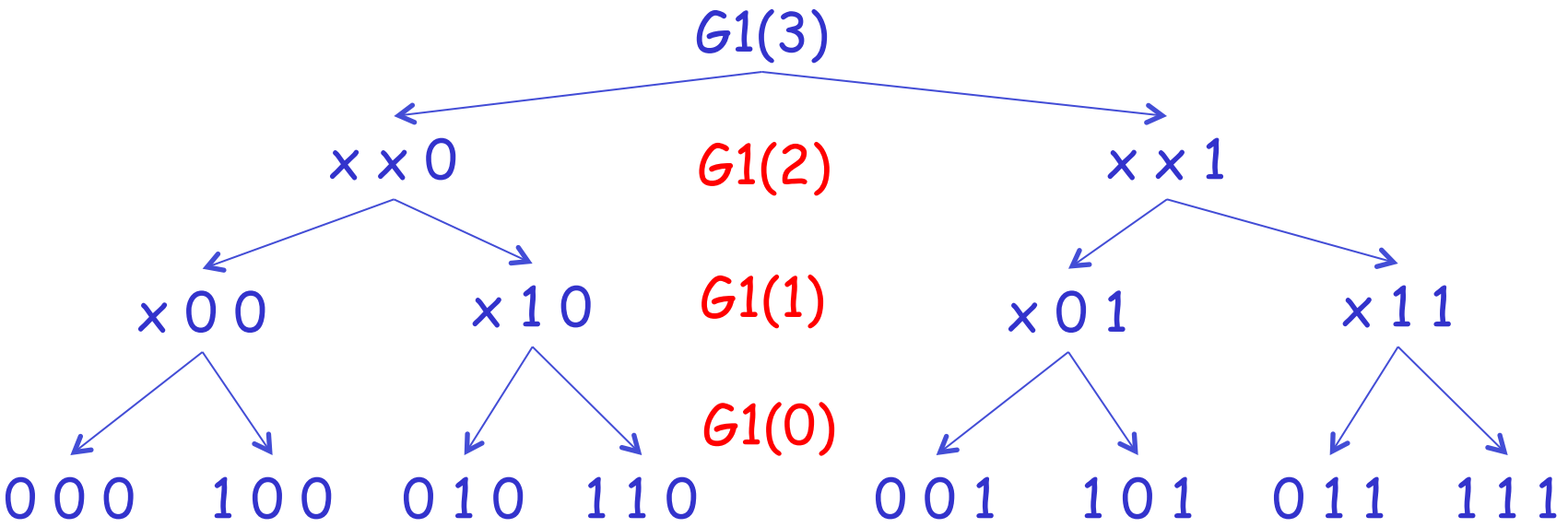
e.g. If n = 3, the algorithm should output all the followings:
000, 001, 010, 011, 100, 101, 110, 111

Can be defined recursively:
(1) Fix the last bit to be 0, then generate all (n-1)-bit strings.
(2) Fix the last bit to be 1, then generate all (n-1)-bit strings.

```
G1(n) // assume A stores the bit pattern
{
        if (n = 0) return
        A[n] = 0; G1(n-1);
        A[n] = 1; G1(n-1);
}
```

Q: When to print each generated string?

Look at the recursion in details (e.g. n = 3)



Each string in the lowest level (G1(0)) needs to be printed!

```
G2(n) // assume A stores the bit pattern
{
        if (n = 0) then
                print(A); return;
        A[n] = 0; G2(n-1);
        A[n] = 1; G2(n-1);
}
```

Q13. Given an array of n numbers and a value k, design an algorithm that determines if there exists two of the numbers whose sum is exactly k. Analyze the time complexity of your algorithm.

Q14. Given an integer n, the value of n is known to be 2^k for some positive integer k, design an efficient algorithm to find k. Assume that you cannot use the log function to do it.

Other topics:

Simple data structures, e.g. stack, queue

Huffman code

Hashing

....

Note:

The questions given here and last lecture may NOT relate directly to the questions in the examination. They are just SAMPLES!!

Good luck to all your examinations!