CS3230 Tutorial 6 (Transform and Conquer) Sample Solution

March 10, 2017

1 Question 1

Assume first that $V = 0$. Then $A[i] + A[j] = 0$ if and only if $A[i] = -A[j]$, i.e., these two elements have the same absolute value but opposite signs. We can check for presence of such elements in a given array in several different ways. If all the elements are known to be distinct, we can simply replace each element $A[i]$ by its absolute value $|A[i]|$ and solve the element uniqueness problem for the array of the absolute values in $O(n \log n)$ time with the presorting based algorithm. If a given array can have equal elements, we can modify our approach as follows: We can sort the array in nondecreasing order of their absolute values (e.g., -6, 3, -3, 1, 3 becomes 1, 3, -3, 3, -6), and then scan the array sorted in this fashion to check whether it contains a consecutive pair of elements with the same absolute value and opposite signs (e.g., 1, 3, -3, 3, -6 does). If such a pair of elements exists, the algorithm returns yes, otherwise, it returns no.

The case of an arbitrary value of s is reduced to the case of s = 0 by the following substitution: $A[i]+A[j] = s$ if and only if $(A[i]-V/2)+(A[j]-V/2) = 0$. In other words, we can start the algorithm by subtracting $V/2$ from each element and then proceed as described above.

(Note that we took advantage of the instance simplification idea twice: by reducing the problem's instance to one with s = 0 and by presorting the array.)

2 Question 2

Sort all the a_i's and b_j's by a $O(n \log n)$ algorithm in a single nondecreasing list, **treating b_j as if it were smaller than a_i in case of the tie $a_i = b_j$.**

Scan the list left to right computing the running difference D between the number of a_i's and b_j's seen so far. In other words, initialize D to 0 and then increment or decrement it by 1 depending on whether the next element on the list is a left endpoint a_i or a right endpoint b_j , respectively.

The maximum value of D is the number in question.

3 Question 3

The answer is given is Figure 1. Note that according to the definition of the height of a tree given in the textbook, the solution should be the one to the left. However, sometimes people also define the height of a tree as the "level". If you follow that definition, the solution to the right is acceptable. (Refer to textbook page 34, second paragraph.)

4 Question 4

Let's consider the max heap. Here are the solutions:

1. $O(n)$
2. $O(\log n)$
3. $O(\log n)$ (**the position of the element in the array is known to you**)

Detailed explanations were given in the tutorial.

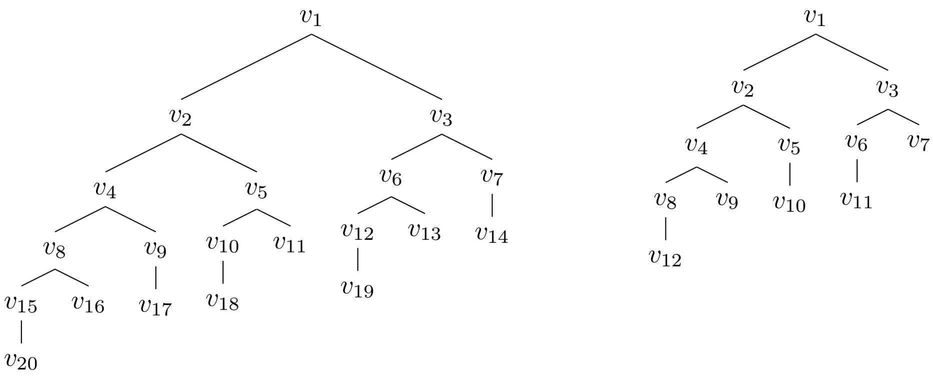

Figure 1: The height 5 AVL tree with the minimum number of nodes.

5 Question 5

Refer to the below link for detailed solution to this question:

http://mathworld.wolfram.com/LightsOutPuzzle.html

6 Question 6

Assume $n_{(10)} = d_k d_{k-1} \dots d_2 d_1 d_{0(3)}$, we have:

$$\begin{aligned} n_{(10)} = d_k d_{k-1} \dots d_2 d_1 d_{0(3)} &= d_k \cdot 3^k + d_{k-1} \cdot 3^{k-1} + \dots + d_2 \cdot 3^2 + d_1 \cdot 3^1 + d_0 \\ &= (d_k \cdot 3^{k-1} + d_{k-1} \cdot 3^{k-2} + \dots + d_2 \cdot 3^1 + d_1) \cdot 3 + d_0 \end{aligned}$$

We can develop a recursive procedure as follows:

ternary digits of n	d_k	d_{k-1}	$\dots$
product acumulator	a^{d_k}	$(a^{d_k})^3 \cdot a^{d_{k-1}} = a^{3d_k + d_{k-1}}$	$\dots$

7 Question 7

We can solve this problem by converting it into a graph. The state-space graph for this problem is given in Figure 2. Its vertices are labeled to indicate the states they represent: P, w, g, c stand for the peasant, the wolf, the goat, and the cabbage, respectively; the two bars || denote the river; for convenience we also label the edges by indicating the boat's occupants for each crossing. In terms of this graph, we are interested in finding a path from the intial-state vertex, labeled Pwgc||, to the final-state vertex, labeled ||Pwgc.

It is easy to see that there exist two distinct simple paths from the initial-state vertex to the final state vertex. If we find them be applying breadth-first search (BFS), we get a formal proof that these paths have the smallest number of edges possible. Hence, this puzzle has two solutions, each of which requires seven river crossings.

Any bugs and typos, please report to Roger Zimmermann (rogerz@comp.nus.edu.sg).

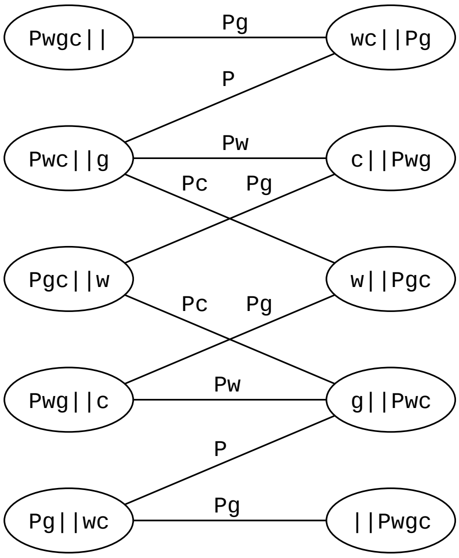

Figure 2: State-space graph for the peasant, wolf, goat, and cabbage puzzle.