# CS2223
# HW#7

**DUE**: Thursday, December 16

1. (6 points) A typical dynamic programming algorithm provides the cost of a solution or establishes the existence of a solution without actually constructing the solution. To see how to construct a solution by using an efficient mechanism which tests for the existence of a solution, solve the following:

You are given a boolean function *BlackBox* of two inputs:

-a list of integers $x_1,x_2,...,x_n$,

-an integer $q$

and you are told that, in time $O(1)$, *BlackBox* will return "true" if and only if there is some subset of $x_1,x_2,...,x_n$ whose sum is $q$. Design an algorithm (a program is not needed) with the same input which will return an actual subset of $x_1,x_2,...,x_n$ whose sum is $q$ if such a subset exists, or it should return "failure".

For example, *BlackBox*( (23,27,41,72,-4,6), 29) would return "true", but your algorithm with the same input would return (27,-4,6). Your algorithm may call *BlackBox* as often as it wishes and it should work in time $O(n)$.

2. (3 points) For the graph of **Figure 9.19** of our text, show how the algorithm identifies articulation points assuming the search starts at node 1 and that nodes of adjacency lists are stored in increasing order. Show the *dfs-tree*, as well as all values of *prenum* and *highest*. Also, identify all articulation points.

3. (5 points) An element $x$ is a *majority element* of array $A[n]$ if at least half the elements in $A$ are $x$. Thus, $A$=(0, -1, 5, 5, 5, 6, 5) has majority element 5, and $A$=(5, -1, 0, 12, 5, 6, 5) does not have a majority element. Design a probabilistic algorithm (precise pseudo-code suffices) to test if $A$ has a majority element. Your algorithm should work in time $O(n)$. If your algorithm says that $A$ has a majority element, it should be correct, and if your algorithm says that $A$ does not have a majority element, then for any array $A$ it may be wrong with probability $=\frac{1}{2}$.