1. (6 points) A typical dynamic programming algorithm computes 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 \textit{BlackBox} of two inputs:
- a list of integers \(x_1, \ldots, x_n\),
- an integer \(q\)

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

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

2. (6 points) Let \(\tau_s\) be the tree produced by a breadth-first search of graph \(G = (V,E)\) starting at \(s \in V\), and let \(\tau_d\) be the tree produced by a depth-first search of graph \(G = (V,E)\) starting at \(s \in V\), where the adjacency lists for each \(v \in V\) are the same for each search. Must it be the case that \(height(\tau_d) \geq height(\tau_s)\)? If you answer yes, then justify your answer. If you answer no, then provide a counterexample. Note that for any tree \(\tau\), \(height(\tau)\) is the number of edges on a longest path from \(s\) to a leaf of \(\tau\).

3. (8 points) Assume that you computed a dfs-tree \(\tau\) for graph \(G\), and you want to find the shortest cycle in \(G\). Neil makes the following claims (for all graphs \(G\)):

- Every cycle of \(G\) contains a backedge of \(\tau\).
- For any backedge from \(w\) to \(v\), the length of the only possible cycle containing edge \((v,w)\) is 1 plus the height of \(w\) in \(\tau\) minus the height of \(v\) in \(\tau\).
- Thus, to find the length of a shortest cycle of \(G\), you only need to add 1 to the minimum (over all backedges of \(\tau\)) of the difference between the heights of the vertices of the backedge.

Either prove that Neil is correct or give a counterexample to refute him.
CS525DA
HW#6 SOLUTIONS

1. \( S \leftarrow x_1, \ldots, x_n \)
   \[
   \text{if not BlackBox}(S,q) \text{ then return } \text{("failure")} \quad O(1)
   \]
   \[
   \text{for } i \leftarrow 1 \text{ to } n \text{ do} \quad n \text{ times}
   \]
   \[
   S \leftarrow S \setminus x_i
   \]
   \[
   \text{if not BlackBox}(S,q) \text{ then } \quad \triangleright \text{we really need } x_i; \text{ put it back}
   \]
   \[
   S \leftarrow S \cup x_i \quad O(1)
   \]

Since the body of the loop is executed in \( O(1) \) time, the time to execute the loop (and the program) is \( O(n) \).

2. It is always the case that \( \text{height}(\tau_d) \geq \text{height}(\tau_s) \). For any \( v \in V \), the height of \( v \) in \( \tau_s \) is the distance \( \sigma \) from to \( v \), and the corresponding path is a shortest path. Since in any tree the path from \( \sigma \) to \( v \) is a path in the underlying graph, the path from \( \sigma \) to \( v \) in \( \tau_d \) can not be shorter than a shortest path from \( \sigma \) to \( v \) in \( \tau_s \). Thus, the height of any vertex in \( \tau_s \) is less than or equal to its height in \( \tau_d \).

3. As usual, Neil is wrong. Consider the dfs tree

\[
\begin{align*}
& v \\
& \quad \downarrow \\
& a \\
& \quad \downarrow \\
& b \\
& \quad \downarrow \\
& w \\
& \quad \downarrow \\
& c
\end{align*}
\]

for graph
Neil would have us believe that the shortest cycle in $G$ has length 4, whereas it contains a triangle.