Vladimir Hugec

Question 1:

I was unable to do this problem.

Question 2:

2.1 A

If I know that all the numbers are positive, then the subsequence with the largest sum has to be the sum of all the numbers in the array. Computing this sum is $O(n)$.

2.2 B

If I know that all the numbers are negative, the largest subsequence is going to be a single number, $(i = j)$ and where i is the min of the array. Walk through the array and test each value, keep track of the min at each step, return the min at the end. This would be $O(n)$.

2.3 C

Keep track of only the maximum value computed thus far. For each index compute the cumulative sum to every other index except for those behind the starting index. Return the stored value.

2.4 D

Assume the array we are working on is: $A = [1, -2, 3, 6, -5]$ For any index i and j, the cumulative sum of those indexes and those in that range can be written as the sum from i to $j-1$ plus j. Similarly, computing sum of $i+1$ to j is sum of i to j minus i. Once one of these intermediate sums has been computed, they can be stored in a n x n array where index (i, j) stores the sum from i to j. However, since we also know that $i \leq j$. We will need to fill in $\frac{1}{2}$ the n x n array in order to be certain of our answer Call this array S. We start with $i = 0$ and $j = 0$, $S[0][0] = A[0] = 1$, this is our base case, and for all j we compute and save into the n x n array. So $S[0][1]$ as discussed above would be $S[0][0] + A[1]$. Fill in the table like that for all j as shown below:

j \ i	0	1	2	3	4
0	1	x	x	x	x
1	-1		x	x	x
2	2			x	x
3	8				x
4	3				

Continuing to fill in the table, lets begin with $i = 1$ and $j = 1$ we already have all the information we need to know this value since we have already computed ($i = 0$,$j = 1$). So cell $S[i][j]$ is simply $S[i-1][j] - A[i-1] = -1 - 1 = -2$. Filling in all the subsequent indexes the same way we get:

j \ i	0	1	2	3	4
0	1	x	x	x	x
1	-1	-2	x	x	x
2	2	1	3	x	x
3	8	7	9	6	x
4	3	2	4	1	-5

Since we were keeping track of the highest index found in S at each step, the algorithm would return the indices $(2, 3)$ which translate to a value of 9. This algorithm would be $O(\frac{n^2}{2})$ time and space needed would be $n^2 + n$