Midterm 2. Examples of questions from previous exams

In your preparation to the Midterm exam on April 7, check out these questions taken from the previous exams.

Also, get over the problems presented in Chapters 15 and 16.

Problem 1 A (12 points). The 1-0 Knapsack problem: Given is a knapsack with maximum capacity *W*, and a set *S* consisting of *n* items. Each item *i* has some weight w_i and benefit value b_i (all w_i , b_i and *W* are integer values). Problem: How to pack the knapsack to achieve maximum total value of packed items? Each item must be either entirely accepted or rejected.

(1) Explain the following optimality condition where $B[k, w]$ is the total benefit of for a k with total weight w:

$$B[k,w] = \begin{cases} B[k-1,w] & \text{if } w_k > w \\ \max\{B[k-1,w], B[k-1,w-w_k]+b_k\} & \text{else} \end{cases}$$

(2) Consider the following numerical example:

$n = 4$ (# of elements); $W = 5$ (max weight); Items (weight, benefit): (2,3), (3,4), (4,5), (5,6).

Fill in the table of $B[k, w]$:

i\W	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0					
2	0					
3	0					
4	0					

Problem 1 B (12 points). Maximum value contiguous subsequence.

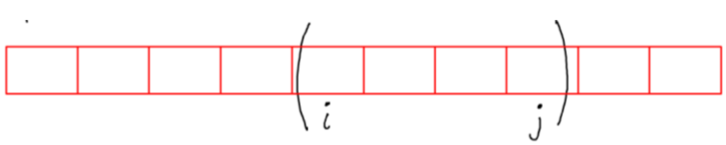
Given an input array A[1..n] of real numbers, both positive and integer, find the contiguous subsequence A[i..j] that maximizes $\sum_{l=i}^{j} A[l]$.

Consider the optimality condition based on the following observation:

For M[j] as the max sum over all windows ending at j, evaluate:

M[j] = max{M[j-1]+A[j], A[j]}.

Depending on what is greater: (M[j-1]+A[j]) or A[j], extend the previous subsequence or start a new one containing only A[j].



Task

(1) Consider an example of an array A=(5,-2,6,10,1,-4). Apply the optimality condition and find the indexes of the max value contiguous subsequence.
(2) Sketch an algorithm.
(3) What is the running time complexity O(g(n))?

Problem 2 (8 points). Coin change. An integer value *N* should be represented as a sum of an unlimited number of coins with *k* given denominations *S*. The problem is to find the minimal number *m* of coins to change *N*.

(1) What algorithm will you try to use: based on the greedy approach, or on the dynamic programming?
(2) Consider an example: *S* = (1, 5, 10, 25), *N* = 57. Find the minimal number of coins *m*. Will be it possible to use the greedy approach?
(3) Consider an example: *S* = (1, 2, 5, 6), *N* = 10. What is *m* here? Did you use the greedy approach?
(4) Which of the above approaches will be most general?

Problem 3 (10 points). Horn clause

A Horn clause is a disjunction of variables of one of two types: (a) with all negative and a single positive variables; or (b) with all variables being negative.

Given a set of clauses of these two types, the goal is to determine whether there is a consistent explanation: an assignment of true/false values to the variables that satisfies all the clauses.

(1) Describe the greedy algorithm of finding the satisfying assignment.
(2) Find the satisfying assignment for the following Horn clauses if such an assignment exists:

$(w \wedge y) \Rightarrow x$, $x \Rightarrow y$, $\Rightarrow x$, $(x \wedge y) \Rightarrow w$, $(\bar{z})$.

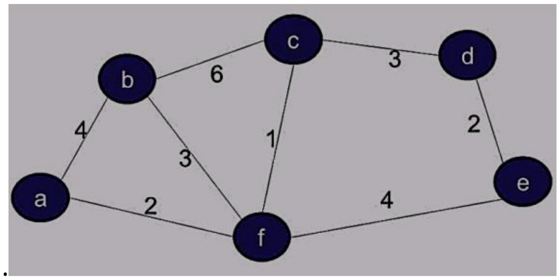
Assign the values T(rue) or F(alse) to the variables below: *x* = *y* = *z* = *w* =

Problem 4 (10 points). Huffman encoding

(1) Explain the greedy algorithm for the Huffman encoding
(2) Find a binary Huffman code for the source alphabet $\{a_1, a_2, a_3, a_4, a_5\}$, with symbol probabilities $p_1 = 0.60$, $p_2 = 0.10$, $p_3 = 0.10$, $p_4 = 0.05$, and $p_5 = 0.15$. Show your work.

Problem 5 (10 points).

The Minimal Spanning Tree. Consider the following undirected graph



.

(1) Use one of the methods of solving the MST problem (Kruskal or Prim) and explain the steps leading to the solution.
(2) Show a solution on the graph.

Other examples of questions

Problem 1. The Longest Common Subsequence algorithm:

(1) Compute the table *c*[*i,j*] "bottom-up" using the following definition:

$$c[i,j] = \begin{cases} 0 & \text{if } i = 0 \text{ or } j = 0, \\ c[i-1, j-1] + 1 & \text{if } i, j > 0 \text{ and } x_i = y_j, \\ \max(c[i, j-1], c[i-1, j]) & \text{otherwise.} \end{cases}$$

(2) Store the value *b*[*i,j*]={←} {↑} or {↖}, depending on whether the optimum is *c*[*i,j*-1], *c*[*i*-1,*j*], or *c*[*i*-1,*j*-1]+1.

(a) For two given sequences X={**abcbdab**} and Y={**bdcababa**} find the solution of LCS problem?

(b) The table below shows the first steps of the algorithm implementation. Continue and fill in the empty cells with numbers and arrows.

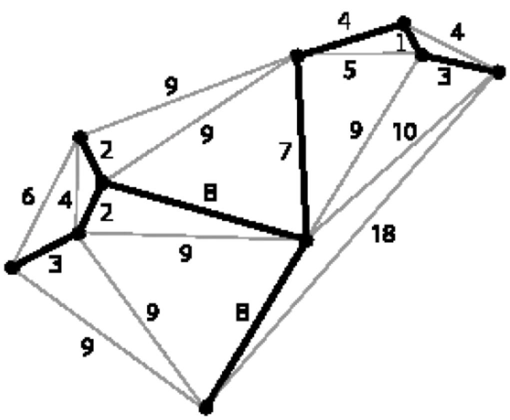
i \ *j*	1 *b*	2 *d*	3 *c*	4 *a*	5 *b*	6 *a*
1 *a*	↑ 0	↑ 0	↑ 0	↖ 1	← 1	↖ 1
2 *b*	↖ 1	← 1	← 1	↑ ← 1	↖ 2	← 2
3 *c*	↑ 1	↑ ← 1	↖ 2	← 2	↑ ← 2	↑ ← 2
4 *b*	↖ 1	↑ ← 1	↑ 2	↑ ← 2	↖ 3	← 3
5 *d*	↑ 1	↖ 2	↑ ← 2			
6 *a*	↑ 1	↑ 2	↑ ← 2			
7 *b*	↖ 1	↑ 2	↑ ← 2			

Solution:

i \ j	1 b	2 d	3 c	4 a	5 b	6 a
1 a	↑ 0	↑ 0	↑ 0	↖ 1	← 1	↖ 1
2 b	↖ 1	← 1	← 1	↑ ← 1	↖ 2	← 2
3 c	↑ 1	↑ ← 1	↖ 2	← 2	↑ ← 2	↑ ← 2
4 b	↖ 1	↑ ← 1	↑ 2	↑ ← 2	↖ 3	← 3
5 d	↑ 1	↖ 2	↑ ← 2	↑ ← 2	↑ 3	↑ ← 3
6 a	↑ 1	↑ 2	↑ ← 2	↖ 3	↑ ← 3	↖ 4
7 b	↖ 1	↑ 2	↑ ← 2	↑ 3	↖ 4	↑ ← 4

(c) Explain the Longest Common Subsequence algorithm.

Problem 2. Kruskal method is applied to find the solution (solid lines) of the Minimal Spanning tree problem for the following undirected graph.
Explain how it was produced with reference to the functions makeset(x), find(x), and union_by_rank(x,y).



Problem 3. Determine the maximum profit with knapsack capacity 11 if the following items are available:

$w1 = 7, w2 = 5, w3 = 4, w4 = 2, w5 = 1, w6 = 5$

$p1 = 40, p2 = 35, p3 = 18, p4 = 4, p5 = 10, p6 = 2$

Which items should you take? Explain the algorithm.

Answer.

63: 2, 3, 5.