On-line «Master in Computer Vision» educational program.

Entrance examination test (each problem is evaluated by 10 points)

Variant 2.

Task 1. Find all pairs *(a, b)* of positive integer numbers *a* and *b*, such that, the following system of equations has infinitely many solutions

$ax_1+6x_2 = 2$

$2x_1+bx_2 =1$

Solution.

For a>0, initial system of equations is equivalent to the system

$2ax_1+12x_2 = 4$ (first equation is multiplied by 2)

$2ax_1+abx_2 = a$ (second equation is multiplied by a)

For any solution (x_1,x_2) one has

$(ab-12)x_2=a-4$

If $ab-12\neq0$, then the system has a unique solution

$x_1=(2b-6)/(ab-12)$, $x_2=(a-4)/(ab-12)$,

If $ab-12=0$, then $ab=12$ and the system of equations takes the form

$2ax_1+12x_2 = 4$

$2ax_1+12x_2 = a$

If $a\neq4$, then this system has no solutions.

If $a=4$, then $b=3$ $(ab=12)$ and initial system of equations is

$4x_1+6x_2 = 2$

$2x_1+3x_2 =1$

This is in fact only one equation $2x_1+3x_2 =1$, and this equation has infinitely many solutions (x_1,x_2).

Answer: the system of equations has infinitely many solutions if and only if a=4, b=3.

Task 2. A simple, undirected graph is given by its adjacency matrix.

- find the degrees of the vertices of the graph, and the diameter of the graph
- show that this graph has no Euler cycle
- what is the minimum number of edges (and exactly which edges) to add to the graph for the Euler cycle to appear.

0 1 0 1 1 0
1 0 1 1 0 0
0 1 0 0 1 1
1 1 0 0 0 1
1 0 1 0 0 0
0 0 1 1 0 0

Solution.

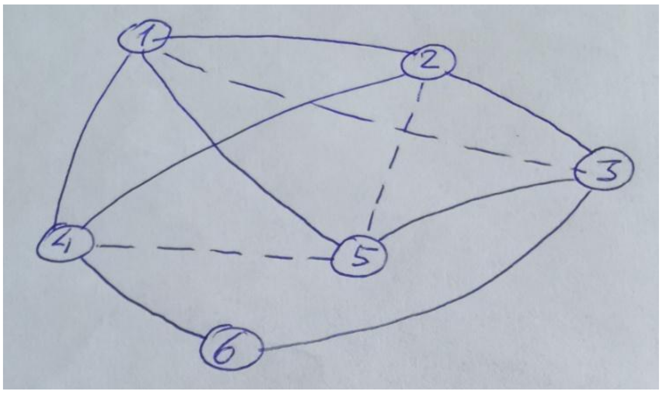
- Denote by G the given graph, and by d_i the degree of the vertex I in G. Degree of the vertex i is the sum of elements of the row i (or column i) of adjacency matrix. Therefore one obtain

$d_1=3$, $d_2=3$, $d_3=3$, $d_4=3$, $d_5=2$, $d_6=2$

Diameter of the graph is the length of a maximum shortest path between pairs of vertexes of the graph. Investigating all pairs of vertexes one get diam(G)=2 (in fact it is sufficient to investigate only 7 pairs of vertexes which are not directly connected by an edge).

- Euler cycle in the graph exists if and only if the degrees of all vertexes are even. This is not true for the given graph.
- one need to add edges such that the degree of all vertexes in the obtained graph will be even. This is not possible with 1 edge (1 edge can change degree of only 2 vertexes).

To change degree of vertexes 1,2,3,4 with two edges we need to draw edges between pairs of these vertexes. For example edge (1,3) and (2,4). For the given graph it is not possible because for all two pairs of vertexes from 1, 2, 3, 4 the graph already has an edge and we can't double edges.
It is possible to add 3 edges and make all degrees of vertexes even. For example (1,3), (3,5), (4,5). Therefore the minimum number of edges to add to the graph to make Euler cycle appear is 3.



Task 3. Cumulative distribution function for random variable X has the form

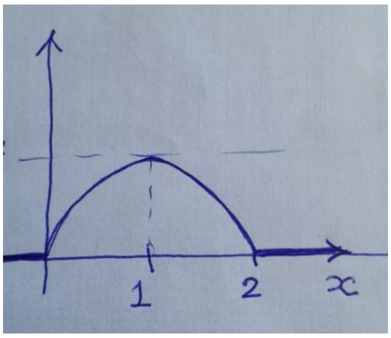
$F(x)= (3x^2 - x^3)/4$ on $[0;2]$ and $F(x)=0, x<0$, $F(x)=1, x>2$.

- find the probability density function of X. Draw the graph of it.
- calculate the expected value (mathematical expectation) of X.
- For the random variable Y=2X-1, find the probability P(1<Y<5).

Solution.
- Probability density function *f(x)* is the derivative of the cumulative distribution function *F(x)*. Therefore

$f(x)= (6x - 3x^2)/4$ on $(0;2)$ and $f(x)=0, x<0$, $f(x)=0, x>2$.

For the points $x=0$, $x=2$ one needs an additional investigation of derivatives from the left and from the right. In our case the derivatives of *F(x)* at $x=0$ from the left and from the right are the same and equal to 0. The same is for the one side derivatives of *F(x)* at $x=2$. Therefore $f(0)=f(2)=0$.



- By definition, **e**xpected value (mathematical expectation) of X is

E[X]=$\int xf(x)dx = \int_0^2 \frac{1}{4}(6x - 3x^2)dx = 1$

- P(1<Y<5)= P(1<2X-1<5)= P(2<2X<6)= P(1<X<3)=F(3)-F(1)=1-1/2=1/2=0.5

Task 4. Write a pseudo-code (or code on any programming language) of an algorithm, which find two elements in an array of integer numbers (positive and negative), such that the sum of these elements is maximal. Discuss the computational complexity of your algorithm.

Solution:

Bad:

get_elements_with_max_sum(*array*):
max, *elem*1, *elem*2 = -inf, 0, 0
for *i* = 0, …, |*array*| - 1
for *j* = *i*+1, …, |*array*| - 1
if *array*[*i*] + *array*[*j*] > *max*
max, *elem*1, *elem*2 = *array*[*i*] + *array*[*j*], *array*[*i*], *array*[*j*]
return *elem*1, *elem*2

The computational complexity of this solution is $O(n^2)$, because there are two nested loops with $O(n)$ iterations. More precisely we have $n + n-1 + \ldots + 1 = n(n+1)/2$ operations. Here n is the size of the array.

Normal:

get_elements_with_max_sum(*array*):
sort(*array*)
n = |*array*|
return *array*[*n*-2], *array*[*n*-1]

The computational complexity of this solution is $O(n\log n)$ because of sort() procedure.

Good:

get_elements_with_max_sum(*array*):
max1, *max*2 = -inf, -inf
for *i* = 0, …, |*array*| - 1
if *array*[*i*] > *max*1
max2, *max*1 = *max*1, *array*[*i*]
else if *array*[*i*] > *max*2
max2 = *array*[*i*]
return *max*2, *max*1

The computational complexity of this solution is $O(n)$, because there is only one loop with n iterations.