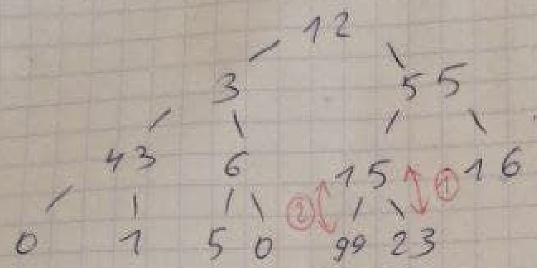
SOFYA AKSENYUK

150284

ex. 1. O(n logn)

HS best, av, worst.

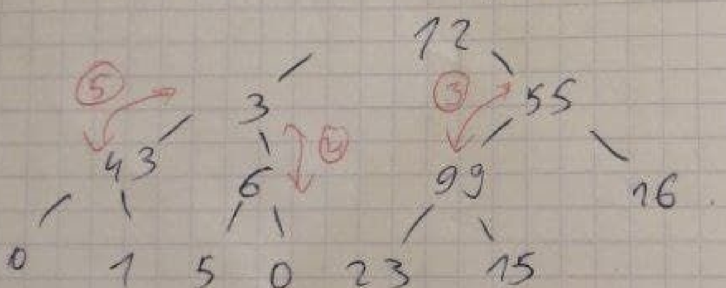
12, 3, 55, 43, 6, 15, 16, 0, 1, 5, 0, 99, 23



Steps:

1) 23 ↔ 15, 23 > 15

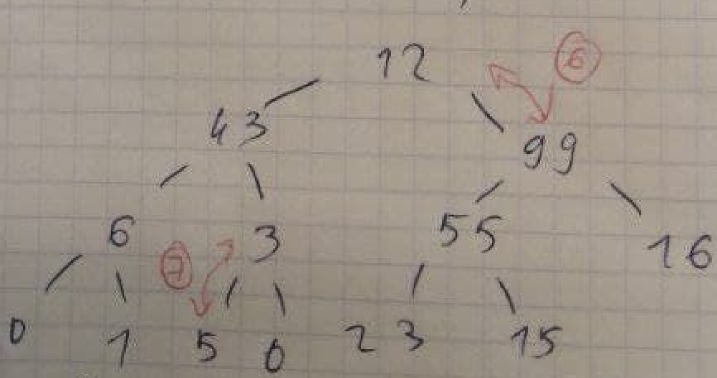
2) 99 ↔ 23, 99 > 23



3) 99 ↔ 55, 99 > 55

4) 6 ↔ 3, 6 > 3

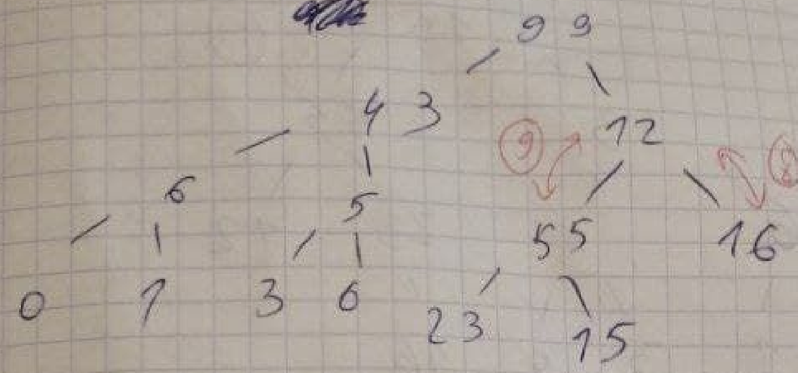
5) 43 ↔ 6, 43 > 6



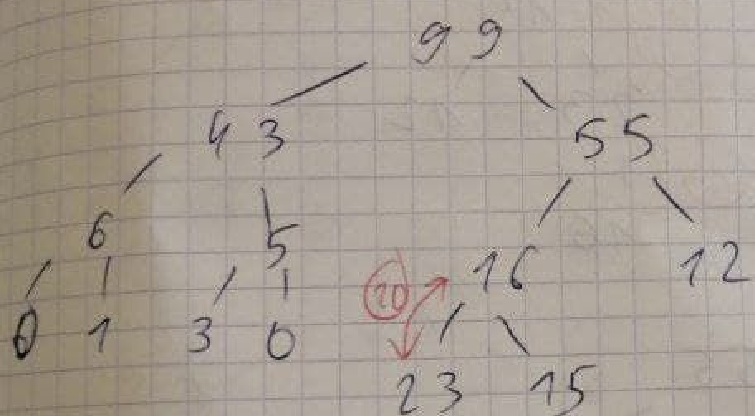
6) 99 ↔ 12, 99 > 12

7) 5 ↔ 3, 5 > 3.

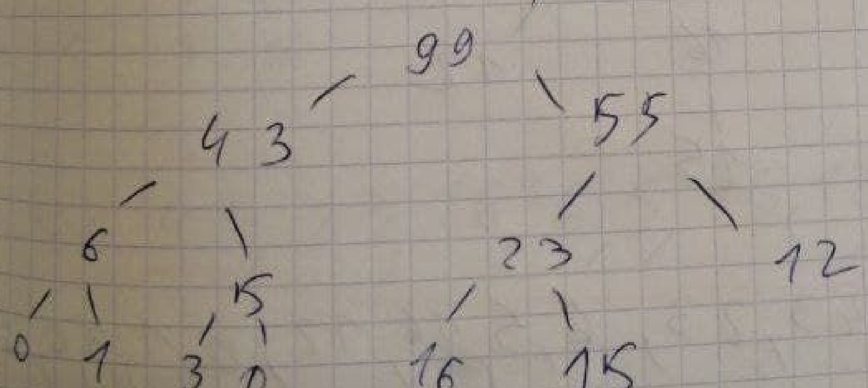
SOFYA AKSENYUK
150284



8) 16 ⟷ 12, 16 > 12
9) 55 ⟷ 16, 55 > 16



10) 23 ⟷ 16, 23 > 16

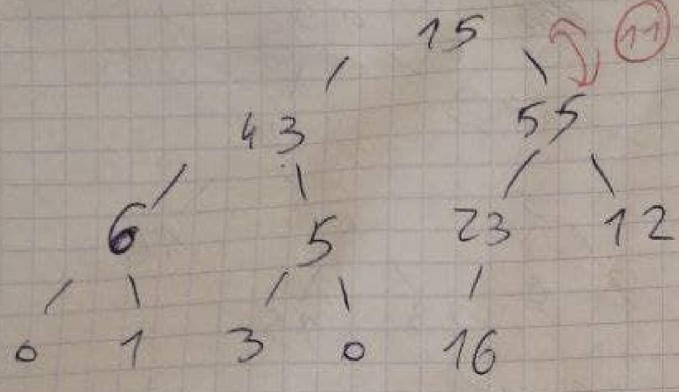


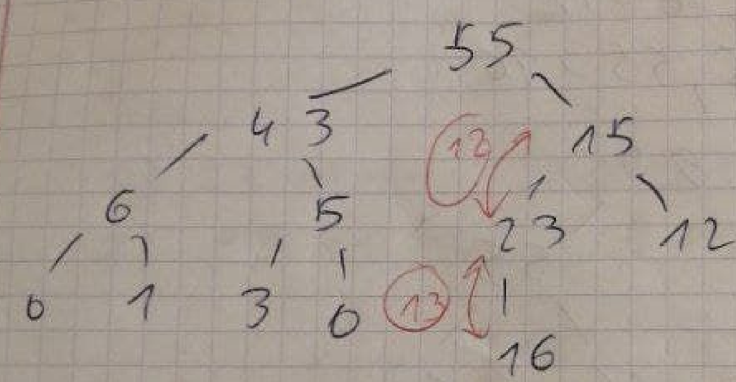
=> $\boxed{99}$, 43, 55, 6, 5, 23, 12, 0, 1, 3, 0, 16, 15

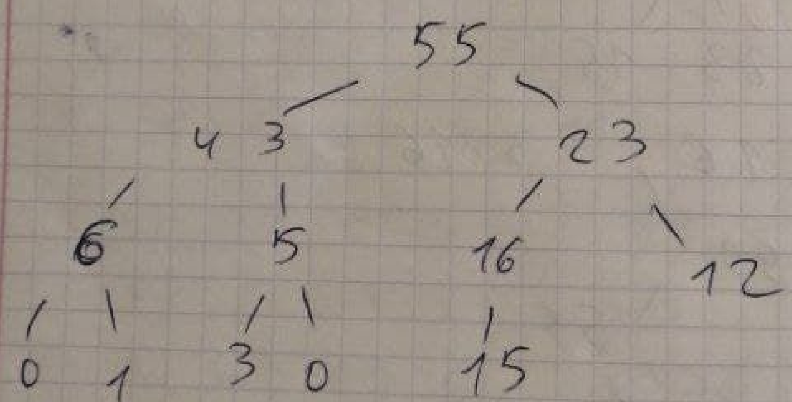
swap

SOFYA AKSENYUK

150284

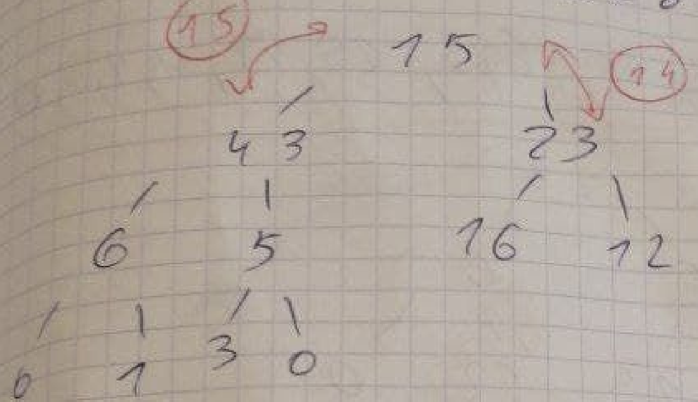


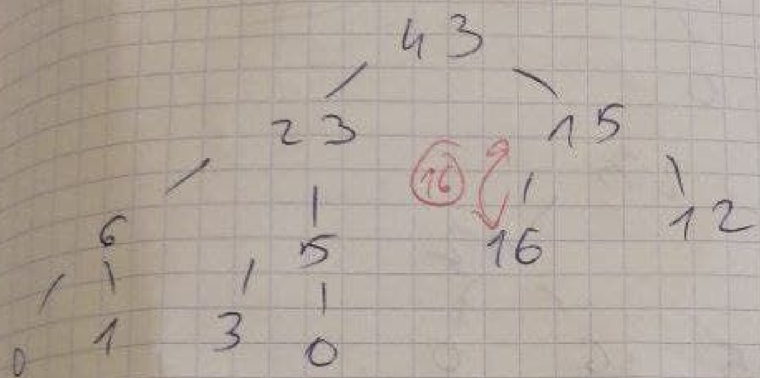


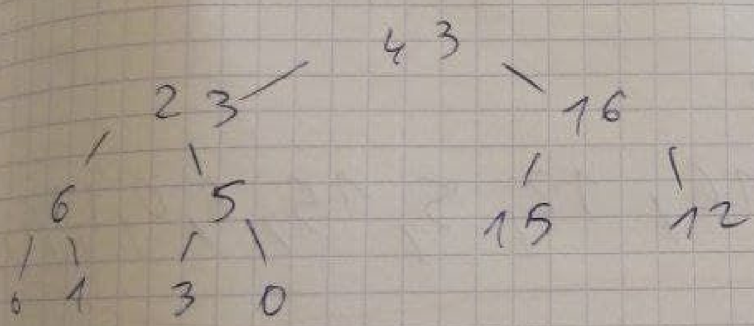


=> $\boxed{55}$, 43, 23, 6, 5, 16, 12, 0, 1, 3, 0, 15

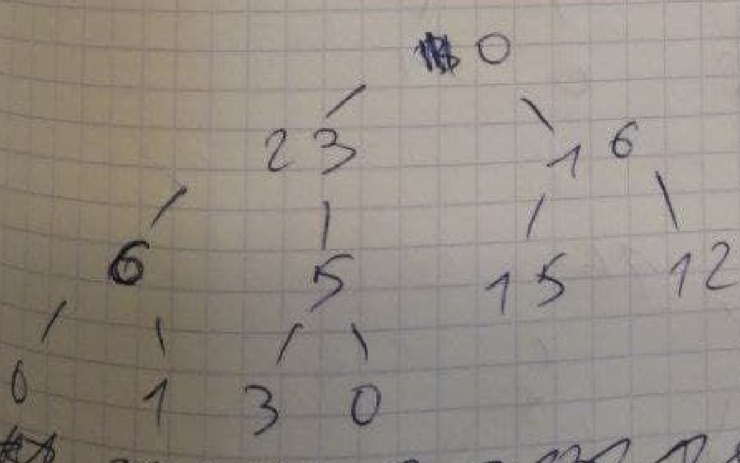
SOFYA AKSENYUK
150284







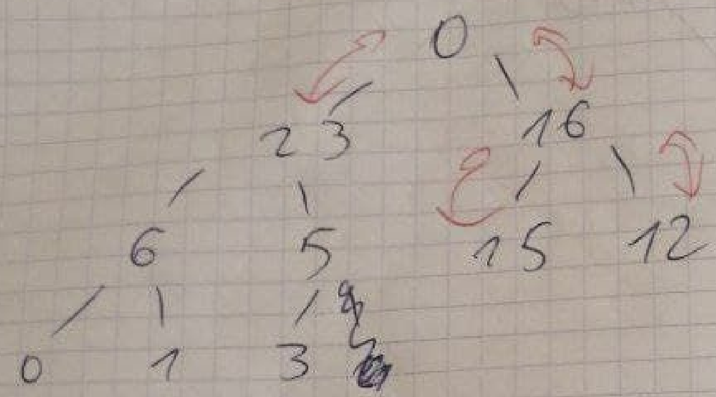
=> $\boxed{43}$, 23, 16, 6, 5, 15, 12, 0, 1, 30.

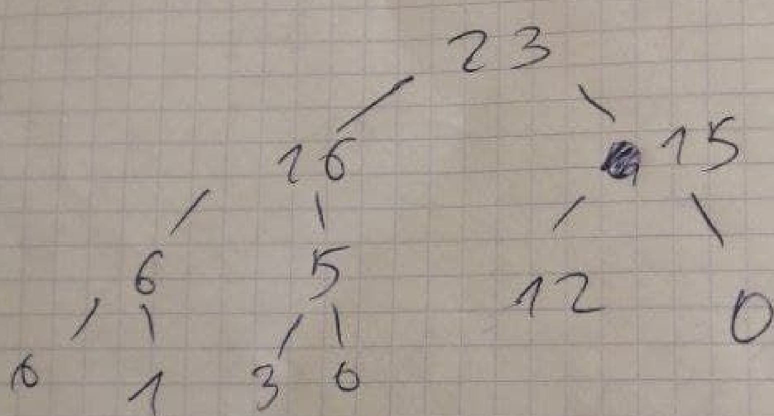


SOFYA AKSENYUK

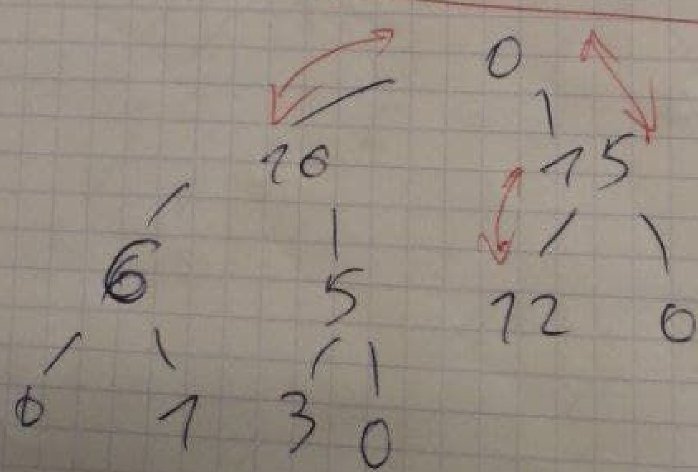
150284

=> 43, 23, 16, 6, 5, 15, 12, 0, 1, 3, 0





=> 23, 16, 15, 6, 5, 12, 0, 0, 1, 3, 0



TAXI

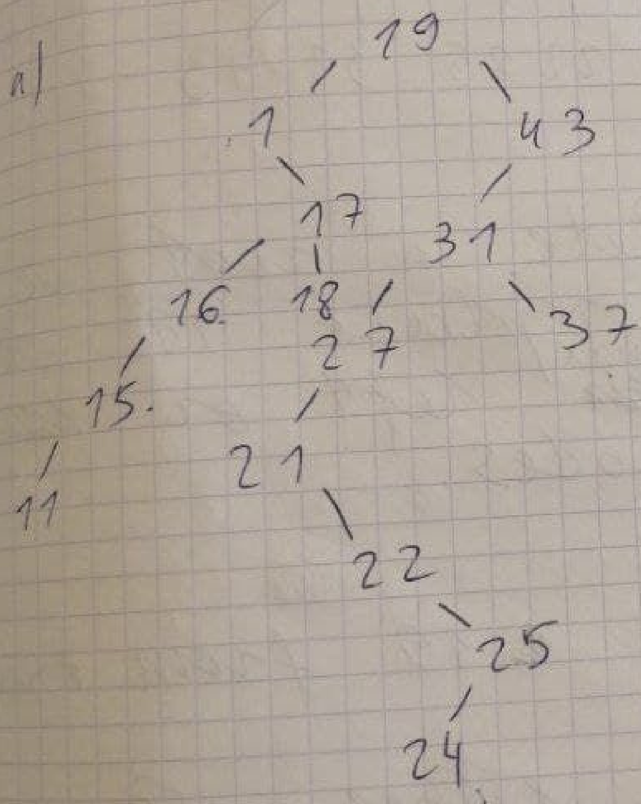
SOFYA AKSENYUK
150284.

ex. 2.

19, 43, 31, 1, 27, 17, 21, 16, 18, 15, 22, 37, 25,
11, 24

a) BST
b) pre-order...
c) BBST
d) how to delete root from BST

a)

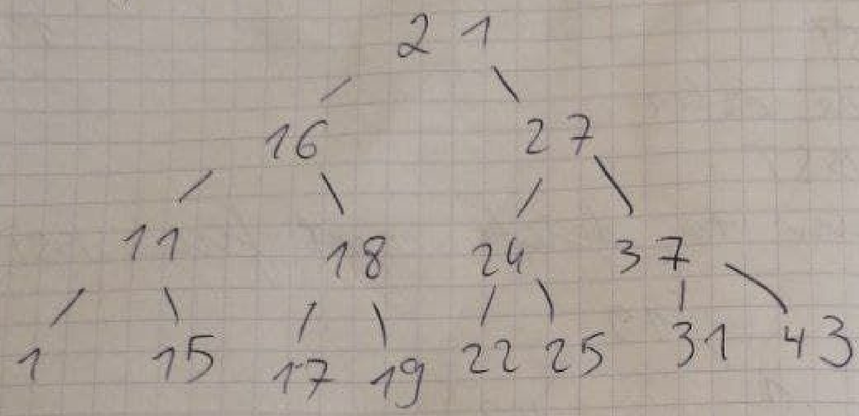


b) pre-order (r, L, R): 19, 1, 17, 16, 15,
11, 18, 43, 31, 27, 21, 22, 25, 24, 37
in-order (L, r, R): 11, 15, 16, 17, 18,
~~post order~~ 37, 43.
1, 19, 21, 22, 24, 25, 27, 31, 11, 15, 16, 18,
post-order (L, R, r): 27, 37, 31, 43, 19.
17, 1, 24, 25, 22, 21,

c) BBST: SOFYA AKSENYUK 150227

1) sort the array:

1, 11, 15, 16, 17, 18, 19, 21, 22, 24, 25, 27, 31, 37, 43



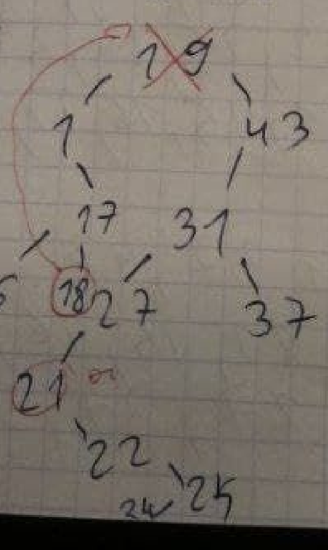
d) to delete a root we have to replace it with either max left or min right nodes.

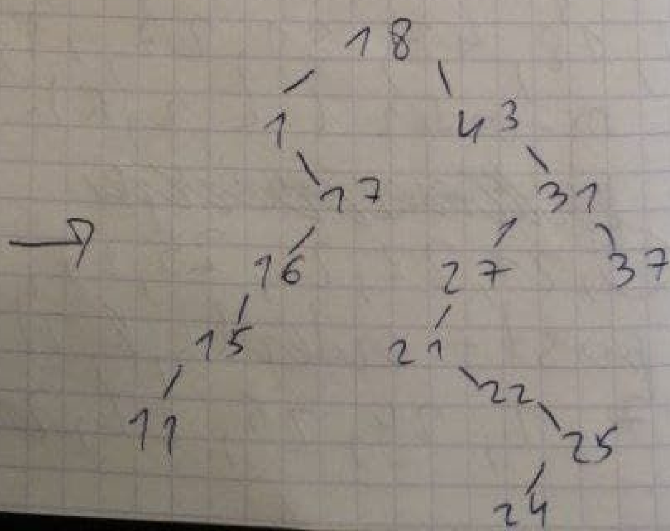
example:

delete 19 from tree from @.

=> replace either with 18 or with 21.

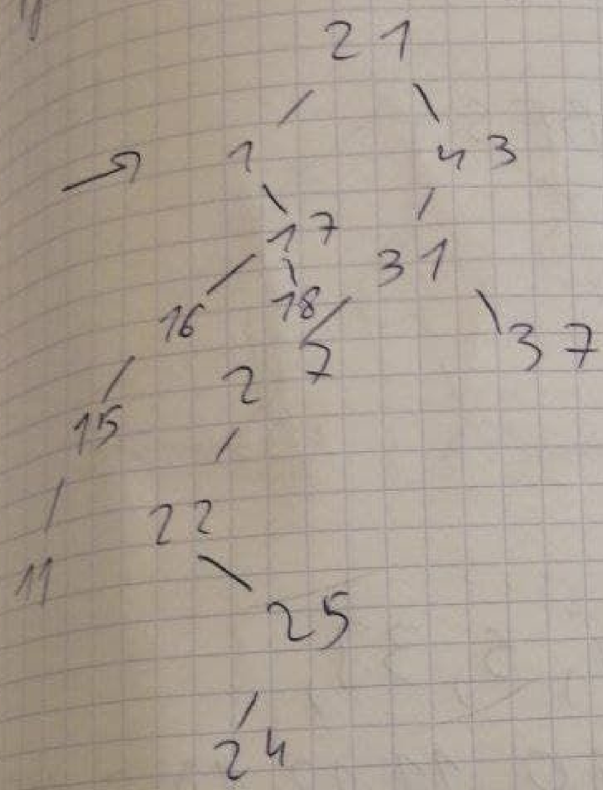
• If max left (i.e. 18):





SOFYA ARSENYUK, 150284,

g with min right (i.e, 21):



METRO.

SOFYA AKSENYUK
150284.

ex. 3.

a) DFS and BFS.

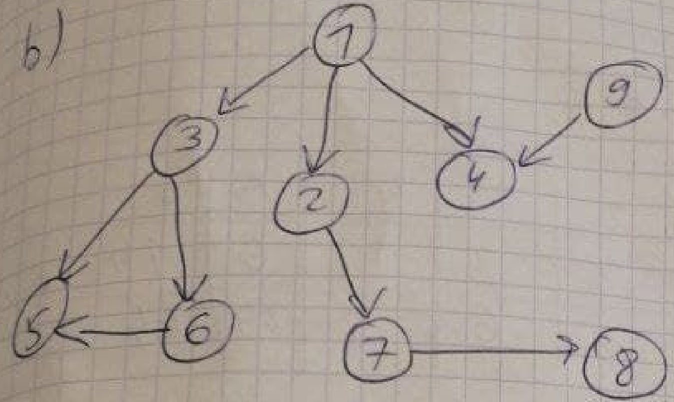
DFS: [1, 2, 4, 5, 3],
[6, 8, 10, 11, 12, 13, 7, 15].
[9, 14, 16]

BFS: [1, 2, 3, 5, 4],
[6, 8, 11, 12, 13, 15, 10, 7]
[9, 14, 16]

Since each of Searching Algorithms consist of several sets (particularly, three), disconnected graph can be concluded.

SOFYA AKSENYUK
150284.

b)



Steps:

	in-degree	
1	0	
2	1	$-1=0$
3	1	$-1=0$
4	2	$-2=0$
5	2	$-2=0$
6	1	$-1=0$
7	1	$-1=0$
8	1	$-1=0$
9	0	

· Topological order:

1, 9, 2, 3, 4, 7, 8, 6, 5.

In steps we calculate the in-degree of each node to check if there are any dependent nodes to a particular one.

If in-degree = 0, then it goes to the final sequence.

TRAIN

SOFYA AKSENYUK
150284

ex. 4.

a) <u>DP</u>:

$b = 7, n = 6.$

i	1	2	3	4	5	6
s_i	4	3	2	1	6	14
w_i	3	2	6	2	24	64

b) optimal solutions
c) optimal value for $b=5, n=4$.
d) complexity of DP.

We use formulas (using DP):

- $f(i, l) = f(i-1, l)$ if $l < s_i$.
- $f(i, l) = \max\{ f(i-1, l), f(i-1, l-s_i) + w_i \}$

if $i \geq s_i$.

i/l	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	0	0	3	3	3	3	3
2	0	0	2	2	3	3	3	5
3	0	0	6	6	6	8	9	9
4	0	2	6	8	8	8	10	11
5	0	2	6	8	8	8	24	26
6	0	2	6	8	8	8	24	26

SOFYA AKSENYUK
150284

=> b) optimal solution for $b=7$ and $n=6$: 26 which we got from elements 5 and 4.

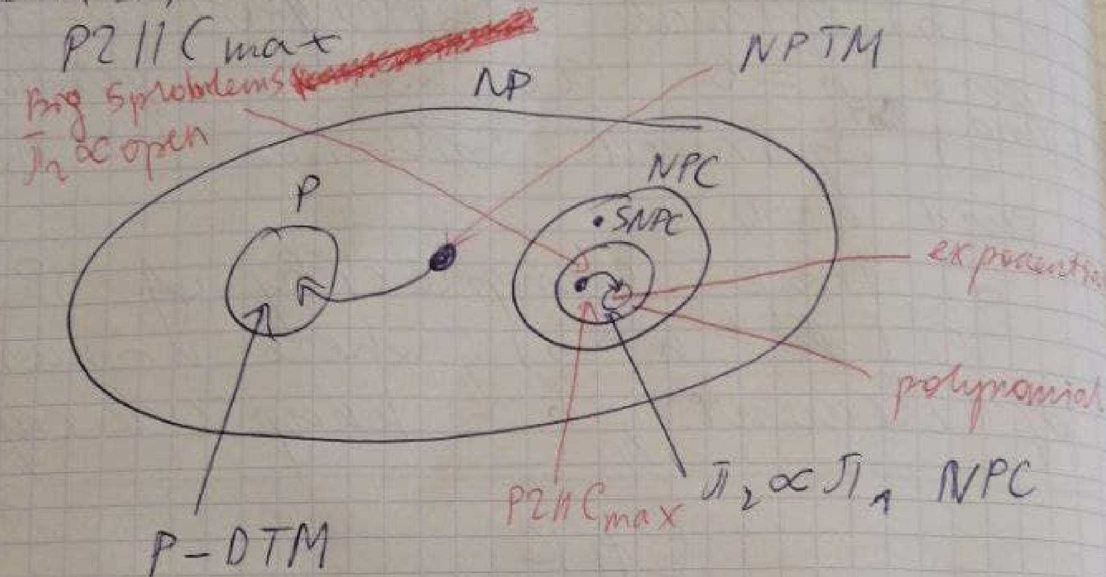
c) optimal solution for $b=5$ and $n=4$: 8.

d) time complexity of DP:
$O(n * b)$, where n - number of elements; b - knapsack capacity.

CAR

SOFYA ARSENYUK,
150284

ex. 5.



NP-complete:

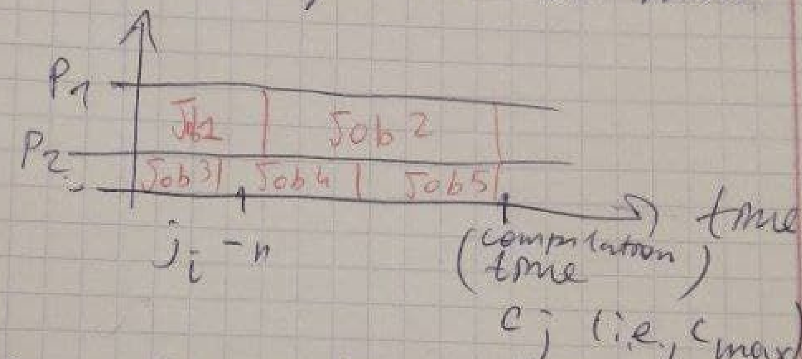
1) weakly NPC:
   - $P_2||C_{max}$
   - Knapsack problem
   - Set Partition
2) strongly NPC:
   - TSP
   - Big 5 problems: (except Set Partition)
     - Vertex Cover
     - 3D-Matching
     - Clique Problem
     - Hamiltonian Circuit

SOFYA AKSENYUK

150284

We can transform $P_2||C_{max}$ problem (multiprocessor scheduling) into set partition problem:

i.e., we are given, for ex., two processors (machines) and some number of jobs for them to accomplish in time



And the question: is it possible to accomplish these jobs by c_{max} time (decision problem: Y/N).

Since, DP is hard, we can conclude NP-hardness of $P_2||C_{max}$.

BUS