* 1. Two languages $L1$ and $L2$ over the alphabet {0, 1} are given. Words of $L1$ are described by the expression 0*1*0*1*0* and words of $L2$ are described by the expression (01+10)*. Find

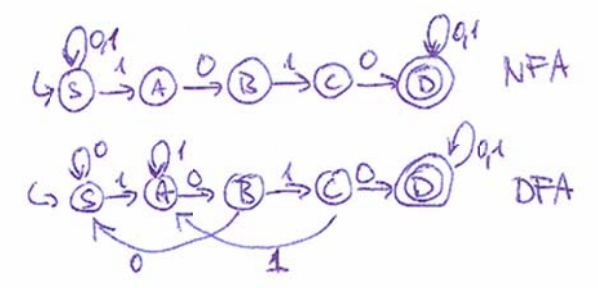
a) the shortest word in the intersection $L1 \cap L2$,
b) the longest word in the intersection $L1 \cap L2$,
c) the shortest word which is in $L1$ and is not in $L2$,
d) the shortest word which is in $L2$ and is not in $L1$,
e) the shortest word which is neither in $L1$ nor in $L2$.

a) 01 and 10
b) 01100110
c) 0 and 1
d) 010101 and 101010
e) 10101

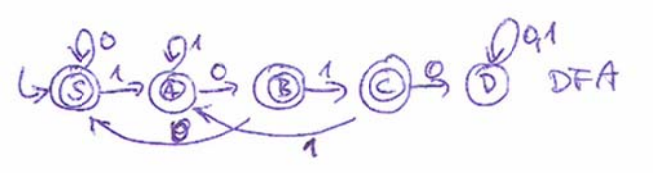
* 2. Draw a transition diagram of an automaton which accepts all words over the alphabet {0,1} which

a) contain the substring 1010 at least once
b) do not contain the substring 1010
c) contain the substring 1010 exactly once
d) contain the substring 1010 at most twice

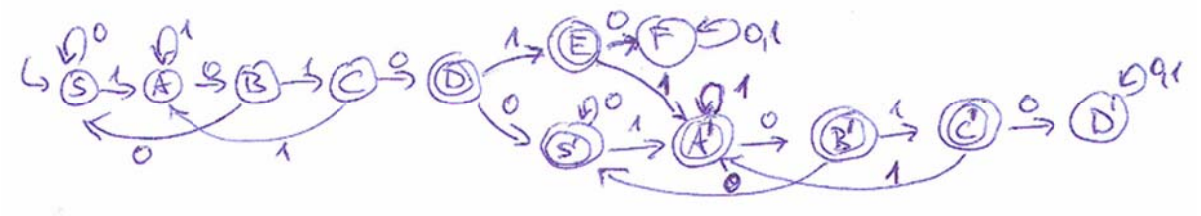
a)



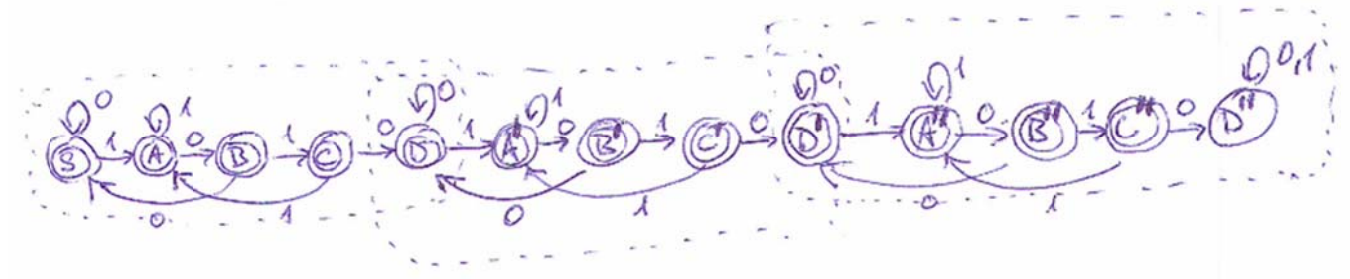
b)



c) States E and F "take care" of case when two 1010's oferlaf forming a single 6-character substring 101010.



d) Dotted lines delimit the parts which intuitively correspond to the automata built in a) and b)



* 3. Write a regular expression describing a language over the alphabet {0, 1} which is maximum possible and

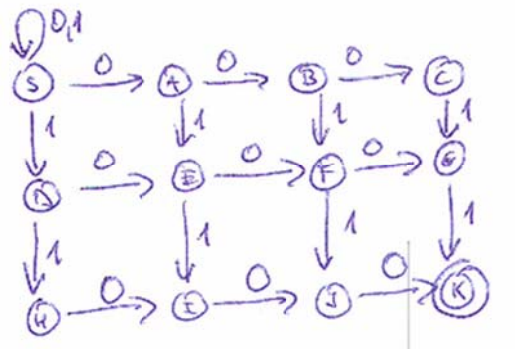
a) which words contain only 0's,
b) each word of which contains exactly one 1,
c) each word of which contains at least one 1,
d) each word of which contains at least two 1,
e) each word of which contains even number of 1's,
f) each word of which contains odd number of 1's.

a) 00*
b) 0*10*
c) 0*1(0+1)* or (0+1)*10* or (0+1)*1(0+1)*
d) 0*10*1(0+1)* or (0+1)*10*10*

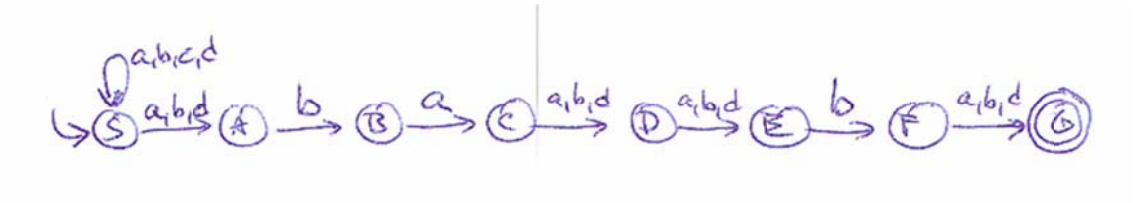
e) (0*10*1)*0* or 0*(10*10*)*
f) (0*10*1)*0*10* or 0*10*(10*10*)*

* 4. Construct a NFA over alphabet {0, 1, 2} which detects in a text all substrings containing exactly three 0's and exactly two 1's.

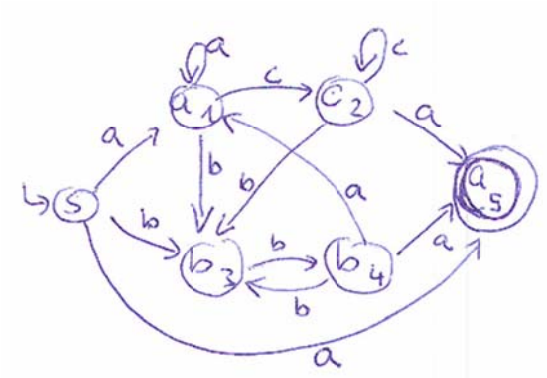
Note the absence of any self-loop in state K. Typically, some students miss the idea that the automaton is a search automaton, thus when it is in the final state it just reports that it has found the substring. The particular branch of the computation (remember, this is NFA performing *many* branches of computation) then ends.



* 5. Construct a NFA over the alphabet {*a*, *b*, *c*, *d*} which detects in a text all occurences of the string in the format #*ba*##*b*#. The symbol # represents exactly one (and any one) symbol from the set {*a*, *b*, *d*}. The automaton must be able to process correctly a text of any length.



6. Construct an automaton over alphabet A = {a,b,c}, which will detect in a text all words characterized by the regular expression R = (ac* + bb)*a.



7. Consider the alphabet A= {*a*, *b*, *c*, ..., *z*}. Let us define the order of symbol *a* to be 1, the order of symbol *b* to be 2 etc, the order of symbol *z* to be 26. We say that a word over A is ordered if for any of its character χ holds that the order of every character which appears later in the word than χ is higher than order of χ. Construct a NFA which detects in a text all occurences of all ordered words.

Create start state S0. For alphabet symbols *a*, *b*, *c*, ..., *z* create states S*a*, S*b*, S*c*, ..., S*z*, that is 27 states in total. Consider the symbols *a*, *b*, *c*, ..., *z* to be formal labels of those states.

Add transitions from S0 to all other states and transitions from each state S*a*, S*b*, S*c*, ..., S*y* to all states labeled with higher order of character. This will yield 26 + 25 + 24 + ... + 2 + 1 = 351 transitions. Label each transition by the character which is the label of the transition target state.
Add self-loop at S0 and label it by all characters *a*, *b*, *c*, ..., *z*. Mark all states S*a*, S*b*, S*c*, ..., S*z* as final.
Needless to say, although the automaton construction is correct, the automaton itself is virtually useless. By definition, each substring of length one is an ordered word. The automaton will thus detect an ordered word at each position of the input text.

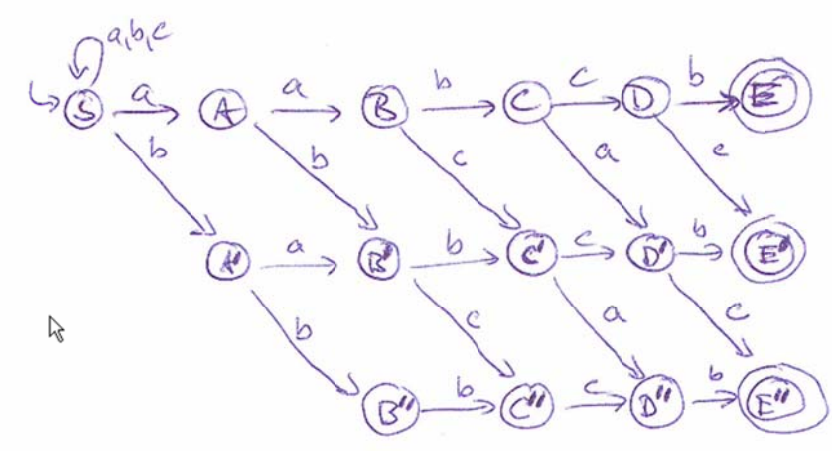
8. Construct a NFA over alphabet {0, 1, 2} which detects in a text all substrings in which the number of symbols 0, 1 and 2 is the same.

Such finite automaton does not exist. To compare the number of 0's and 1's in any possible word, it would need infinitely many states, each of which would register a particular numbers of (unmatched by 1's) 0's read so far. Its non-existence is a consequence of the fundamental theorem called pumping lemma for regular languages. Although we do not present/recall the lemma in Algorithms course, we expect the better students of the course to be familiar with the notion that any language which recognition requires counting unlimited number of occurences of some symbols/substringsc is not regular. A typical example of such impossibility is the language of all well-formed parenthesis expressions or just matching parentheses
https://en.wikipedia.org/wiki/Context-free_grammar#Well-formed_parentheses
https://en.wikipedia.org/wiki/Context-free_grammar#Matching_pairs

9. We would like to construct a NFA over the alphabet {0, 1} which accepts the language of all words which are certificates of some undirected tree. Decide whether such construction is possible and if it is possible describe it.

As each certificate represents a string of well-formed parentheses (0 and 1 correspond to opening and closing parenthesis, respectively), the same reasoning as in the previous problem applies to this problem.

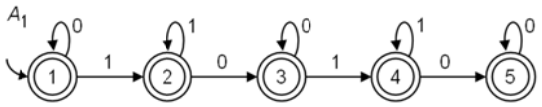
10. Consider the alphabet A = {*a*, *b*, *c*}. Operation ROT chooses an arbitrary character *x* in a given string and substitutes it by a character which immediately follows after x in A. When x is the last character in A the ROT operation substitutes it by the first character in A. Construct a NFA which detects in a text all such substrings which can be obtained from the pattern *aabcb* by applying operation ROT at most twice.



11. Decide whether two given regular expression represent the same regular language.
a) $(01+0)^*0$ b) $0(10+0)^*$

They do. Both contain all finite strings consisting of just zeros. Each string in both languages strats with zero and ends with zero. Any string which does not contain substring 11 is a word in both languages.

12. Characterize informally the language over the alphabet {0, 1} accepted by the given automaton. Write a regular expression describing this language.



The language consists of strings each of which contains at most two contiguous substrings of 1's.
It is the language $L1$ from the problem 1., described by regular expression 0*1*0*1*0*.

13. Write a regular expression which describes maximum set M of strings over alphabet $\{a, b, c\}$. It holds for M:
a) Each string in M begins and starts with symbol b.
b) Each string in M contains exactly one symbol c .
c) No string in M contains symbol a at odd position. Positions are indexed from 1.

a) b(a+b+c)*b
b) (a+b)*c(a+b)*
c) ((b+c)(a+b+c))*