Gradiance Online Accelerated Learning

Zayd

- Home Page
- Assignments Due
- Progress Report
- Handouts
- Tutorials
- Homeworks
- Lab Projects
- Log Out

Help

Submission number:	81859
Submission certificate:	IE126972
Submission time:	2014-04-30 03:58:23 PST (GMT - 8:00)

Number of questions:	25
Positive points per question:	4.0
Negative points per question:	0.0
Your score:	80

1. Apply the CYK algorithm to the input ababaa and the grammar:

```
S → AB | BC
A → BA | a
B → CC | b
C → AB | a
```

Compute the table of entries X_{ij} = the set of nonterminals that derive positions *i* through *j*, inclusive, of the string ababaa. Then, identify a true assertion about one of the X_{ij}'s in the list below.

a) $X_{16} = \{S,A,C\}$

b) $X_{15} = \{B\}$

c) $X_{26} = \{S,A\}$

d) $X_{34} = \{C\}$

Answer submitted: **d)**

Your answer is incorrect.

While it is true that C derives positions 3 and 4 of ababaa, i.e., C => AB => aB => ab, there is another variable that also derives ab. The complete CYK algorithm is described in Section 7.4.4 (p. 303).

2. Here are eight simple grammars, each of which generates an infinite language of strings. These strings tend to look like alternating *a*'s and *b*'s, although there are some exceptions, and not all grammars generate all such strings.

1. S → abS | ab
2. S → SS | ab
3. S → aB; B → bS | a
4. S → aB; B → bS | b
5. S → aB; B → bS | ab
6. S → aB | b; B → bS
7. S → aB | a; B → bS
8. S → aB | ab; B → bS

The initial symbol is S in all cases. Determine the language of each of these grammars. Then, find, in the list below, the pair of grammars that define the same language.

a) G1: S → aB, B → bS, B → b
G2: S → aB, B → bS, S → ab

b) G1: S → abS, S → ab
G2: S → aB, B → bS, S → a

c) G1: S → aB, B → bS, B → b
G2: S → aB, B → bS, S → b

d) G1: S → aB, B → bS, B → ab

G2: $S \rightarrow SS$, $S \rightarrow ab$

Answer submitted: **a)**

You have answered the question correctly.

3. Here are seven regular expressions:

1. (0*+10*)*
2. (0+10)*
3. (0*+10)*
4. (0*+1*)*
5. (0+1)*
6. (0+1*0)*
7. (0+1*)*

Determine the language of each of these expressions. Then, find in the list below a pair of equivalent expressions.

a) (0*+10*)* and (0+10)*

b) (0+1)* and (0*+1*)*

c) (0+1*0)* and (0*+1*)*

d) (0+10)* and (0*+1*)*

Answer submitted: **b)**

You have answered the question correctly.

4. Consider the descriptions of the following problems:

1. Problem P_1: Given a set S of positive integers $\{s_1, s_2,..., s_n\}$ and an integer C as input, is there a subset T of S such that the integers in T add up to C?
2. Problem P_2: Given a set S of positive integers $\{s_1, s_2,..., s_n\}$ as input, is there a subset T of S such that the integers in T add up to c_0?. Here c_0 is a positive integer constant.
3. Problem P_3: Given an undirected graph G and an integer K as input, is there a clique in G of size K?
4. Problem P_4: Given an undirected graph G as input does G contain a clique of size m? Here m is a positive integer constant.

Now consider some additional propositions about the above problems (These may be TRUE or FALSE):

1. Proposition F_1: There is an algorithm A_1 that solves problem P_1 in O(nC) time.
2. Proposition F_2: There is an algorithm A_2 that solves problem P_2 in O(n) time.
3. Proposition F_3: P_3 is NP-complete.
4. Proposition F_4: There is an algorithm A_3 that solves problem P_4 in $O(m^2n^m)$ time.

Choose a correct statement from the choices below:

a) If F_1 and F_2 are both TRUE, P_1 is in P and P_2 is in P.

b) P_2 is reducible to P_1.

c) F_4 must be FALSE. Such an algorithm A_3 cannot exist.

d) F_2 must be FALSE. Such an algorithm A_2 cannot exist.

Answer submitted: **c)**

Your answer is incorrect.

Whether something is part of the input or not may make a difference in whether a problem is solvable in polynomial time. The algorithm A_3 is a polynomial time algorithm for P_4.

Some of the issues regarding "size" of inputs are illustrated in Section 10.1.2 (p. 426).

5. Identify from the list below the regular expression that generates all and only the strings over alphabet {0,1} that end in 1.

a) $(0+1)^*1^+1$

b) $(0^*1)?$

c) $(0^*1^+)^+$

d) 1^*0^*1

Answer submitted: **a)**

Your answer is incorrect.

Although this expression generates only strings of 0's and 1's that end in 1, it cannot generate all of those strings. In particular, it cannot generate the string 1 alone. You may wish to review the basic operators of regular expressions and how they fit together, in Section 3.1 (p. 85) as well as the extended "UNIX" operators from Section 3.3.1 (p. 109).

6. Consider the grammar G with start symbol S:

$S \rightarrow bS \mid aA \mid b$
$A \rightarrow bA \mid aB$
$B \rightarrow bB \mid aS \mid a$

Which of the following is a word in L(G)?

a) babbbaaaaaba

b) babbbbaaab

c) ababbb

d) babbbabaaaa

Answer submitted: **d)**

You have answered the question correctly.

7. Let L_1 and L_2 be two languages produced by grammars of a certain type. Let L be the language which is the concatenation of L_1 and L_2. We want to tell for various types of grammars that produce L_1 and L_2 what type is the concatenation L. Choose the triple ($type_1$, $type_2$, $type_3$) so that when the grammar that produces the language L_1 is of $type_1$ and the grammar that produces the language L_2 is of $type_2$, then the grammar that produces the concatenation language L may not be of $type_3$.

Note: A *linear grammar* is a context-free grammar in which no production body has more than one occurrence of one variable. For example, $A \rightarrow 0B1$ or $A \rightarrow 001$ could be productions of a linear grammar, but $A \rightarrow BB$ or $A \rightarrow A0B$ could not. A *linear language* is a language that has at least one linear grammar.

a) (regular,context-free,context-free)

b) (regular,regular,context-free)

c) (regular, linear,regular)

d) (linear,linear,context-free)

Answer submitted: **c)**

You have answered the question correctly.

8. Apply the construction in Figure 3.16 (p. 104) and Figure 3.17 (p. 105) to convert the regular expression $(0+1)^*(0+\varepsilon)$ to an epsilon-NFA. Then, identify the true statement about your epsilon-NFA from the list below:

a) There are 10 states.

b) There are 16 states.

c) There are 4 states with more than one arc in.

d) There are 17 arcs labeled ε.

Answer submitted: **c)**

You have answered the question correctly.

9. If *h* is the homomorphism defined by h(a) = 0 and h(b) = ε, which of the following strings is in $h^{-1}(000)$?

a) baabbbabb

b) babab

c) abbbabaab

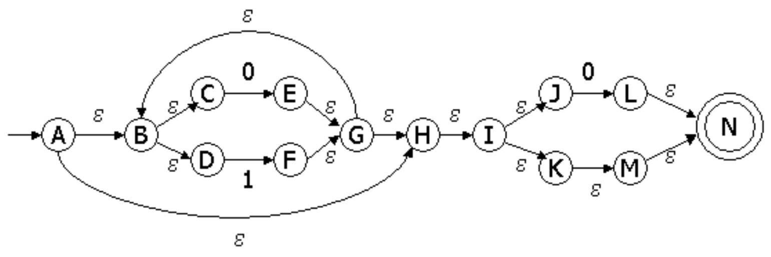
d) abbba

Answer submitted: **c)**

Your answer is incorrect.

h(abbbabaab) = h(a)h(b)h(b)h(b)h(a)h(b)h(a)h(a)h(b) = 0000. Remember that h(b) = ε. Homomorphisms are defined in Section 4.2.3 (p. 140) and inverse homomorphisms in Section 4.2.4 (p. 142).

10. Here is an epsilon-NFA:



Suppose we construct an equivalent DFA by the construction of Section 2.5.5 (p. 77). That is, start with the epsilon-closure of the start state A. For each set of states *S* we construct (which becomes one state of the DFA), look at the transitions from this set of states on input symbol 0. See where those transitions lead, and take the union of the epsilon-closures of all the states reached on 0. This set of states becomes a state of the DFA. Do the same for the transitions out of *S* on input 1. When we have found all the sets of epsilon-NFA states that are constructed in this way, we have the DFA and its transitions.

Carry out this construction of a DFA, and identify one of the states of this DFA (as a subset of the epsilon-NFA's states) from the list below.

a) ABCDEFGHIJKLMN

b) IJKMN

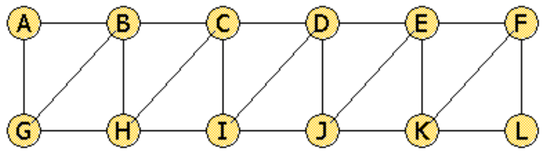
c) ABCDHIJKMN

d) BCDGHIJKMN

Answer submitted: **c)**

You have answered the question correctly.

11. How large can an independent set be in the graph below?



Identify one of the maximal independent sets below.

a) {C,F,G,J}

b) {G,I,K}

c) {A,C,E,J,L}

d) {A,C,I,L}

Answer submitted: **a)**

You have answered the question correctly.

12. Consider the pushdown automaton with the following transition rules:

1. $\delta(q,0,Z_0) = \{(q,XZ_0)\}$
2. $\delta(q,0,X) = \{(q,XX)\}$
3. $\delta(q,1,X) = \{(q,X)\}$
4. $\delta(q,\varepsilon,X) = \{(p,\varepsilon)\}$
5. $\delta(p,\varepsilon,X) = \{(p,\varepsilon)\}$
6. $\delta(p,1,X) = \{(p,XX)\}$
7. $\delta(p,1,Z_0) = \{(p,\varepsilon)\}$

From the ID $(p,1101,XXZ_0)$, which of the following ID's can NOT be reached?

a) (p,ε,XZ_0)

b) $(p,01,XXXZ_0)$

c) $(p,101,XXXZ_0)$

d) $(p,01,XXXXZ_0)$

Answer submitted: **a)**

You have answered the question correctly.

13. Which of the following pairs of grammars define the same language?

a) G_1: S → AB|a, A → b
G_2: S → a

b) G_1: S → SaScSa|aca|ε
G_2: S → SaSAaS|aca, A → cS|ε

c) G_1: S → AB|a, A → b, B → b
G_2: S → a

d) G_1: S → SaScSaS|aca|ε
G_2: S → SaBaS|aca, B → cS|ε

Answer submitted: **a)**

You have answered the question correctly.

14. For the purpose of this question, we assume that all languages are over input alphabet {0,1}. Also, we assume that a Turing machine can have any fixed number of tapes.

Sometimes restricting what a Turing machine can do does not affect the class of languages that can be recognized --- the restricted Turing machines can still be designed to accept any recursively enumerable language. Other restrictions limit what languages the Turing machine can accept. For example, it might limit the languages to some subset of the recursive languages, which we know is smaller than the recursively enumerable languages. Here are some of the possible restrictions:

1. Limit the number of states the TM may have.
2. Limit the number of tape symbols the TM may have.
3. Limit the number of times any tape cell may change.
4. Limit the amount of tape the TM may use.
5. Limit the number of moves the TM may make.
6. Limit the way the tape heads may move.

Consider the effect of limitations of these types, perhaps in pairs. Then, from the list below, identify the combination of restrictions that allows the restricted form of Turing machine to accept all recursively enumerable languages.

a) Allow tape heads to move only right or remain stationary on their tape.

b) Allow the TM to run for only 2^n moves when the input is of length n.

c) Limit the number of states to 1,000,000 and the number of tape symbols to 1,000,000.

d) Allow the TM to use only n^3 tape cells when the input is of length n.

Answer submitted: **d)**

Your answer is incorrect.

Can you show that in this case, every language accepted by such a TM must be recursive? Design another TM that simulates the first, and if the first doesn't accept before repeating an ID, then reject. Recall the definition of a recursive Turing machine in Section 9.2.1 (p. 383). Also, you may wish to examine Section 11.2.1 (p. 487), which talks about polynomial-space-bounded Turing machines.

15. Here is a context-free grammar:

```
S → AB | CD
A → BG | 0
B → AD | ε
C → CD | 1
D → BB | E
E → AF | B1
F → EG | 0C
G → AG | BD
```

Find all the nullable symbols (those that derive ε in one or more steps). Then, identify the true statement from the list below.

a) S is nullable.

b) C is nullable.

c) G is not nullable.

d) B is not nullable.

Answer submitted: **a)**

You have answered the question correctly.

16. The Turing machine M has:

- States q and p; q is the start state.
- Tape symbols 0, 1, and B; 0 and 1 are input symbols, and B is the blank.
- The following next-move function:

State	Tape	Move
	Symbol	
q	0	(q,0,R)
q	1	(p,0,R)
q	B	(q,B,R)
p	0	(q,0,L)
p	1	none (halt)
p	B	(q,0,L)

Your problem is to describe the property of an input string that makes M halt. Identify a string that makes M halt from the list below.

a) 0000

b) 00100

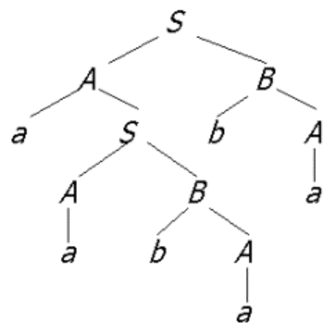
c) 1001

d) 0011

Answer submitted: **d)**

You have answered the question correctly.

17. The parse tree below represents a rightmost derivation according to the grammar `S → AB, A → aS|a, B → bA`.



Which of the following is a right-sentential form in this derivation?

a) abaAbA

b) aSbA

c) abaAba

d) aABba

Answer submitted: **d)**

You have answered the question correctly.

18. The classes of languages P and NP are closed under certain operations, and not closed under others, just like classes such as the regular languages or context-free languages have closure properties. Decide whether P and NP are closed under each of the following operations.

1. Union.
2. Intersection.
3. Intersection with a regular language.
4. Concatenation.
5. Kleene closure (star).
6. Homomorphism.
7. Inverse homomorphism.

Then, select from the list below the true statement.

a) P is not closed under union.

b) P is not closed under Kleene closure.

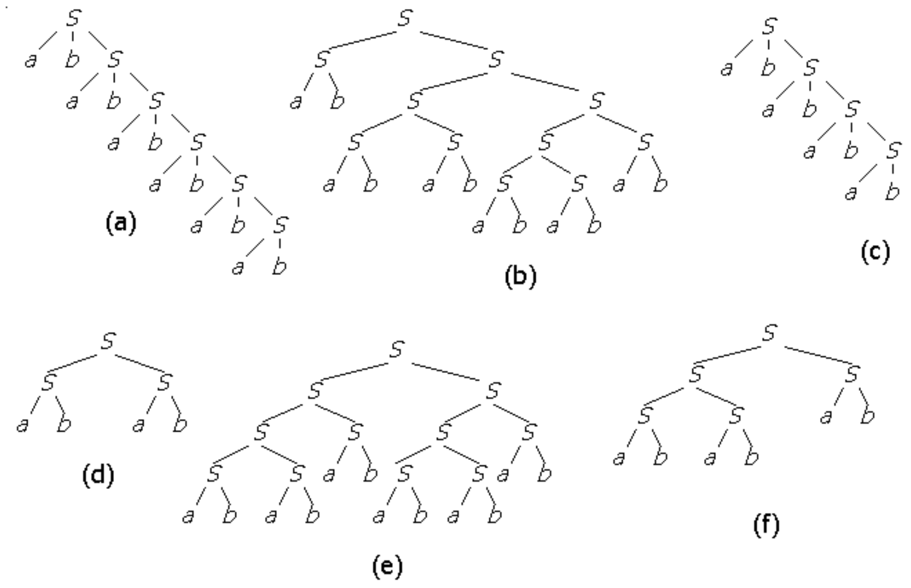
c) P is not closed under intersection.

d) P is closed under Kleene closure.

Answer submitted: **d)**

You have answered the question correctly.

19. Consider the grammar: S → SS, S → ab. Identify in the list below the one set of parse trees which includes a tree that is NOT a parse tree of this grammar?



a) {(b),(e)}

b) {(d),(f)}

c) {(e)}

d) {(a),(d)}

Answer submitted: **d)**

You have answered the question correctly.

20. Design the minimum-state DFA that accepts all and only the strings of 0's and 1's that have 110 as a substring. To verify that you have designed the correct automaton, we will ask you to identify the true statement in a list of choices. These choices will involve:

1. The number of *loops* (transitions from a state to itself).
2. The number of transitions into a state (including loops) on input 1.
3. The number of transitions into a state (including loops) on input 0.

Count the number of transitions into each of your states ("in-transitions") on input 1 and also on input 0. Count the number of loops on input 1 and on input 0. Then, find the true statement in the following list.

a) There is one state that has three in-transitions on input 0.

b) There are three states that have one in-transition on input 1.

c) There is one loop on input 1 and one loop on input 0.

d) There are two states that have one in-transition on input 1.

Answer submitted: **d)**

You have answered the question correctly.

21. Consider the grammars:

G_1:S → AB | a | abC, A → b, C → abC | c

G_2:S → a | b | cC, C → cC | c

These grammars do not define the same language. To prove, we use a string that is generated by one but not by the other grammar. Which of the following strings can be used for this proof?

a) a
b) ababccc
c) ababababc
d) ababa

Answer submitted: **c)**

You have answered the question correctly.

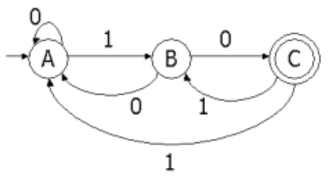
22. The Boolean expression wxyz+u+v is equivalent to an expression in 3-CNF (a product of clauses, each clause being the sum of exactly three literals). Find the simplest such 3-CNF expression and then identify one of its clauses in the list below. Note: -*e* denotes the negation of *e*. Also note: we are looking for an expression that involves only u, v, w, x, y, and z, no other variables. Not all boolean expressions can be converted to 3-CNF without introducing new variables, but this one can.

a) (w+y+v)
b) (y+u+v)
c) (u+v+-w)
d) (w+z+v)

Answer submitted: **b)**

You have answered the question correctly.

23. The following nondeterministic finite automaton:



$$\begin{tikzcd}
\arrow[r] & A \arrow[loop above, "0"] \arrow[r, "1"] & B \arrow[l, bend left, "0"] \arrow[r, "0"] & C \arrow[l, bend left, "1"] \arrow[ll, bend left=60, "1"]
\end{tikzcd}$$

accepts which of the following strings?

a) 00010111
b) 010111
c) 10001010
d) 00110100

Answer submitted: **c)**

You have answered the question correctly.

24. Here is a context-free grammar G:

```
S → AB
A → 0A1 | 2
B → 1B | 3A
```

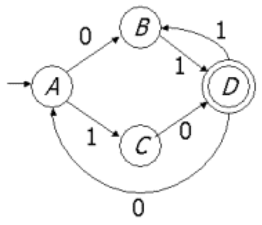
Which of the following strings is in L(G)?

a) 000021130011
b) 000211132
c) 0021131100211
d) 0021113002111

Answer submitted: **b)**

You have answered the question correctly.

25. The finite automaton below:



accepts no word of length zero, no word of length one, and only two words of length two (01 and 10). There is a fairly simple recurrence equation for the number N(k) of words of length k that this automaton accepts. Discover this recurrence and demonstrate your understanding by identifying the correct value of N(k) for some particular k. Note: the recurrence does not have an easy-to-use closed form, so you will have to compute the first few values by hand. You do not have to compute N(k) for any k greater than 14.

a) N(11) = 76

b) N(14) = 114

c) N(14) = 16

d) N(12) = 1366

Answer submitted: **b)**

You have answered the question correctly.