BILL START RECORDING

Midterm Solutions

Problem 2a

There is an algorithm that will, given an NFA M of size n, return a DFA for $L(M)$ of size FILLIN.

Problem 2a

There is an algorithm that will, given an NFA M of size n, return a DFA for $L(M)$ of size FILLIN.

PUT ANSWER HERE:
2^n.

Problem 2a

There is an algorithm that will, given an NFA M of size n, return a DFA for $L(M)$ of size FILLIN.

PUT ANSWER HERE:
2^n.

Short Explanation (not needed on midterm): This is the Power set construction.

Problem 2b

There is an algorithm that will, given a DFA M of size n, return a regex for $L(M)$ of size FILLIN.

Problem 2b

There is an algorithm that will, given a DFA M of size n, return a regex for $L(M)$ of size FILLIN.

PUT ANSWER HERE:
$2^{O(n)}$.

Problem 2b

There is an algorithm that will, given a DFA M of size n, return a regex for $L(M)$ of size FILLIN.

PUT ANSWER HERE:
$2^{O(n)}$.

Short Explanation (not needed on midterm): This is the $R(i, j, k)$ construction.

Problem 2b

There is an algorithm that will, given a DFA M of size n, return a regex for $L(M)$ of size FILLIN.

PUT ANSWER HERE:
$2^{O(n)}$.

Short Explanation (not needed on midterm): This is the $R(i, j, k)$ construction.

Grading: We accepted 2^n, $O(2^n)$ even though technically they are not correct.

Problem 2c

There is an algorithm that will, given a regex α of size n, return an NFA for $L(\alpha)$ of size FILLIN.

Problem 2c

There is an algorithm that will, given a regex α of size n, return an NFA for $L(\alpha)$ of size FILLIN.

PUT ANSWER HERE:
$O(n)$.

Problem 2c

There is an algorithm that will, given a regex α of size n, return an NFA for $L(\alpha)$ of size FILLIN.

PUT ANSWER HERE:
$O(n)$.
Short Explanation (not needed on midterm): This is done inductively based on the definition of a regex. We use the closure properties of NFA's.

Problem 3a

There is an algorithm that will, given an DFA M of sizes n, return a CFG for $L(M)$ and it will have FILLIN number of rules. (The CFG need not be in Chomsky Normal Form.)

Problem 3a

There is an algorithm that will, given an DFA M of sizes n, return a CFG for $L(M)$ and it will have FILLIN number of rules. (The CFG need not be in Chomsky Normal Form.)

PUT ANSWER HERE:
$O(n)$.

Problem 3a

There is an algorithm that will, given an DFA M of sizes n, return a CFG for $L(M)$ and it will have FILLIN number of rules. (The CFG need not be in Chomsky Normal Form.)

PUT ANSWER HERE:
$O(n)$.
Short Explanation (not needed on midterm): This was the problem I had you look up how to do for HW07.

Problem 3a

There is an algorithm that will, given an DFA M of sizes n, return a CFG for $L(M)$ and it will have FILLIN number of rules. (The CFG need not be in Chomsky Normal Form.)

PUT ANSWER HERE:
$O(n)$.
Short Explanation (not needed on midterm): This was the problem I had you look up how to do for HW07.
Interesting The grammar produces has most of the rules in the form needed for Chomsky Normal Form even though this is not required.

Problem 3a

There is an algorithm that will, given an DFA M of sizes n, return a CFG for $L(M)$ and it will have FILLIN number of rules. (The CFG need not be in Chomsky Normal Form.)

PUT ANSWER HERE:

$O(n)$.

Short Explanation (not needed on midterm): This was the problem I had you look up how to do for HW07.

Interesting The grammar produces has most of the rules in the form needed for Chomsky Normal Form even though this is not required.

The only rules that are not in that form are of the form $A \rightarrow e$.

Problem 3b

There is an algorithm that will, given n, return a Chomsky Normal Form CFG for

$$\{e, a, a^2, \ldots, a^n\}$$

which has FILLIN nonterminals.

Problem 3b

There is an algorithm that will, given n, return a Chomsky Normal Form CFG for

$$\{e, a, a^2, \ldots, a^n\}$$

which has FILLIN nonterminals.

PUT ANSWER HERE:
$O(\log n)$.

Problem 3b

There is an algorithm that will, given n, return a Chomsky Normal Form CFG for

$$\{e, a, a^2, \ldots, a^n\}$$

which has FILLIN nonterminals.

PUT ANSWER HERE:

$O(\log n)$.

Short Explanation (not needed on midterm): This was the problem I had you do on HW07.

Problem 3b

There is an algorithm that will, given n, return a Chomsky Normal Form CFG for

$$\{e, a, a^2, \ldots, a^n\}$$

which has FILLIN nonterminals.

PUT ANSWER HERE:
$O(\log n)$.
Short Explanation (not needed on midterm): This was the problem I had you do on HW07.
Grading Some students wrote $O(n^{1/3})$. This is not correct. I will discuss the issue after Problem 3.

Problem 3c

There is an algorithm that will, given n and some

$$X \subseteq \{e, a, a^2, \ldots, a^n\},$$

returns a Chomsky Normal Form CFG for X which has FILLIN nonterminals. FILLIN depends only on n, e.g $O(n \log n)$ (THIS IS NOT THE ANSWER).

Problem 3c

There is an algorithm that will, given n and some

$$X \subseteq \{e, a, a^2, \ldots, a^n\},$$

returns a Chomsky Normal Form CFG for X which has FILLIN nonterminals. FILLIN depends only on n, e.g $O(n \log n)$ (THIS IS NOT THE ANSWER).

PUT ANSWER HERE:
$O(n^{1/3})$

Problem 3c

There is an algorithm that will, given n and some

$$X \subseteq \{e, a, a^2, \ldots, a^n\},$$

returns a Chomsky Normal Form CFG for X which has FILLIN nonterminals. FILLIN depends only on n, e.g $O(n \log n)$ (THIS IS NOT THE ANSWER).

PUT ANSWER HERE:
$O(n^{1/3})$
Grading This is the lecture **CFG's for Finite Unary Languages**.

Problem 3c

There is an algorithm that will, given n and some

$$X \subseteq \{e, a, a^2, \ldots, a^n\},$$

returns a Chomsky Normal Form CFG for X which has FILLIN nonterminals. FILLIN depends only on n, e.g $O(n \log n)$ (THIS IS NOT THE ANSWER).

PUT ANSWER HERE:
$O(n^{1/3})$
Grading This is the lecture **CFG's for Finite Unary Languages**. **Which version of Chomsky to use?** In the lecture I allowed $A \rightarrow BCD$. Normally I do not. Students asked which to use.

Problem 3c

There is an algorithm that will, given n and some

$$X \subseteq \{e, a, a^2, \ldots, a^n\},$$

returns a Chomsky Normal Form CFG for X which has FILLIN nonterminals. FILLIN depends only on n, e.g $O(n \log n)$ (THIS IS NOT THE ANSWER).

PUT ANSWER HERE:
$O(n^{1/3})$
Grading This is the lecture **CFG's for Finite Unary Languages**. **Which version of Chomsky to use?** In the lecture I allowed $A \rightarrow BCD$. Normally I do not. Students asked which to use. YOU CAN USE $A \rightarrow BCD$ since all I am using O-notation.

Problem 3c

There is an algorithm that will, given n and some

$$X \subseteq \{e, a, a^2, \ldots, a^n\},$$

returns a Chomsky Normal Form CFG for X which has FILLIN nonterminals. FILLIN depends only on n, e.g $O(n \log n)$ (THIS IS NOT THE ANSWER).

PUT ANSWER HERE:
$O(n^{1/3})$
Grading This is the lecture **CFG's for Finite Unary Languages**.
Which version of Chomsky to use? In the lecture I allowed $A \rightarrow BCD$. Normally I do not. Students asked which to use.
YOU CAN USE $A \rightarrow BCD$ since all I am using O-notation.
I will discuss the contrast between 3b and 3c on the next slide.

CFG's for Finite Unary Languages: Number of NTs

Here is what I proved.

CFG's for Finite Unary Languages: Number of NTs

Here is what I proved.
1) There is a CNFG for $A = \{e, a, a^2, \ldots, a^n\}$ that uses $O(n^{1/3})$ NTs. This CFGN is **not** optimal for number of NTs; **however** it has a nice property:

CFG's for Finite Unary Languages: Number of NTs

Here is what I proved.
1) There is a CNFG for $A = \{e, a, a^2, \ldots, a^n\}$ that uses $O(n^{1/3})$ NTs. This CFGN is **not** optimal for number of NTs; **however** it has a nice property:
For every $w \in A$ there is a rule that, if removed, will remove ONLY w from the set of strings generated.

CFG's for Finite Unary Languages: Number of NTs

Here is what I proved.
1) There is a CNFG for $A = \{e, a, a^2, \ldots, a^n\}$ that uses $O(n^{1/3})$ NTs. This CFGN is **not** optimal for number of NTs; **however** it has a nice property:
For every $w \in A$ there is a rule that, if removed, will remove ONLY w from the set of strings generated.

2) USING (1) I showed that for **every** $X \subseteq \{e, a, a^2, \ldots, a^n\}$ there is a CNFG that generates X and has $O(n^{1/3})$ NTs.

CFG's for Finite Unary Languages: Number of NTs

Here is what I proved.
1) There is a CNFG for $A = \{e, a, a^2, \ldots, a^n\}$ that uses $O(n^{1/3})$ NTs. This CFGN is **not** optimal for number of NTs; **however** it has a nice property:
For every $w \in A$ there is a rule that, if removed, will remove ONLY w from the set of strings generated.

2) USING (1) I showed that for **every** $X \subseteq \{e, a, a^2, \ldots, a^n\}$ there is a CNFG that generates X and has $O(n^{1/3})$ NTs.

3) Can the $O(n^{1/3})$ be improved. No: $\exists\ X \subseteq \{e, a, a^2, \ldots, a^n\}$ such that **every** CNFG for X has $\Omega(n^{1/3})$ NTs

CFG's for Finite Unary Languages: Number of NTs

Here is what I proved.

1) There is a CNFG for $A = \{e, a, a^2, \ldots, a^n\}$ that uses $O(n^{1/3})$ NTs. This CFGN is **not** optimal for number of NTs; **however** it has a nice property:

For every $w \in A$ there is a rule that, if removed, will remove ONLY w from the set of strings generated.

2) USING (1) I showed that for **every** $X \subseteq \{e, a, a^2, \ldots, a^n\}$ there is a CNFG that generates X and has $O(n^{1/3})$ NTs.

3) Can the $O(n^{1/3})$ be improved. No: $\exists\ X \subseteq \{e, a, a^2, \ldots, a^n\}$ such that **every** CNFG for X has $\Omega(n^{1/3})$ NTs

4) Is there a better CFGN for the original language A? On HW07 we showed YES, there is a CFGN for A with $O(\log n)$ NTs. This CNFG does not have any nice property.

CFG's for Finite Unary Languages: Number of NTs (More)

CFG's for Finite Unary Languages: Number of NTs (More)

1) $\forall\ X \subseteq \{e, a, a^2, \ldots, a^n\}\ \exists$ CNFG that generates X and has $O(n^{1/3})$ NTs.

CFG's for Finite Unary Languages: Number of NTs (More)

1) $\forall X \subseteq \{e, a, a^2, \ldots, a^n\}$ $\exists$ CNFG that generates X and has $O(n^{1/3})$ NTs.

2) $\exists X \subseteq \{e, a, a^2, \ldots, a^n\}$ $\forall$ CNFG that generates X have $\Omega(n^{1/3})$ NTs.

CFG's for Finite Unary Languages: Number of NTs (More)

1) $\forall X \subseteq \{e, a, a^2, \ldots, a^n\}$ $\exists$ CNFG that generates X and has $O(n^{1/3})$ NTs.

2) $\exists X \subseteq \{e, a, a^2, \ldots, a^n\}$ $\forall$ CNFG that generates X have $\Omega(n^{1/3})$ NTs.

3) Some X use far less NT's.

CFG's for Finite Unary Languages: Number of NTs (More)

1) $\forall\ X \subseteq \{e, a, a^2, \ldots, a^n\}\ \exists$ CNFG that generates X and has $O(n^{1/3})$ NTs.

2) $\exists\ X \subseteq \{e, a, a^2, \ldots, a^n\}\ \forall$ CNFG that generates X have $\Omega(n^{1/3})$ NTs.

3) Some X use far less NT's.

4) **Question** Does $\exists\ X \subseteq \{e, a, a^2, \ldots, a^n\}$ such that

CFG's for Finite Unary Languages: Number of NTs (More)

1) $\forall\ X \subseteq \{e, a, a^2, \ldots, a^n\}\ \exists$ CNFG that generates X and has $O(n^{1/3})$ NTs.

2) $\exists\ X \subseteq \{e, a, a^2, \ldots, a^n\}\ \forall$ CNFG that generates X have $\Omega(n^{1/3})$ NTs.

3) Some X use far less NT's.

4) **Question** Does $\exists\ X \subseteq \{e, a, a^2, \ldots, a^n\}$ such that
a) $\exists$ CNFG that generates X and has $O(n^{1/4})$ NTs.

CFG's for Finite Unary Languages: Number of NTs (More)

1) $\forall\ X \subseteq \{e, a, a^2, \ldots, a^n\}\ \exists$ CNFG that generates X and has $O(n^{1/3})$ NTs.

2) $\exists\ X \subseteq \{e, a, a^2, \ldots, a^n\}\ \forall$ CNFG that generates X have $\Omega(n^{1/3})$ NTs.

3) Some X use far less NT's.

4) **Question** Does $\exists\ X \subseteq \{e, a, a^2, \ldots, a^n\}$ such that
a) $\exists$ CNFG that generates X and has $O(n^{1/4})$ NTs.

b) $\forall$ CNFG that generates X have $\Omega(n^{1/4})$ NTs.

CFG's for Finite Unary Languages: Number of NTs (More)

1) $\forall X \subseteq \{e, a, a^2, \ldots, a^n\}$ $\exists$ CNFG that generates X and has $O(n^{1/3})$ NTs.

2) $\exists X \subseteq \{e, a, a^2, \ldots, a^n\}$ $\forall$ CNFG that generates X have $\Omega(n^{1/3})$ NTs.

3) Some X use far less NT's.

4) **Question** Does $\exists X \subseteq \{e, a, a^2, \ldots, a^n\}$ such that
a) $\exists$ CNFG that generates X and has $O(n^{1/4})$ NTs.

b) $\forall$ CNFG that generates X have $\Omega(n^{1/4})$ NTs.

Vote YES there is such a lang, NO there is no such lang, Unknown to Science, Unknown to Bill.

Problem 4

Let $L_1 = \{a^i : i \neq 109\}$.
Want NFA for L_1 that has much less than 109 states.
Don't want NFA. Want x, y, t primes,and number of states.

Problem 4

Let $L_1 = \{a^i : i \neq 109\}$.
Want NFA for L_1 that has much less than 109 states.
Don't want NFA. Want x, y, t primes,and number of states.
I have talked about this kind of problem **a lot**!

Problem 4

Let $L_1 = \{a^i : i \neq 109\}$.
Want NFA for L_1 that has much less than 109 states.
Don't want NFA. Want x, y, t primes,and number of states.
I have talked about this kind of problem **a lot**!
Need x, y REL PRIME: $xy - x - y + 1 \leq 109$ (but close).

Problem 4

Let $L_1 = \{a^i : i \neq 109\}$.
Want NFA for L_1 that has much less than 109 states.
Don't want NFA. Want x, y, t primes,and number of states.
I have talked about this kind of problem **a lot**!
Need x, y REL PRIME: $xy - x - y + 1 \leq 109$ (but close).
Take $x = 11$ and $y = 12$: $xy - x - y = 109$ CANNOT be written as a sum of 11's and 12's but everything larger can. So $x = 11$, $y = 12$, $t = 0$. (There are other x, y, t that work.)

Problem 4

Let $L_1 = \{a^i : i \neq 109\}$.
Want NFA for L_1 that has much less than 109 states.
Don't want NFA. Want x, y, t primes,and number of states.
I have talked about this kind of problem **a lot**!
Need x, y REL PRIME: $xy - x - y + 1 \leq 109$ (but close).
Take $x = 11$ and $y = 12$: $xy - x - y = 109$ CANNOT be written as a sum of 11's and 12's but everything larger can. So $x = 11$, $y = 12$, $t = 0$. (There are other x, y, t that work.)
We need PRIMES that had product ≥ 110. $2 \times 3 \times 5 \times 7 = 210$.

Problem 4

Let $L_1 = \{a^i : i \neq 109\}$.
Want NFA for L_1 that has much less than 109 states.
Don't want NFA. Want x, y, t primes,and number of states.
I have talked about this kind of problem **a lot**!
Need x, y REL PRIME: $xy - x - y + 1 \leq 109$ (but close).
Take $x = 11$ and $y = 12$: $xy - x - y = 109$ CANNOT be written as a sum of 11's and 12's but everything larger can. So $x = 11$, $y = 12$, $t = 0$. (There are other x, y, t that work.)
We need PRIMES that had product ≥ 110. $2 \times 3 \times 5 \times 7 = 210$.

GIVE x HERE: 11,

Problem 4

Let $L_1 = \{a^i : i \neq 109\}$.
Want NFA for L_1 that has much less than 109 states.
Don't want NFA. Want x, y, t primes,and number of states.
I have talked about this kind of problem **a lot**!
Need x, y REL PRIME: $xy - x - y + 1 \leq 109$ (but close).
Take $x = 11$ and $y = 12$: $xy - x - y = 109$ CANNOT be written as a sum of 11's and 12's but everything larger can. So $x = 11$, $y = 12$, $t = 0$. (There are other x, y, t that work.)
We need PRIMES that had product ≥ 110. $2 \times 3 \times 5 \times 7 = 210$.

GIVE x HERE: 11, GIVE y HERE: 12,

Problem 4

Let $L_1 = \{a^i : i \neq 109\}$.
Want NFA for L_1 that has much less than 109 states.
Don't want NFA. Want x, y, t primes,and number of states.
I have talked about this kind of problem **a lot**!
Need x, y REL PRIME: $xy - x - y + 1 \leq 109$ (but close).
Take $x = 11$ and $y = 12$: $xy - x - y = 109$ CANNOT be written as a sum of 11's and 12's but everything larger can. So $x = 11$, $y = 12$, $t = 0$. (There are other x, y, t that work.)
We need PRIMES that had product ≥ 110. $2 \times 3 \times 5 \times 7 = 210$.

GIVE x HERE: 11, GIVE y HERE: 12,GIVE t HERE: 0

Problem 4

Let $L_1 = \{a^i : i \neq 109\}$.
Want NFA for L_1 that has much less than 109 states.
Don't want NFA. Want x, y, t primes,and number of states.
I have talked about this kind of problem **a lot**!
Need x, y REL PRIME: $xy - x - y + 1 \leq 109$ (but close).
Take $x = 11$ and $y = 12$: $xy - x - y = 109$ CANNOT be written as a sum of 11's and 12's but everything larger can. So $x = 11$, $y = 12$, $t = 0$. (There are other x, y, t that work.)
We need PRIMES that had product ≥ 110. $2 \times 3 \times 5 \times 7 = 210$.

GIVE x HERE: 11, GIVE y HERE: 12,GIVE t HERE: 0
GIVE THE PRIMES HERE: $2, 3, 5, 7$.

Problem 4

Let $L_1 = \{a^i : i \neq 109\}$.
Want NFA for L_1 that has much less than 109 states.
Don't want NFA. Want x, y, t primes,and number of states.
I have talked about this kind of problem **a lot**!
Need x, y REL PRIME: $xy - x - y + 1 \leq 109$ (but close).
Take $x = 11$ and $y = 12$: $xy - x - y = 109$ CANNOT be written as a sum of 11's and 12's but everything larger can. So $x = 11$, $y = 12$, $t = 0$. (There are other x, y, t that work.)
We need PRIMES that had product ≥ 110. $2 \times 3 \times 5 \times 7 = 210$.

GIVE x HERE: 11, GIVE y HERE: 12,GIVE t HERE: 0
GIVE THE PRIMES HERE: $2, 3, 5, 7$.
GIVE THE NUMBER OF STATES HERE:
$1 + 11 + 12 + 2 + 3 + 5 + 7 = 41$.

Problem 4

Let $L_1 = \{a^i : i \neq 109\}$.
Want NFA for L_1 that has much less than 109 states.
Don't want NFA. Want x, y, t primes,and number of states.
I have talked about this kind of problem **a lot**!
Need x, y REL PRIME: $xy - x - y + 1 \leq 109$ (but close).
Take $x = 11$ and $y = 12$: $xy - x - y = 109$ CANNOT be written as a sum of 11's and 12's but everything larger can. So $x = 11$, $y = 12$, $t = 0$. (There are other x, y, t that work.)
We need PRIMES that had product ≥ 110. $2 \times 3 \times 5 \times 7 = 210$.

GIVE x HERE: 11, GIVE y HERE: 12,GIVE t HERE: 0
GIVE THE PRIMES HERE: $2, 3, 5, 7$.
GIVE THE NUMBER OF STATES HERE:
$1 + 11 + 12 + 2 + 3 + 5 + 7 = 41$.
Grading NEED to have x, y rel prime. NEED to have the primes are primes (some students included 1- we did not penalize for that on the midterm but will on the final).

Point of Problem 4 That Got Lost

Was going to ask about 109 and 108 and have you find that

Point of Problem 4 That Got Lost

Was going to ask about 109 and 108 and have you find that
The NFA for 109 and **less** *states than the NFA for 108.*

Point of Problem 4 That Got Lost

Was going to ask about 109 and 108 and have you find that
The NFA for 109 and **less** *states than the NFA for 108.*

Interesting that the number of states is not strictly increasing.

Point of Problem 4 That Got Lost

Was going to ask about 109 and 108 and have you find that *The NFA for 109 and* **less** *states than the NFA for 108.*

Interesting that the number of states is not strictly increasing.

While a good point to make, would make the exam longer than I want.

Point of Problem 4 That Got Lost

Was going to ask about 109 and 108 and have you find that
The NFA for 109 and **less** *states than the NFA for 108.*

Interesting that the number of states is not strictly increasing.

While a good point to make, would make the exam longer than I want.

Students might have correct solutions for 109 that DO have a tail.

Problem 5

For this problem $\Sigma = \{a, b\}$.

Problem 5

For this problem $\Sigma = \{a, b\}$.
a) Give an algorithm that will do the following:

Problem 5

For this problem $\Sigma = \{a, b\}$.
a) Give an algorithm that will do the following:
Given a string w*, outputs a Chomsky Normal Form CFG* G *such that* $L(G) = \{w\}$ *(so* G *generates* w *and nothing else).* Your algorithm can use DOT DOT DOT.

Problem 5

For this problem $\Sigma = \{a, b\}$.
a) Give an algorithm that will do the following:
Given a string w, outputs a Chomsky Normal Form CFG G such that $L(G) = \{w\}$ (so G generates w and nothing else). Your algorithm can use DOT DOT DOT.
b) f such that your grammar had $O(f(n))$ rules.

Problem 5

For this problem $\Sigma = \{a, b\}$.
a) Give an algorithm that will do the following:
Given a string w, outputs a Chomsky Normal Form CFG G such that $L(G) = \{w\}$ (so G generates w and nothing else). Your algorithm can use DOT DOT DOT.
b) f such that your grammar had $O(f(n))$ rules.
$w = \sigma_1 \cdots \sigma_n$.

Problem 5

For this problem $\Sigma = \{a, b\}$.

a) Give an algorithm that will do the following:

Given a string w, outputs a Chomsky Normal Form CFG G such that $L(G) = \{w\}$ (so G generates w and nothing else). Your algorithm can use DOT DOT DOT.

b) f such that your grammar had $O(f(n))$ rules.

$w = \sigma_1 \cdots \sigma_n$.

$S \to [\sigma_1][\sigma_2 \cdots \sigma_n]$

Problem 5

For this problem $\Sigma = \{a, b\}$.
a) Give an algorithm that will do the following:
Given a string w, outputs a Chomsky Normal Form CFG G such that $L(G) = \{w\}$ (so G generates w and nothing else). Your algorithm can use DOT DOT DOT.
b) f such that your grammar had $O(f(n))$ rules.
$w = \sigma_1 \cdots \sigma_n$.
$S \to [\sigma_1][\sigma_2 \cdots \sigma_n]$
$[\sigma_2 \cdots \sigma_n] \to [\sigma_3 \cdots \sigma_n]$

Problem 5

For this problem $\Sigma = \{a, b\}$.

a) Give an algorithm that will do the following:

Given a string w, outputs a Chomsky Normal Form CFG G such that $L(G) = \{w\}$ (so G generates w and nothing else). Your algorithm can use DOT DOT DOT.

b) f such that your grammar had $O(f(n))$ rules.

$w = \sigma_1 \cdots \sigma_n$.

$S \to [\sigma_1][\sigma_2 \cdots \sigma_n]$

$[\sigma_2 \cdots \sigma_n] \to [\sigma_3 \cdots \sigma_n]$

$[\sigma_3 \cdots \sigma_n] \to [\sigma_4 \cdots \sigma_n]$

$\vdots$

Problem 5

For this problem $\Sigma = \{a, b\}$.

a) Give an algorithm that will do the following:

Given a string w, outputs a Chomsky Normal Form CFG G such that $L(G) = \{w\}$ (so G generates w and nothing else). Your algorithm can use DOT DOT DOT.

b) f such that your grammar had $O(f(n))$ rules.

$w = \sigma_1 \cdots \sigma_n$.

$S \rightarrow [\sigma_1][\sigma_2 \cdots \sigma_n]$

$[\sigma_2 \cdots \sigma_n] \rightarrow [\sigma_3 \cdots \sigma_n]$

$[\sigma_3 \cdots \sigma_n] \rightarrow [\sigma_4 \cdots \sigma_n]$

$\vdots$

$[\sigma_{n-1}\sigma_n] \rightarrow [\sigma_{n-1}][\sigma_n]$

Problem 5

For this problem $\Sigma = \{a, b\}$.

a) Give an algorithm that will do the following:

Given a string w*, outputs a Chomsky Normal Form CFG* G *such that* $L(G) = \{w\}$ *(so* G *generates* w *and nothing else).* Your algorithm can use DOT DOT DOT.

b) f such that your grammar had $O(f(n))$ rules.

$w = \sigma_1 \cdots \sigma_n$.

$S \rightarrow [\sigma_1][\sigma_2 \cdots \sigma_n]$

$[\sigma_2 \cdots \sigma_n] \rightarrow [\sigma_3 \cdots \sigma_n]$

$[\sigma_3 \cdots \sigma_n] \rightarrow [\sigma_4 \cdots \sigma_n]$

$\vdots$

$[\sigma_{n-1}\sigma_n] \rightarrow [\sigma_{n-1}][\sigma_n]$

$[a] \rightarrow a$

Problem 5

For this problem $\Sigma = \{a, b\}$.

a) Give an algorithm that will do the following:

Given a string w, outputs a Chomsky Normal Form CFG G such that $L(G) = \{w\}$ (so G generates w and nothing else). Your algorithm can use DOT DOT DOT.

b) f such that your grammar had $O(f(n))$ rules.

$w = \sigma_1 \cdots \sigma_n$.

$S \rightarrow [\sigma_1][\sigma_2 \cdots \sigma_n]$

$[\sigma_2 \cdots \sigma_n] \rightarrow [\sigma_3 \cdots \sigma_n]$

$[\sigma_3 \cdots \sigma_n] \rightarrow [\sigma_4 \cdots \sigma_n]$

$\vdots$

$[\sigma_{n-1}\sigma_n] \rightarrow [\sigma_{n-1}][\sigma_n]$

$[a] \rightarrow a$

$[b] \rightarrow b.$

Problem 5

For this problem $\Sigma = \{a, b\}$.

a) Give an algorithm that will do the following:

Given a string w, outputs a Chomsky Normal Form CFG G such that $L(G) = \{w\}$ (so G generates w and nothing else). Your algorithm can use DOT DOT DOT.

b) f such that your grammar had $O(f(n))$ rules.

$w = \sigma_1 \cdots \sigma_n$.

$S \to [\sigma_1][\sigma_2 \cdots \sigma_n]$

$[\sigma_2 \cdots \sigma_n] \to [\sigma_3 \cdots \sigma_n]$

$[\sigma_3 \cdots \sigma_n] \to [\sigma_4 \cdots \sigma_n]$

$\vdots$

$[\sigma_{n-1}\sigma_n] \to [\sigma_{n-1}][\sigma_n]$

$[a] \to a$

$[b] \to b$.

2) This has $O(n)$ rules so $f(n) = n$.

Problem 5

For this problem $\Sigma = \{a, b\}$.
a) Give an algorithm that will do the following:
Given a string w, outputs a Chomsky Normal Form CFG G such that $L(G) = \{w\}$ (so G generates w and nothing else). Your algorithm can use DOT DOT DOT.
b) f such that your grammar had $O(f(n))$ rules.
$w = \sigma_1 \cdots \sigma_n$.
$S \rightarrow [\sigma_1][\sigma_2 \cdots \sigma_n]$
$[\sigma_2 \cdots \sigma_n] \rightarrow [\sigma_3 \cdots \sigma_n]$
$[\sigma_3 \cdots \sigma_n] \rightarrow [\sigma_4 \cdots \sigma_n]$
$\vdots$
$[\sigma_{n-1}\sigma_n] \rightarrow [\sigma_{n-1}][\sigma_n]$
$[a] \rightarrow a$
$[b] \rightarrow b$.
2) This has $O(n)$ rules so $f(n) = n$.
Grading Notes on Next Slide

Problem 5-Grading

Problem 5-Grading

1) Grammar WRONG or VERY UNCLEAR but DO say $f(n) = n$. Student gets 5 points. The 5 points are generous on our part since they clearly did not come from your incorrect grammar. If you ask for a regrade request you will lose those 5 points.

Problem 5-Grading

1) Grammar WRONG or VERY UNCLEAR but DO say $f(n) = n$. Student gets 5 points. The 5 points are generous on our part since they clearly did not come from your incorrect grammar. If you ask for a regrade request you will lose those 5 points.
2) Grammar is probably correct (done recursively) but $f(n) = \log n$ is given. This is incorrect and provably so: there are w of length n that **require** $\Omega(n)$ rules. Student gets 10 points.

Problem 5-Grading

1) Grammar WRONG or VERY UNCLEAR but DO say $f(n) = n$. Student gets 5 points. The 5 points are generous on our part since they clearly did not come from your incorrect grammar. If you ask for a regrade request you will lose those 5 points.
2) Grammar is probably correct (done recursively) but $f(n) = \log n$ is given. This is incorrect and provably so: there are w of length n that **require** $\Omega(n)$ rules. Student gets 10 points.
3) Some students did a DFA for $\{w\}$ and used theorem to to from DFA to CFG. This is not quite CNF but we allowed it. A very strange way to do the problem. DO NOT do this on the final.

Problem 5-Grading

1) Grammar WRONG or VERY UNCLEAR but DO say $f(n) = n$. Student gets 5 points. The 5 points are generous on our part since they clearly did not come from your incorrect grammar. If you ask for a regrade request you will lose those 5 points.
2) Grammar is probably correct (done recursively) but $f(n) = \log n$ is given. This is incorrect and provably so: there are w of length n that **require** $\Omega(n)$ rules. Student gets 10 points.
3) Some students did a DFA for $\{w\}$ and used theorem to to from DFA to CFG. This is not quite CNF but we allowed it. A very strange way to do the problem. DO NOT do this on the final.
4) Some students used that every regex is a CFL. I don't know off hand if you get a CNF grammar but we allowed it (prob close to CNF and might even BE CNF). A very strange way to do the problem. DO NOT do this on the final.

Question Inspired By Problem 5

Question Inspired By Problem 5

$\forall w$, $|w| = n$, $\exists$ CNFG for $\{w\}$ of size $O(n)$.

Question Inspired By Problem 5

$\forall w$, $|w| = n$, $\exists$ CNFG for $\{w\}$ of size $O(n)$.
Question Does there exist w such that $|w| = n$ and

Question Inspired By Problem 5

$\forall w$, $|w| = n$, $\exists$ CNFG for $\{w\}$ of size $O(n)$.
Question Does there exist w such that $|w| = n$ and
$\forall$ CNFG G that generate $\{w\}$, G has $\Omega(n)$ rules.

Question Inspired By Problem 5

$\forall w$, $|w| = n$, $\exists$ CNFG for $\{w\}$ of size $O(n)$.
Question Does there exist w such that $|w| = n$ and

$\forall$ CNFG G that generate $\{w\}$, G has $\Omega(n)$ rules.

Vote YES there is such w, NO there is no such w, Unknown to Science, Unknown to Bill.

Question Inspired By Problem 5

$\forall w$, $|w| = n$, $\exists$ CNFG for $\{w\}$ of size $O(n)$.
Question Does there exist w such that $|w| = n$ and

$\forall$ CNFG G that generate $\{w\}$, G has $\Omega(n)$ rules.

Vote YES there is such w, NO there is no such w, Unknown to Science, Unknown to Bill.

Answer There is such a w. Intuition is that w is random so there is no pattern to use to get a small grammar.

Problem 6

Give an algorithm that does the following:
On input a Context Free Grammar G that generates L output a context free grammar that generates L^R.

Problem 6

Give an algorithm that does the following:
On input a Context Free Grammar G that generates L output a context free grammar that generates L^R.

Given a CFG (V, Σ, R, S) do the following:
Replace every rule of the form $A \rightarrow \alpha$ with $A \rightarrow \alpha^R$.

Problem 6

Give an algorithm that does the following:
On input a Context Free Grammar G that generates L output a context free grammar that generates L^R*.*

Given a CFG (V, Σ, R, S) do the following:
Replace every rule of the form $A \to \alpha$ with $A \to \alpha^R$.
DONE

Problem 6

Give an algorithm that does the following:
On input a Context Free Grammar G that generates L output a context free grammar that generates L^R.

Given a CFG (V, Σ, R, S) do the following:
Replace every rule of the form $A \rightarrow \alpha$ with $A \rightarrow \alpha^R$.
DONE

Note Many of you used Chomsky Normal Form. Thats fine but not needed.