Natural Language Processing (JHU 601.465/665)
Answers to "Parsing" practice problems

1. (a) A CNF recognizer combines
Y from [start,mid] + Z from [mid,end].
An FF recognizer must combine
Y from [start,mid-1] + a from [mid-1,mid] + Z from [mid,end]

The resulting constituent will be at least 3 words long; there must be at least two positions
between start and end, namely mid-1 and mid.

Changes needed:

line 7: "width := 2 to n" --> "width := 3 to n"
line 10: "mid := start+1 to end-1" --> "mid := start+2 to end-1"
line 11: "for (X->Y Z) in Rules" --> "for (X->Y a Z) in Rules"
line 12: "if Chart[start,mid,Y] and Chart[mid,end,Z]"
--> "if Chart[start,mid-1,Y] and Input[mid]=a and Chart[mid,end,Z]"

(b) Each ternary rule X -> Y a Z
should be replaced with the rules
X -> Y Z'
Z' -> A Z
A -> a
where the names Z' and A are chosen to be new nonterminals not already in the grammar.

2. (a) (0, Q4) -- created by SCANning "plus" from (0,Q3)
(2, Q0) -- created by PREDICTing "Num" from (0,Q4)

(b) (0, Q5) -- created by SCANning "negative" from (2,Q4).
(3, Q0) -- created by PREDICTing "Num" from (0, Q5).

Note that if the sentence had instead been "one plus four plus two," then we would have obtained (2,Q1) in column 3 by SCANning "four" from (2,Q0).

Here is the full chart:

one	plus	neg	four	plus	two	
0	1	2	3	4	5	6
0, Q2	0, Q1	0, Q4	0, Q5	3, Q1	0, Q4	5, Q1
0, Q0	0, Q3	2, Q0	3, Q0	0, Q6	5, Q0	0, Q6

By the way, notice that the number of entries per column is bounded. For this grammar, that will always be true. On such grammars, Earley's algorithm will run in O(n) time.

3. The best answer is (d).

(a) Wrong. (You may have recognized that discarding the VP under these conditions is a common pruning method that is UNSAFE. If the answer were really yes, then it would be safe.)

(b) is wrong. In CKY, once a cell's probability is computed, it does not change again.

(c) is more on the right track. As it says, there's no guarantee that there's any constituent from 3 to 7. Nonetheless, this doesn't directly rebut the claim being discussed.

(d) This is the best answer. The claim asks about p(NP from 3 to 7 | whole sentence), which would need to consider whether the context actually calls for an NP. But the inside probability is determined by the words from 3 to 7 only, not by the whole

sentence.

(The inside probability specifically represents p(words from 3 to 7 | NP). In other words, how likely is it that starting randsent with NP would generate the words from 3 to 7? To see why this is what the inside probability represents, think about how it is computed from PCFG rule probabilities, and how those rule probabilities relate to the behavior of randsent.)

(e) As it says, the inside probability isn't about the highest-probability parse. But neither is the claim being discussed.

Suppose there are 3 parses:

A	(relative probability 40%)	contains VP from 3 to 7
B	(relative probability 30%)	contains NP from 3 to 7
C	(relative probability 30%)	contains NP from 3 to 7

Then the highest-probability parse has a VP, but if the model is correct, then a speaker of this sentence would intend an NP parse 60% of the time.

So we do want to look at a sum over parses. But the inside probability is not the sum we want, since it is only a sum over subparses (of the substring from 3 to 7). Later in the course, we'll see what sum we do want.

4. (a) Remember, a PCFG starts with the S symbol and decides whether to expand it into NP VP or something else. So we are modeling the probability of NP VP given S:

```
p(NP[...] VP[...] | S[...])
 = 1/Z(S[...]) exp sum_i theta_i f_i(S[...],NP[...] VP[...])
```

A number of people got this backwards and tried to model the conditional probability of the parent S given the children NP VP. But that doesn't really make sense unless NP and VP have previously decided to be siblings and have a common parent. The probability of that choice would have to be modeled and multiplied into the overall probability of the parse tree.

Rather that modeling a sequence of bottom-up parsing choices in that way (yes, it is possible), we have taken a generative PCFG approach. That is, even though we're COMPUTING the probability of a parse from the bottom up, the probability is DEFINED by how likely randsent would be to generate that parse from the top down.

(b) O(V^3). (The full runtime is O(n^3 V^3), but we said that n is a constant here, so n^3 is a constant and can be omitted.)

(c) One answer is that any fine-grained rule in G has to be less probable than the corresponding coarse-grained rule in G'. This ensures that any fine-grained parse has a coarse-grained parse whose probability is at least as high, so there can't be any probable fine-grained parse if there wasn't any probable coarse-grained parse.

As we discussed in class when talking about parsing heuristics, one way to arrange this is to define the coarse-grained p(NP VP | S) to be the MAXIMUM of all fine-grained probabilities of the form p(NP[...] VP[...] | S[...]). Note that p(. | S) is not really a probability distribution in this case -- it doesn't sum to 1. However, we can still run CKY on it and find the "probability" of the best coarse-grained parse, which is an

upper bound on the probability of the best fine-grained parse.

We gave full credit for the above answer. If you read the question carefully, however, you'll see that it actually asks about the probability of the SENTENCE, not the probability of the best PARSE. This requires summing over all possible parses. There are exponentially more fine-grained parses than coarse-grained parses, so if a single coarse-grained parse has "probability" comparable to a single fine-grained parse (even a good one), then the coarse-grained sum will not be as high as the fine-grained sum. The right answer (for extra credit!) is that p(NP VP | S) is the maximum over all S^ of the sum over all NP^, VP^ of p(NP^ VP^ | S^), where S^, NP^, and VP^ are fine-grained nonterminals of the form S[...], NP[...], VP[...]. In that case, the inside algorithm on G' will find the total probability of all fine-grained parses under G, PLUS the probability of illegal parses where a node like NP[singular], once generated, can turn into NP[plural] for free and rewrite using the rules for NP[plural]. This is an upper bound on the inside probability under G, as desired.

(d) No. The UNNORMALIZED probabilities under G will indeed be lower than under G', because the score is decreased by the negative-weighted features before it is exponentiated. (Assuming that the feature functions f_k return values >= 0.) However, the NORMALIZED probabilities are made to sum to 1 again, so some rule probabilities will go up and some will go down.

In the end, both G and G' are true PCFGs under this scheme (in contrast to (c)), so each one gives a probability distribution over all possible sentences. Some sentences will be less likely under G than under G', but some will be more likely, to balance things out.