6. (10 points) Consider a PCFG with the following rules (numbers are provided so you can reference them in your solution):

1. DT $\rightarrow$ the [1.0]
2. N $\rightarrow$ ring [0.5]
3. N $\rightarrow$ rings [0.5]
4. CC $\rightarrow$ and [1.0]

5. ROOT $\rightarrow$ NP [0.5]
6. ROOT $\rightarrow$ NP CC NP [0.5]
7. NP $\rightarrow$ N [0.5]
8. NP $\rightarrow$ DT N [0.5]

Define our PCFG to have these rules, the nonterminals {NP, CC, N, DT}, terminals {the, ring, rings, and} and root symbol ROOT.

a. (4 points) What parses are possible for the sentence *the ring*? For each parse, list both the tree **and its probability**.

ROOT $\rightarrow$ NP $\rightarrow$ DT, N, score = -3. Note that the tree is rooted at the ROOT symbol, so you need to account for this rule.

the	ring
DT[1.0]	NP1[0.5]/NP2[0.5]
	N1[0.5]/N2[0.5]

$0.5 \times 0.5 \times 1.0$
$= .25$

Root [0.5]
|
NP1[0.5]
|
N1[0.5]
|
DT[1.0]

ROOT[0.5]
|
NP2[0.5]
|
N2[0.5]
|
DT[1.0]

$\text{Score} = \ln(0.5 * 0.5 * 0.5 * 1.0)$
$= -3.$

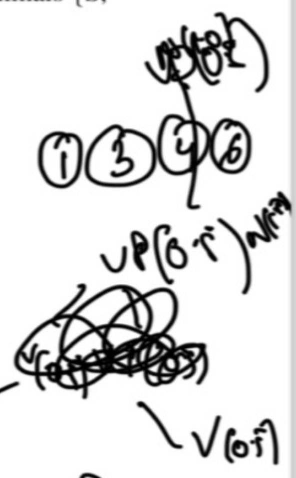
_____ Version A: B / Version B: B (6) Consider the sentence *I like to run* and a PCFG with nonterminals {S, VP, N, P, V}, terminals {I, like, to, run}, start symbol S, and rules:

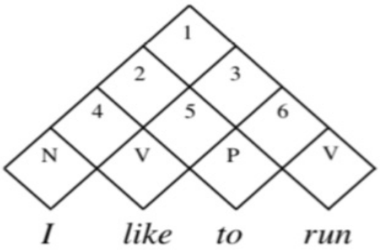
S → N VP [0.5]
S → N V [0.5]
VP → V VP [0.5]
VP → P V [0.5]
V → run [0.5]
V → like [0.5]
N → I [1.0]
P → to [1.0]

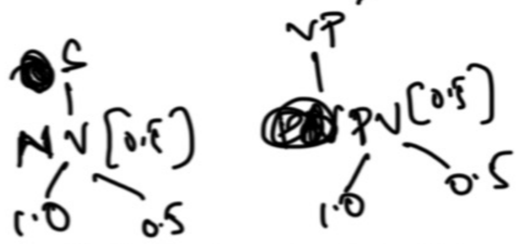
I like to run

N(1.0)	S(0.5)	φ	VP(0.5)
	V(0.5)	φ	VP(0.5)
		P(1.0)	VP(0.5)
			V(0.5)

1 3 4 6







The figure has a CKY chart with possible symbols for the leaves filled in showing nonterminals with log probability greater than negative infinity (i.e., nonterminals that are possible to build over these spans of the sentence). Which other cells contain at least one nonterminal with log probability greater than negative infinity?

A. 1, 3, 6
B. 1, 3, 4, 6
C. 1, 2, 3, 4, 6
D. 1, 3, 4, 5, 6

0.5 * 1 * 0.5 * 0.5 * 0.5 * 1.0 * 0.5

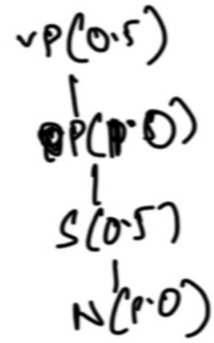
1/25

_____ Version A: B / Version B: C (7) Considering the same sentence and PCFG as above, what is the log probability using log base 2 ($\log(0.5) = -1$) of the most probable tree for this sentence?

A. -4
B. -5
C. -6
D. -7

= -5 ✓

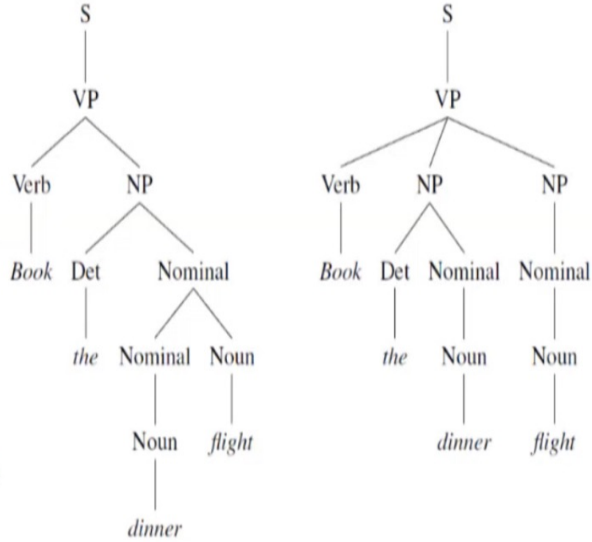
(.25)



PCFGs for disambiguation

Choose the parse tree with highest probability

Rules			P	Rules			P
S	→	VP	.05	S	→	VP	.05
VP	→	Verb NP	.20	VP	→	Verb NP NP	.10
NP	→	Det Nominal	.20	NP	→	Det Nominal	.20
Nominal	→	Nominal Noun	.20	NP	→	Nominal	.15
Nominal	→	Noun	.75	Nominal	→	Noun	.75
				Nominal	→	Noun	.75
Verb	→	book	.30	Verb	→	book	.30
Det	→	the	.60	Det	→	the	.60
Noun	→	dinner	.10	Noun	→	dinner	.10
Noun	→	flight	.40	Noun	→	flight	.40



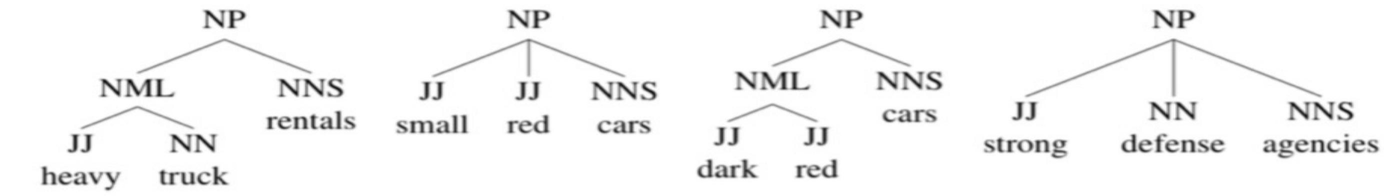
$$P(T_{left}) = .05 * .20 * .20 * .20 * .75 * .30 * .60 * .10 * .40 = \mathbf{2.2 \times 10^{-6}}$$

$$P(T_{right}) = .05 * .10 * .20 * .15 * .75 * .75 * .30 * .60 * .10 * .40 = \mathbf{6.1 \times 10^{-7}}$$

function PROBABILISTIC-CKY(*words,grammar*) **returns** most probable parse and its probability

for $j \leftarrow$ **from** 1 **to** LENGTH(*words*) **do**
  for all $\{ A \mid A \rightarrow words[j] \in grammar\}$
    $table[j-1, j, A] \leftarrow P(A \rightarrow words[j])$
  for $i \leftarrow$ **from** $j-2$ **downto** 0 **do**
    for $k \leftarrow i+1$ **to** $j-1$ **do**
      for all $\{ A \mid A \rightarrow BC \in grammar,$ **and** $table[i,k,B] > 0$ **and** $table[k,j,C] > 0 \}$
        if $(table[i,j,A] < P(A \rightarrow BC) \times table[i,k,B] \times table[k,j,C])$ **then**
          $table[i,j,A] \leftarrow P(A \rightarrow BC) \times table[i,k,B] \times table[k,j,C]$
          $back[i,j,A] \leftarrow \{k, B, C\}$
return BUILD_TREE($back$[1, LENGTH(*words*), S]), $table$[1, LENGTH(*words*), S]

7. (17 points) Suppose you have the following NPs provided to your as your treebank:



You will construct a PCFG from these trees with NP as the start symbol in this grammar.

To binarize the trees, you will introduce a $\overline{\text{NP}}$ symbol to turn the ternary rules into binary rules (you can write NP-B for "NP bar" if you're typing your answer). This will be described in the following questions.

For all question parts, assume that Greg comes along with a great part-of-speech tagger and tags these sentences. So: (a) **do not write down any grammar rules in the lexicon; only include the rules above the part-of-speech tag layer**; (b) **when doing CKY, assume the gold tag has a score of 0.0 and all other tags have a score of** $-\infty$. In this case, each word's tag is unambiguous anyway, so this does not actually change your answer, just reduces the amount of writing.

a. (5 points) Write the grammar (**rules and probabilities**) you get if you use right-binarization: that is, $\overline{\text{NP}}$ is introduced as a right child under the NP symbol for ternary-branching rules. (So NP $\rightarrow$ JJ $\overline{\text{NP}}$ is introduced for the *small red cars* example).

NP → NML NNS [0.5]

NML → JJ NN [0.5]

NML → JJ JJ [0.5]

NP → JJ NP-B [0.5]

NP-B → JJ NNS [0.5]

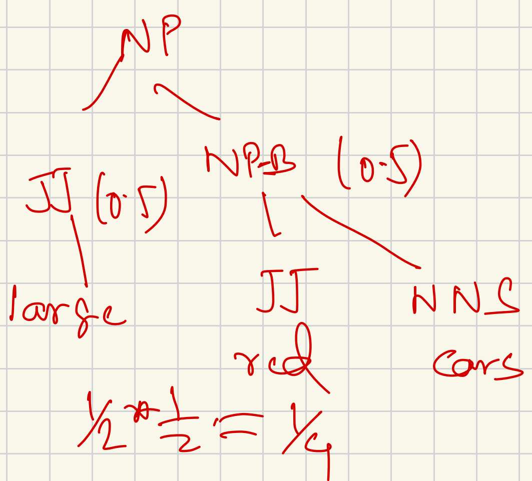
NP-B → ~~NN~~ NNS [0.5]

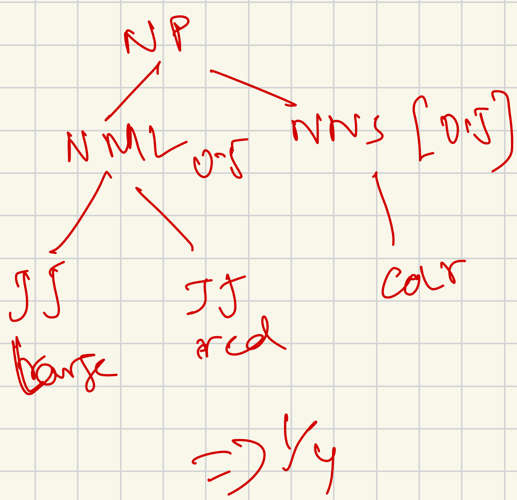
b. (4 points) Now parse the phrase *large red cars* with the tags JJ JJ NNS. Show your work and write out **every parse with nonzero probability** and its probability.

There are two parses:

1/4 for the parse with NML: (NP (NML (JJ large) (JJ red)) (NNS cars))

1/4 for the NP-B parse: (NP (JJ large) (NP-B (JJ red) (NNS cars)))





c. (4 points) Write the grammar you get if you use right-binarization on *strong defense agencies* but left-binarization on *small red cars*. That is, for that one tree, make the $\overline{NP}$ the left child of the root NP.

NP → NML NNS 0.5 NP → NP-B NNS 0.25 NP → JJ NP-B 0.25 NP-B → JJ JJ 0.5 NP-B → NN NNS 0.5 NML → JJ NN 0.5 NML → JJ JJ 0.5

Strong defense agencies
JJ NN NNS

NP → NML NNS [0.5]

d. (4 points) Now parse *large red cars* with tags JJ JJ NNS using this new grammar. Show your work, write out **every parse with nonzero probability** and its probability.

1/4 for the parse with NML: (NP (NML (JJ large) (JJ red)) (NNS cars))
1/8 for the parse with NP-B, which is now left-branching: (NP (NP-B (JJ large) (JJ red)) (NNS cars))

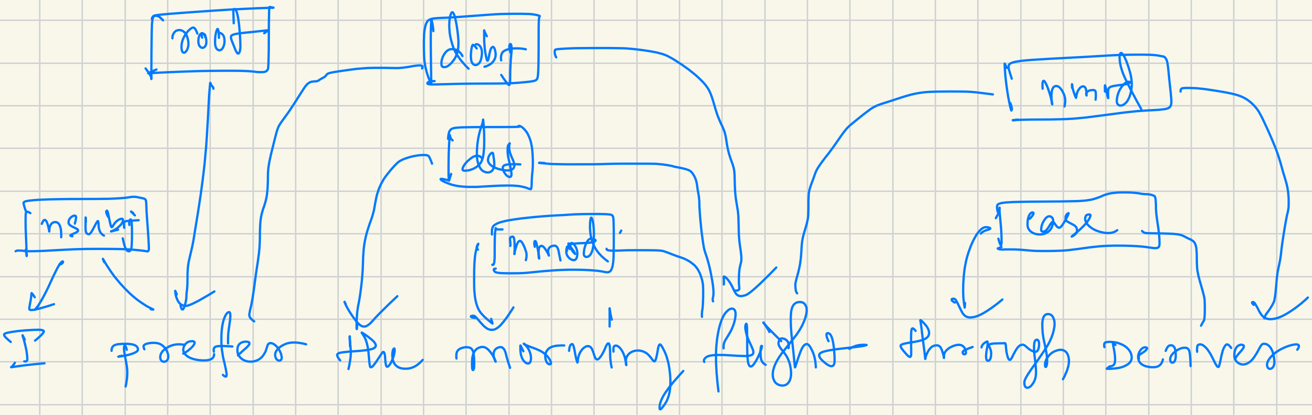
Suppose you are running a shift-reduce dependency parser with the arc-standard system on a sen- tence of length n. How many shift operations are needed?
A. Cannot be determined
B. n – 1
C. n
D. n + 1
D. 2n – 1
E. 2n

Dependency Parsing:

Dependency grammar provides another representation (apart from CFG) of language as a graph.

nodes → words

edges → dependencies.



** Relations among the words may be illustrated as directed, labeled arcs from heads to dependents.

** dependency grammar approach abstract away from word-order information that is necessary for the parse.

** dependency parsing is important for "Coreference Resolution", "QA", "Information Retrival"

☐ Dependency structures (→ dependency tree) are represented by directed graphs that satisfy the following constraints:

1. There is a single designated root node that has no incoming arcs.
2. with the exception of the root node, each vertex has exactly one incoming arc.

3. There is a unique path from the root node to each vertex in V.

- There are multiple algorithms that generate dependency trees from data.

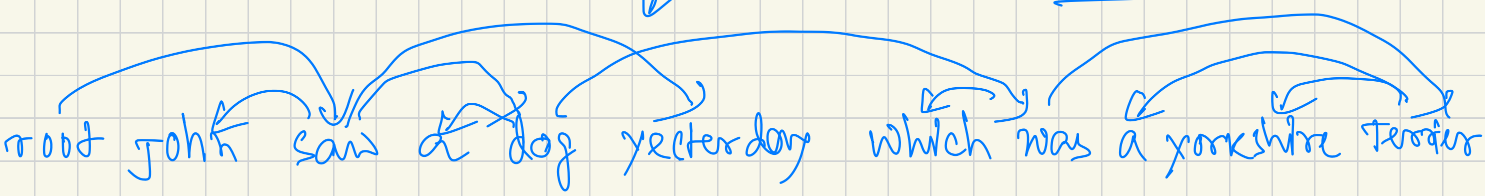
  → Transition based approaches:
  shift-reduce

  → Graph Algorithms:
  Maximum spanning tree.

** Some algorithms are $\boxed{\text{projective}}$:

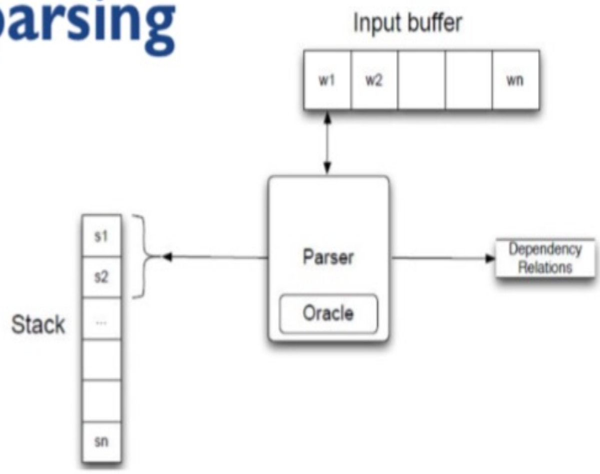
↳ the edges in the graph are not allowed to cross. While most English sentences are projective, they are not in all cases.

these crosses won't be present



Transition based dependency parsing

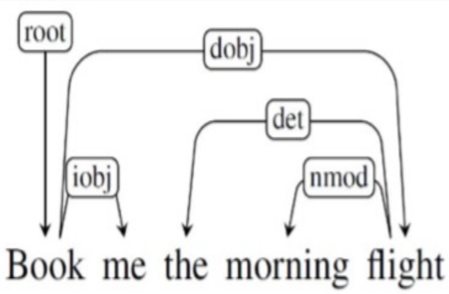
- Based on shift-reduce parsing (from Compiler)
- Initial Configuration
  - Stack contains ROOT node.
  - Word list is initialized with the set of the words
- Goal Configuration
  - Empty stack and word list
  - Set of relations represents the final parse
- Actions
  - LEFT ARC
    - head-dependent relation between top and (top-1)
    - remove (top-1) from the stack
  - RIGHT ARC
    - head-dependent relation between (top-1) and top
    - remove top from the stack
  - SHIFT
    - Remove the word from the front of the input buffer and push it onto the stack



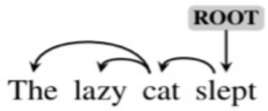

```
function DEPENDENCYPARSE(words) returns dependency tree

  state ← {[root], [words], [] }  ; initial configuration
  while state not final
    t ← ORACLE(state)        ; choose a transition operator to apply
    state ← APPLY(t, state)  ; apply it, creating a new state
  return state
```


Example of transition-based parsing

Step	Stack	Word List	Action	Relation Added
0	[root]	[book, me, the, morning, flight]	SHIFT	
1	[root, book]	[me, the, morning, flight]	SHIFT	
2	[root, book, me]	[the, morning, flight]	RIGHTARC	(book → me)
3	[root, book]	[the, morning, flight]	SHIFT	
4	[root, book, the]	[morning, flight]	SHIFT	
5	[root, book, the, morning]	[flight]	SHIFT	
6	[root, book, the, morning, flight]	[]	LEFTARC	(morning ← flight)
7	[root, book, the, flight]	[]	LEFTARC	(the ← flight)
8	[root, book, flight]	[]	RIGHTARC	(book → flight)
9	[root, book]	[]	RIGHTARC	(root → book)
10	[root]	[]	Done	



f. (3 points) Consider the following sentence:



Give a correct sequence of shift-reduce operations that will lead to the correct parse. Recall that the start state is Stack=[ROOT] and Buffer=[the lazy cat slept].

SSSLLSLR

5. (12 points) Consider the following grammar:

VP $\rightarrow$ V N [0.25]
VP $\rightarrow$ V NP [0.25]
VP $\rightarrow$ V PP [0.25]
VP $\rightarrow$ VP PP [0.25]
NP $\rightarrow$ N PP [1.0]
PP $\rightarrow$ P N [1.0]

N $\rightarrow$ reports [0.5]
N $\rightarrow$ Mars [0.5]
P $\rightarrow$ on [1.0]
V $\rightarrow$ wrote [0.5]
V $\rightarrow$ reports [0.5]

Define a PCFG with this grammar including the nonterminals {VP, NP, PP, N, P, V}, the terminals {reports, Mars, on, wrote}, the start symbol VP, and the probabilities as given above. That is, this is a grammar to generate verb phrases. **For the purposes of this problem, you can assume** $\log 0.5 = -1$ **(i.e., use base 2 for the logarithm).**

a. (3 points) Consider the new verb phrase ***reports on Mars***. Parse this sentence with the grammar. Write **both** the highest-probability parse of the sentence **and** the joint probability of that parse and the words $P(T, x)$.

CKY chart below with scores. The best possible parse is the one rooted in VP. However, many students failed to observe that the root symbol was VP and gave the NP parse. Since the importance of starting with the root symbol wasn't really emphasized in class, we decided not to take off points for this.

```
            VP-4/NP-2
        NONE        PP-1
V-1/N-1   P0    N-1
```

b. (5 points) Consider the new sentence ***wrote reports on Mars***. Parse this sentence with the grammar. Write **both** the highest-probability parse of the sentence **and** the joint probability of that parse and the words $P(T, x)$.

CKY chart below. The best possible parse uses VP $\rightarrow$ V NP at the top level.

```
      VP-5 from right one, -7 from wrong one
   NONE    NP-2/VP-4
 VP-4    NONE   PP-1
V-1  N/V-1   P 0    N-1
```

c. (2 points) How many parses of ***wrote reports on Mars*** can we build with this grammar?

2

d. (2 points) When parsing ***wrote reports on Mars***, what is the largest constituent (in terms of number of words) that we can build that is not part of a valid complete parse?

3. VP over (reports on Mars)

Notes on transition based parsing

① Are these sequence of actions unique?

NO.

→ there can be more than one path that leads to the same results.

→ there may be other transitions that lead to different equally valid parses due to ambiguity.

② How to generate labels of the arcs?

→ parameterize the LEFT_ARC & RIGHT_ARC operates with dependency labels [e.g. LEFT_ARC(NSUBJ) or RIGHT_ARC (DOBJ)]

→ job of oracle will be to return the correct action from a large set of actions.

③ Is oracle always right? NO.

④ How to create good oracle?

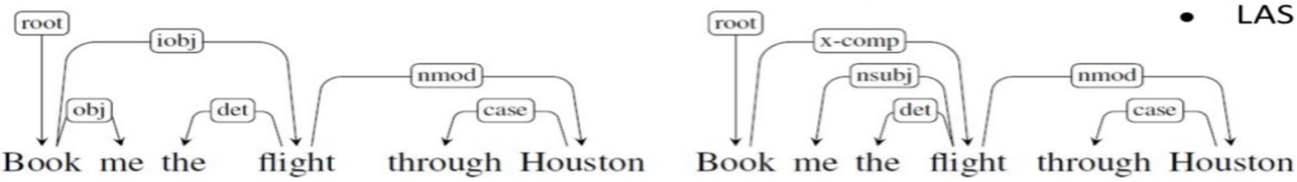
— Supervised ML method.

Evaluation of dependency parsing:

Evaluation of dependency parser

- Unlabeled Attachment Accuracy
  - Percentage of words in input assigned with the correct head
  - Also referred as unlabeled attachment score (UAS)
- Labelled Attachment Accuracy
  - Percentage of words in input assigned with correct head and dependency relation
  - Also referred as labeled attachment score (LAS)

- UAS = 5/6
- LAS = 4/6



Actual Tree

Predicted Tree

□ Semantic parsing :