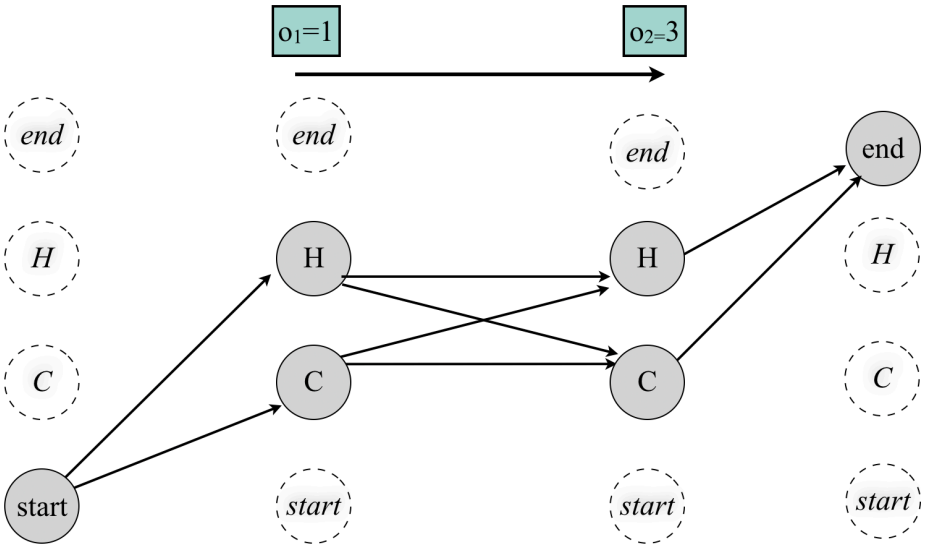
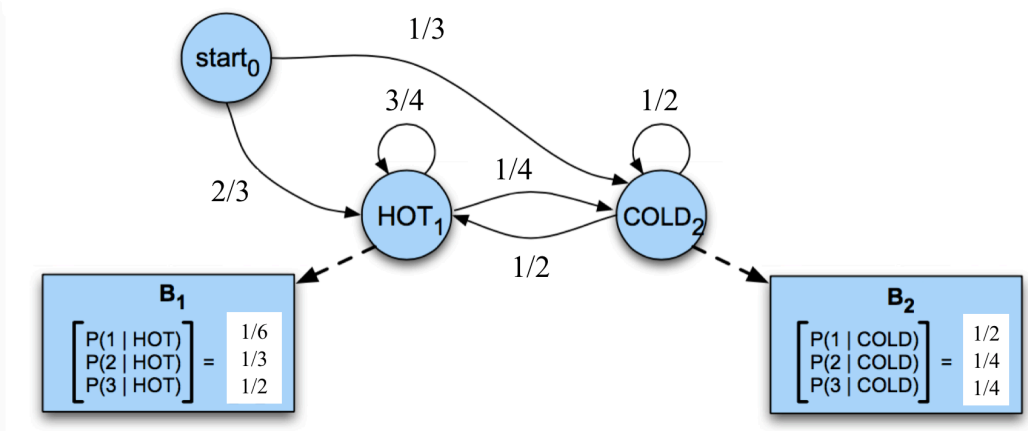
Problem set 2: NLP I. 100 points.
Choose a consistent citation format (e.g. APA, IEEE, LSA) to cite any sources you use in your write-up and in the reference section. You may use study groups. Discussion of pseudocode and theoretical solutions is fine, but final code and write-up should be individual work.

- You must use **Jupyter notebooks**. We will only run your code in the ondemand environment. Include this statement with your submission: *"I have confirmed that my code is functional in the ondemand environment."*
- Teams are assigned. Per problem, report on who did what, e.g.: *Solving*-jointly, *Coding*-team member 1, *Debugging*-jointly, *Analyzing*-team member 2, and Writing-*jointly*. **Take advantage of TA's office hours to discuss bugs, etc.**

Q2.1. Viterbi algorithm (Acknowledgement: Based on E. Prud'hommeaux.)
Part 1: Theory (15 p.): Consider the following Viterbi trellis, transition probabilities, and emission probabilities. They represent an ice cream diary (problem by J. Eisner) where {HOT, COLD} are weather states and the observations are number of ice creams enjoyed {1,2,3} . Then complete the trellis on p. 2.





Problem set 2: NLP I. 100 points.
Choose a consistent citation format (e.g. APA, IEEE, LSA) to cite any sources you use in your write-up and in the reference section. You may use study groups. Discussion of pseudocode and theoretical solutions is fine, but final code and write-up should be individual work.

Fill in the unshaded cells in the probability matrix below, corresponding to the trellis above. Each box should contain the value for that cell **plus** the mat used to arrive at that value. One of the boxes have been filled in for you. Fill in the back pointers where it says **b.p.**

	O_1 = 1 (ice cream)	**O_1 = 3 (ice creams)**	END b.p. = _____
	H v1(H) = P(start->H) * P(1|H) No need to find argmax because there is only one place to come from (START). v1(H) = 2/3 * 1/6 = 2/18	H v2(H) = _____ b.p. = _____	
	C v1(C) = _____	C v2(C) = _____ b.p. = _____	
START			

Problem set 2: NLP I. 100 points.
Choose a consistent citation format (e.g. APA, IEEE, LSA) to cite any sources you use in your write-up and in the reference section. You may use study groups. Discussion of pseudocode and theoretical solutions is fine, but final code and write-up should be individual work.

Part 2: Part-of-speech tagging implementation (45 points)

You will perform part of speech (POS) tagging with a Hidden Markov Models (HMMs). Specifically, your task is to implement the Viterbi algorithm to find the best sequence and tag each word in a sentence with a POS tag. Recall the following:

	POS tagger HMM
hidden states	POS tags
observations	words (tokens) of English
emission probabilities	ex: probability of seeing *dog* given that it's an NN
transition probabilities	ex: probability that a DT is followed by an NN

Two sets of probabilities: emission probabilities and transition probabilities have been calculated for you from a corpus of POS-tagged text. You are provided skeleton code (PS2viterbi.ipynb) that reads these probabilities from two files (emissProbs.txt and transitProbs.txt) into data structures for implementing your algorithm. You also receive code for testing your algorithm on the provided file of input test sentences (test-ps1.txt) and computing performance (accuracy) of your method.

Please review comments in the provided Jupyter notebook. To summarize, you are tasked to write a function that: (1) for a given input sentence, **populates the Viterbi lattice** and (2) using **backpointers, prints out the most probable part of speech sequence** for that sentence.

Q1: What percent of POS tags in the test set did your tagger get right (i.e., accuracy)?

Q2: Compute the most frequent POS tag baseline. You can do this by calling nltk.pos_tag() on each word separately inside your notebook, reusing the testing code as needed. How does the baseline accuracy compare to using Viterbi?

Q3: Discuss why the Viterbi probably will yield higher accuracy than the "pick the most frequent POS" method.

Bonus (+10): Pick 2 sentences each (so four) from two genres that may present problems for the POS tagger (e.g., Shakespearean English, a US politician's Twitter posts). Manually assign labels in the test file forma.. Run your tagger on the data. Compare the input and your implementation's output. Discuss output, including what is more likely: false positives (impact on precision) or false negatives (recall)?

In the submission, include (i) the Jupyter notebook as `ps2.1_LastNames` and (ii) a PDF writeup as `ps2.1_LastNames.pdf` which also includes the table in part A (figure is fine).

Problem set 2: NLP I. 100 points.
Choose a consistent citation format (e.g. APA, IEEE, LSA) to cite any sources you use in your write-up and in the reference section. You may use study groups. Discussion of pseudocode and theoretical solutions is fine, but final code and write-up should be individual work.

Q2.2. Train word embeddings *or* topic models. Complete option A or B (40 p.)

Option A: Select a large set of cohesive texts (e.g, all of Shakespeare translated from Early Modern English into Present-day English from Gutenberg) and train an LDA topic model with `gensim`. Experiment with number of topics and other parameters.

Option B: Select a substantial set of cohesive texts (see option A for an example) and train your own word2vec word embedding model using `gensim`. Examine pairs of words for similarity, explore analogies, and also devise two example to examine potential bias (e.g., consider similarity of the pair *nurse, woman* vs. *doctor, woman*). Some code snippets (not necessarily in order). You must also review documentation about the `gensim` library.

```
model = gensim.models.Word2Vec(toksents, size=100,
window=5, min_count=5, workers=4) #update line as needed
print("Done")
model.save("myw2v.model")
model = Word2Vec.load("myw2v.model")
model.wv['dog']                    #inspecting vector
model.wv.similarity('dog', 'cat')
```

In your write-up for A or B, ensure you discuss the algorithm behind the model, in addition to results.

In the submission, include (i) the Jupyter notebook as `ps2.2{a,b}_LastNames`. (ii) the input corpus file, and (iii) a PDF writeup as `ps2.2{a,b}_LastNames.pdf`.

Q2.3. Optional Bonus: Explaining NLP topic (+15 bonus p.)
Pick an NLP topic we discussed in class, e.g., logistic regression, text classification, Viterbi, kappa, word embeddings, ngram language models, etc. Imagine how you explain the topic to an elementary class (K-5). You must use your own words and should consider the audience. Use a slide deck and you may use your own visuals, point to demos, etc. Your slide deck should contain 3-4 slides.

In the submission, include the PDF of the slide deck as `ps2.3{a,b}_LastNames`.