Homework 3

CS 6347: Statistical methods in AI/ML

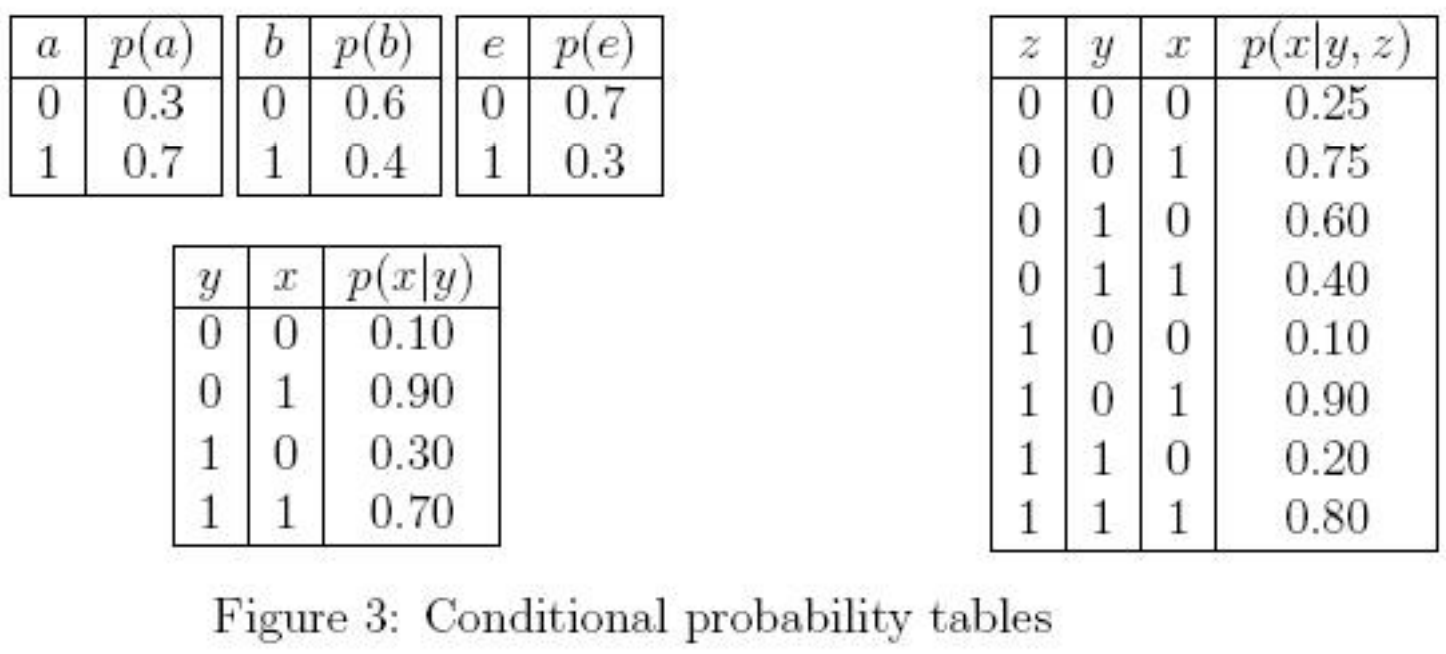
Instructor: Vibhav Gogate

Vibhav.Gogate@utdallas.edu

Student: Xiaodi Li

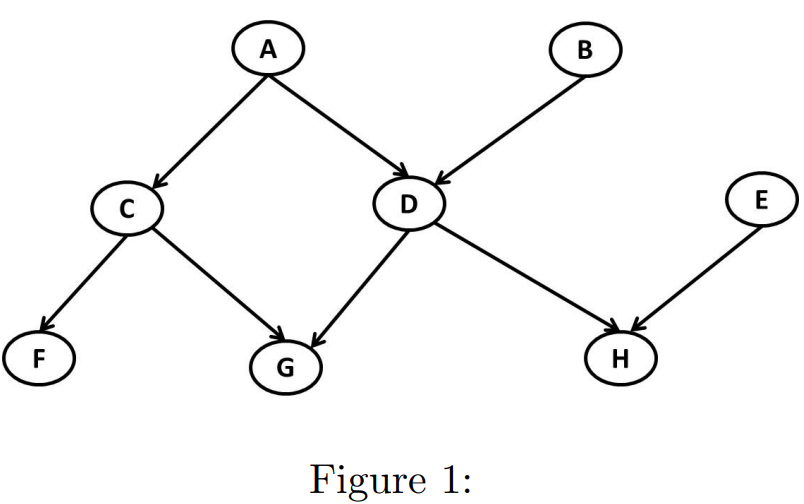
Net ID: XXL170011

**Problem 1: [20 points]\*\***



The question investigates the AND/OR search space of the network given in Figure 1 assuming that each variable is binary. The CPTs are given in Figure 3. The CPTs for , and are identical to the 3-dimensional CPT in Figure 3 and the CPTs for and are identical to the 2-dimensional CPT in the same figure

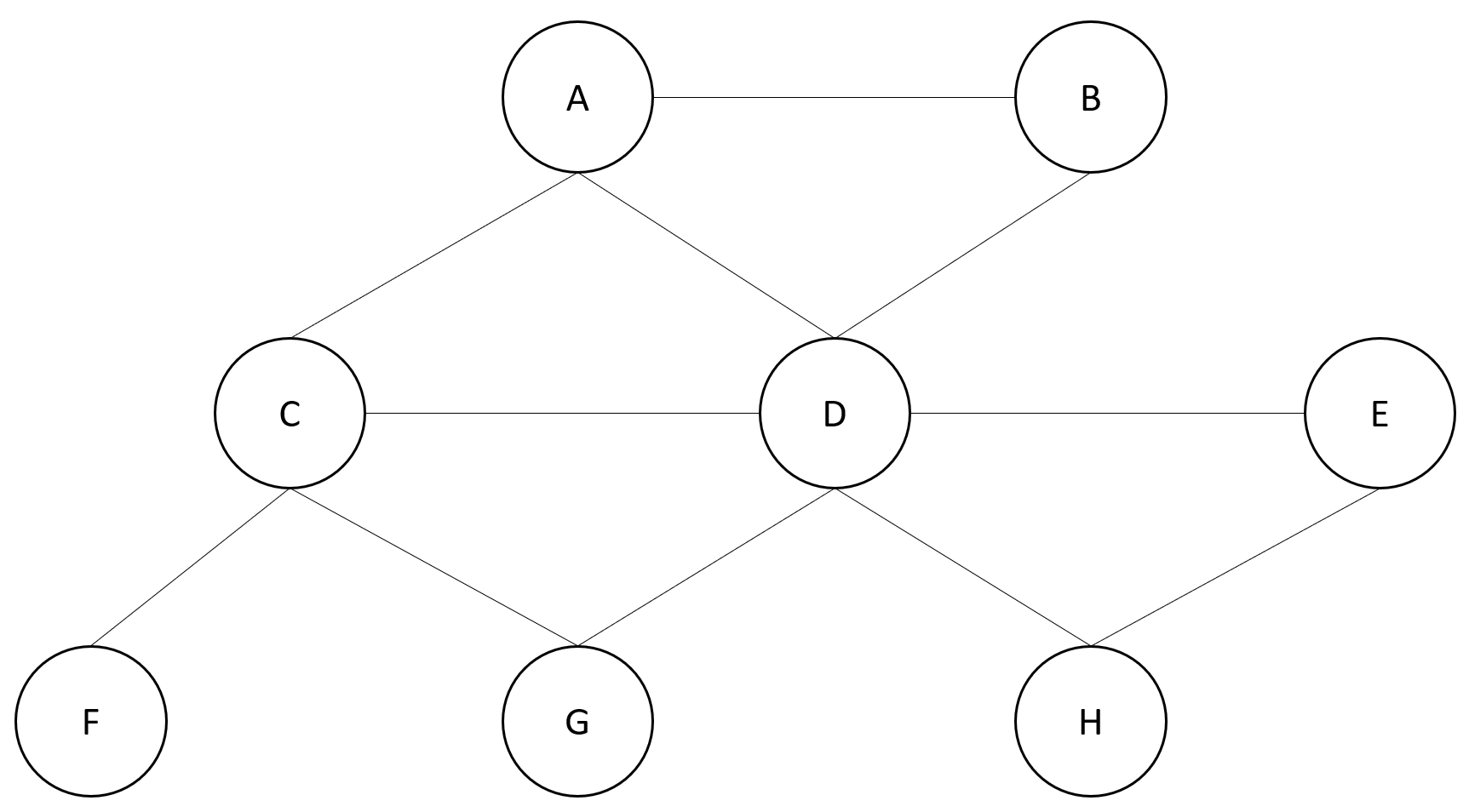
* Find and present a pseudo tree of the network whose depth is minimal. Call it . Do the best you can.
* Generate an AND/OR search tree driven by assuming that each variable has at most two values.
* Annotate the arcs with appropriate weights
* What is the computational cost of computing the probability of evidence and in such a network if you use depth-first search over the AND/OR search tree. Demonstrate your computation.
* Can the AND/OR search tree be reduced to a smaller AND/OR search graph.



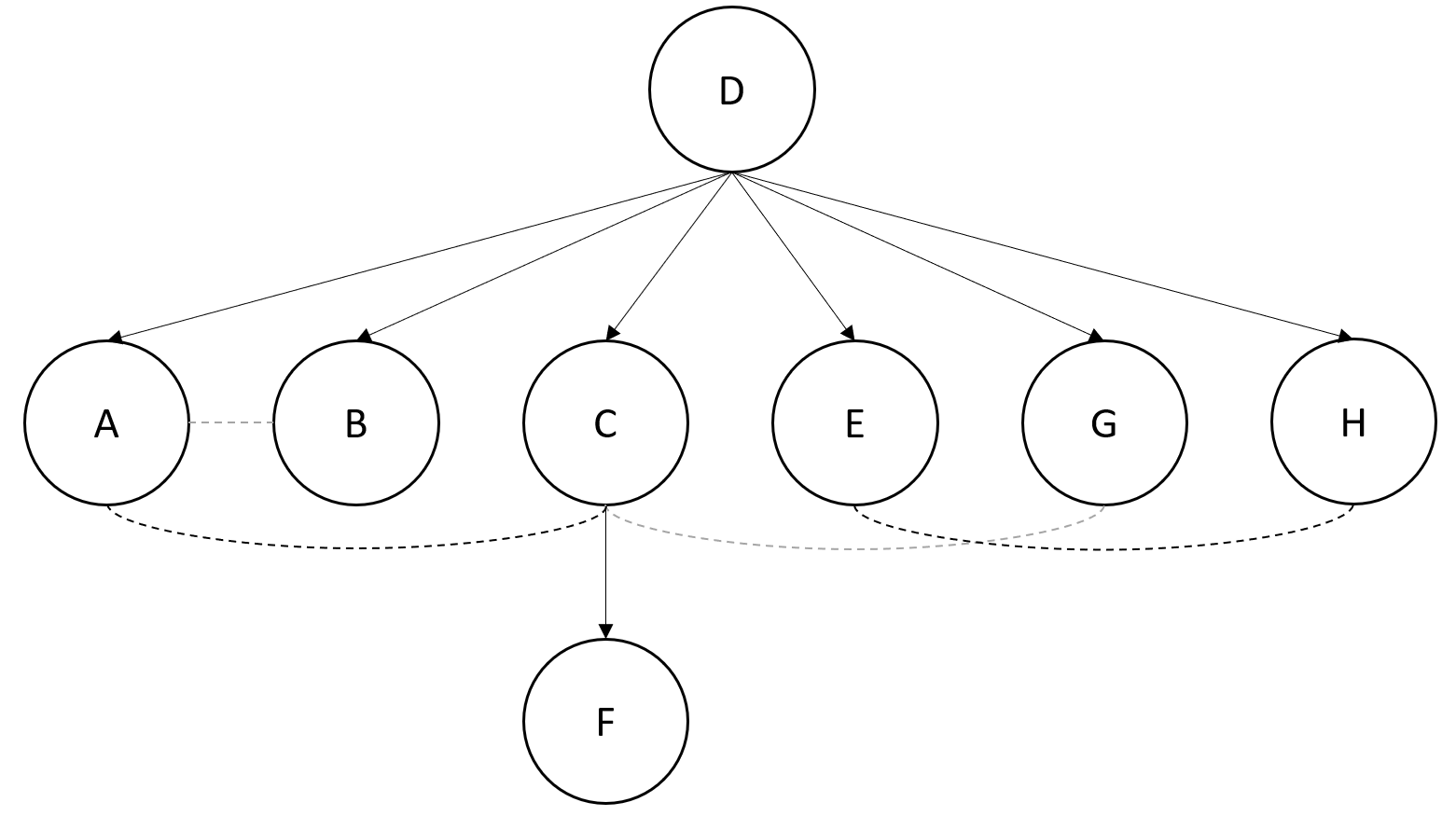
* Assume that the CPT P(x|y, z) is changed by making some entries deterministic as follows: the first two probabilities are changed to 1 and 0 respectively. Similarly, the last two probabilities are changed to 1 and 0 respectively. Show what would be the changes in the AND/OR search tree as a result.

**Answer:**

* First, we transfer the Bayesian network into a Markov network as follows:

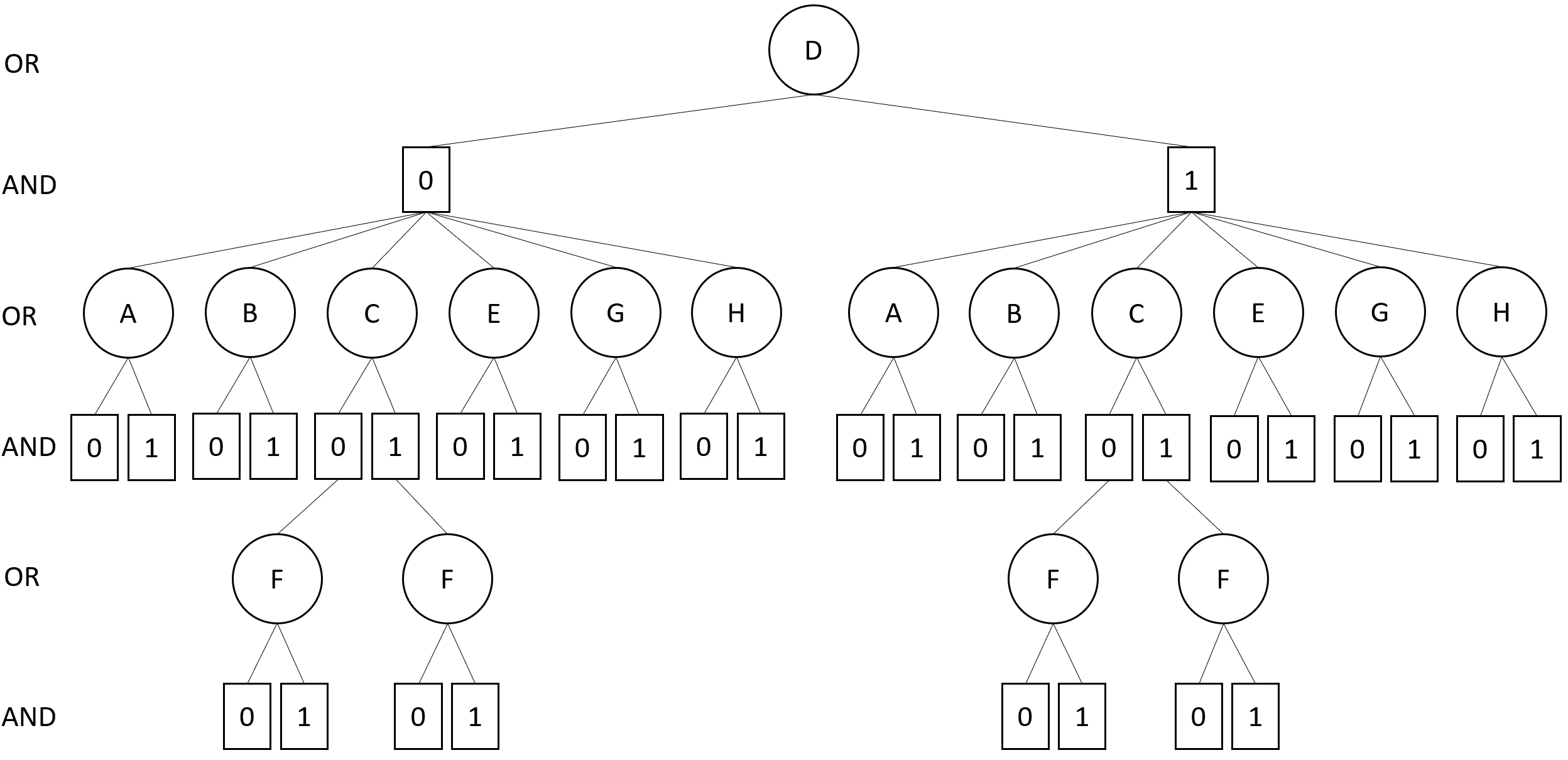
****

Then, using the Max degree heuristic method, we can construct the following pseudo tree (rooted at D):

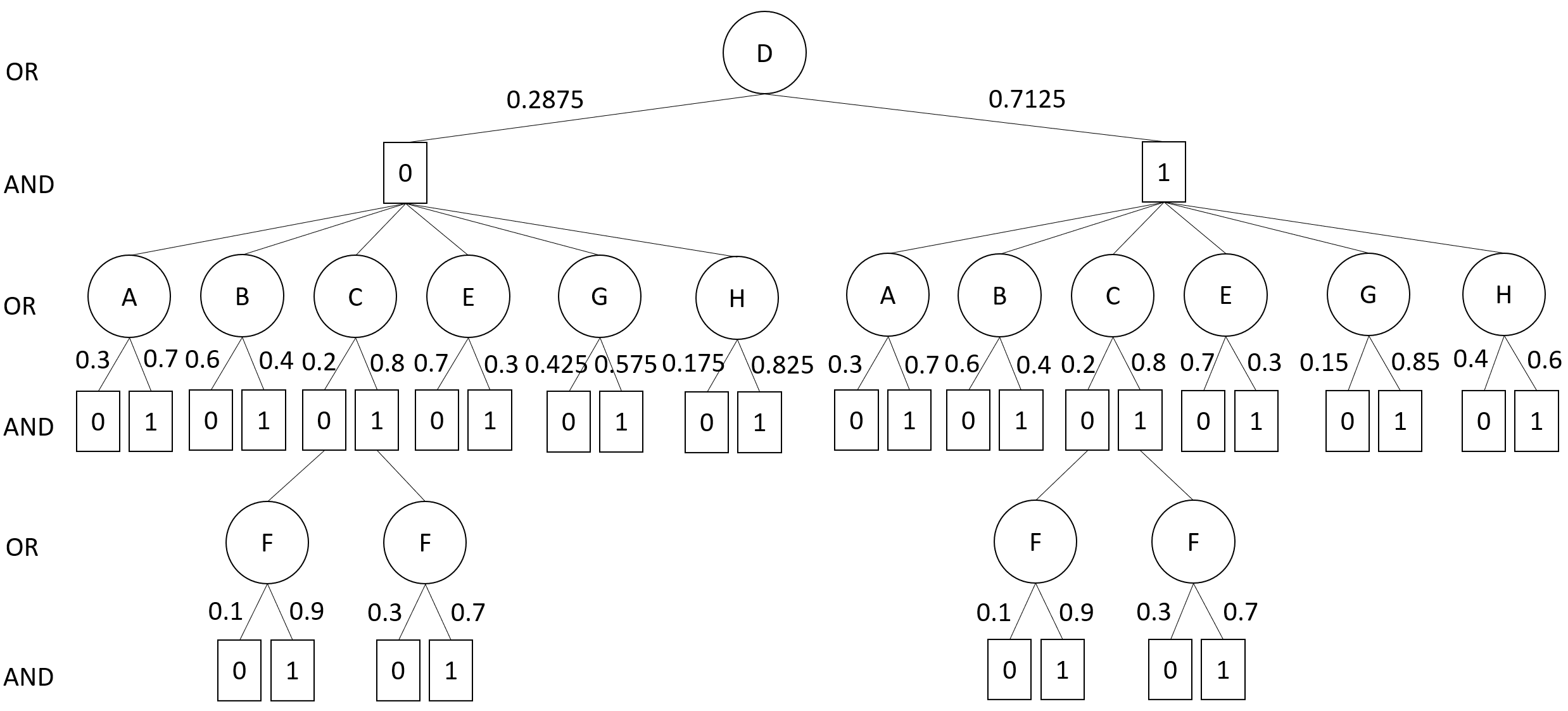
****

Thus, the depth of  is 2.

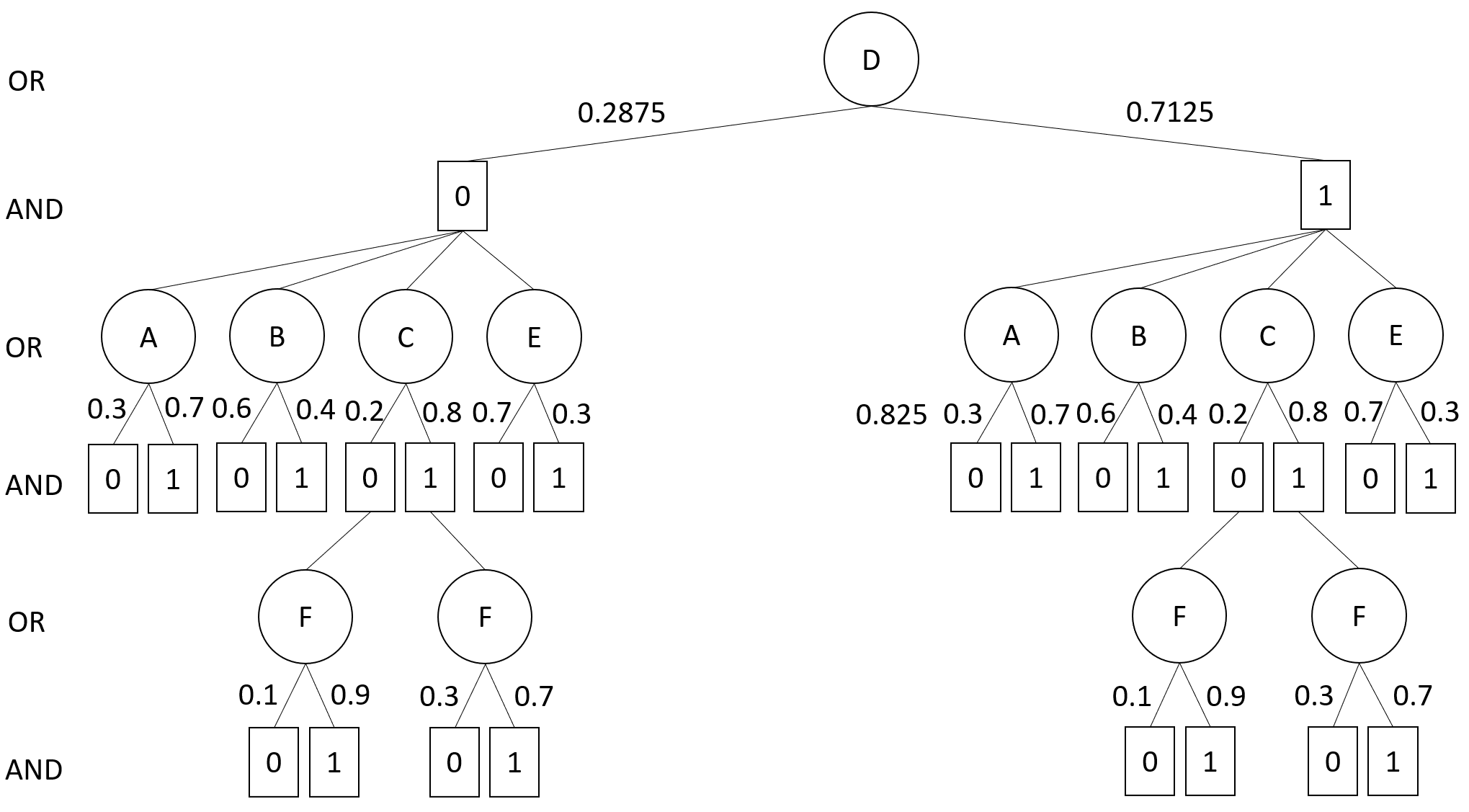
* The AND/OR search tree is as follows:



* The weights of arcs are as follows:

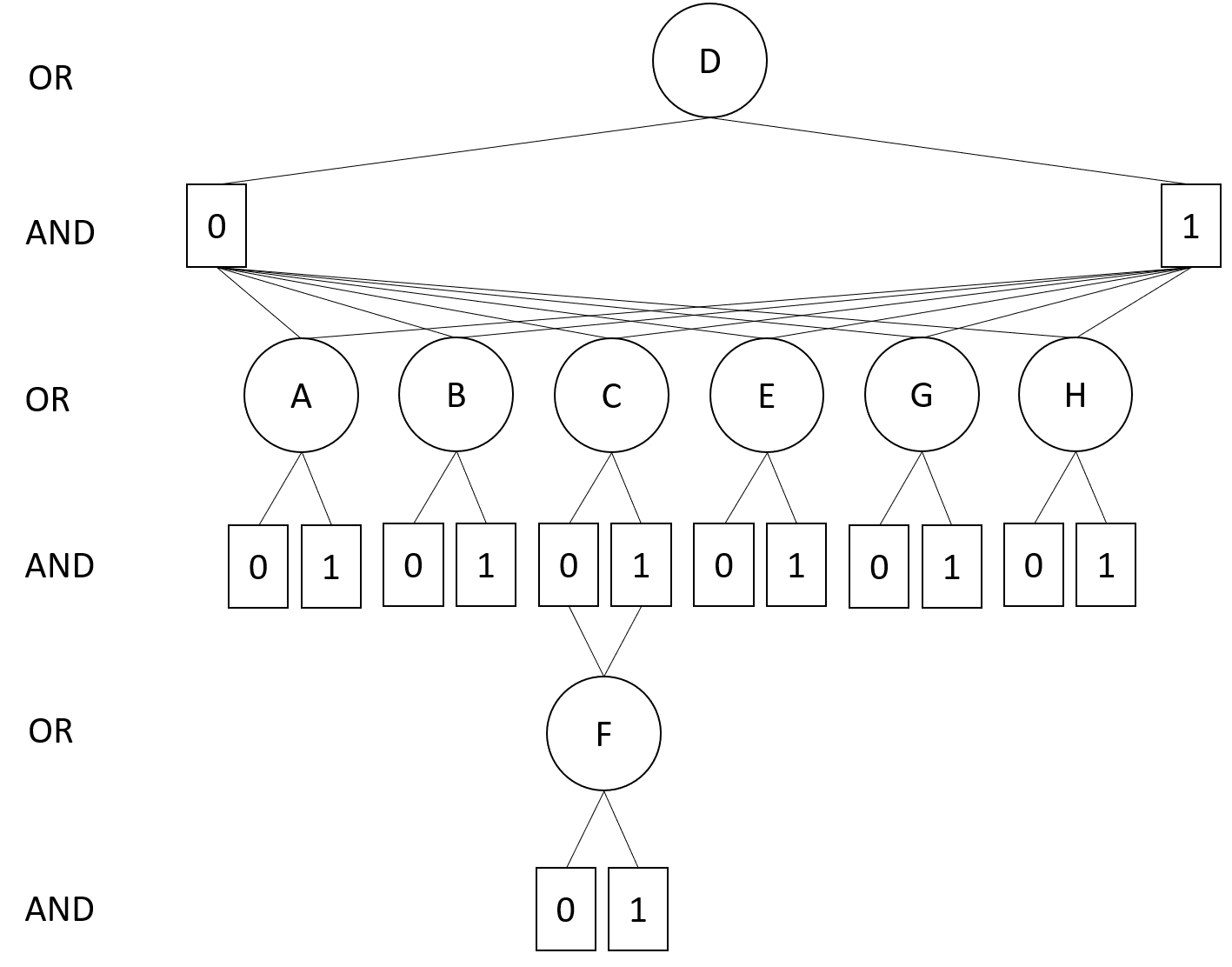
****

* First, after eliminating node and node , our AND/OR search tree will become:

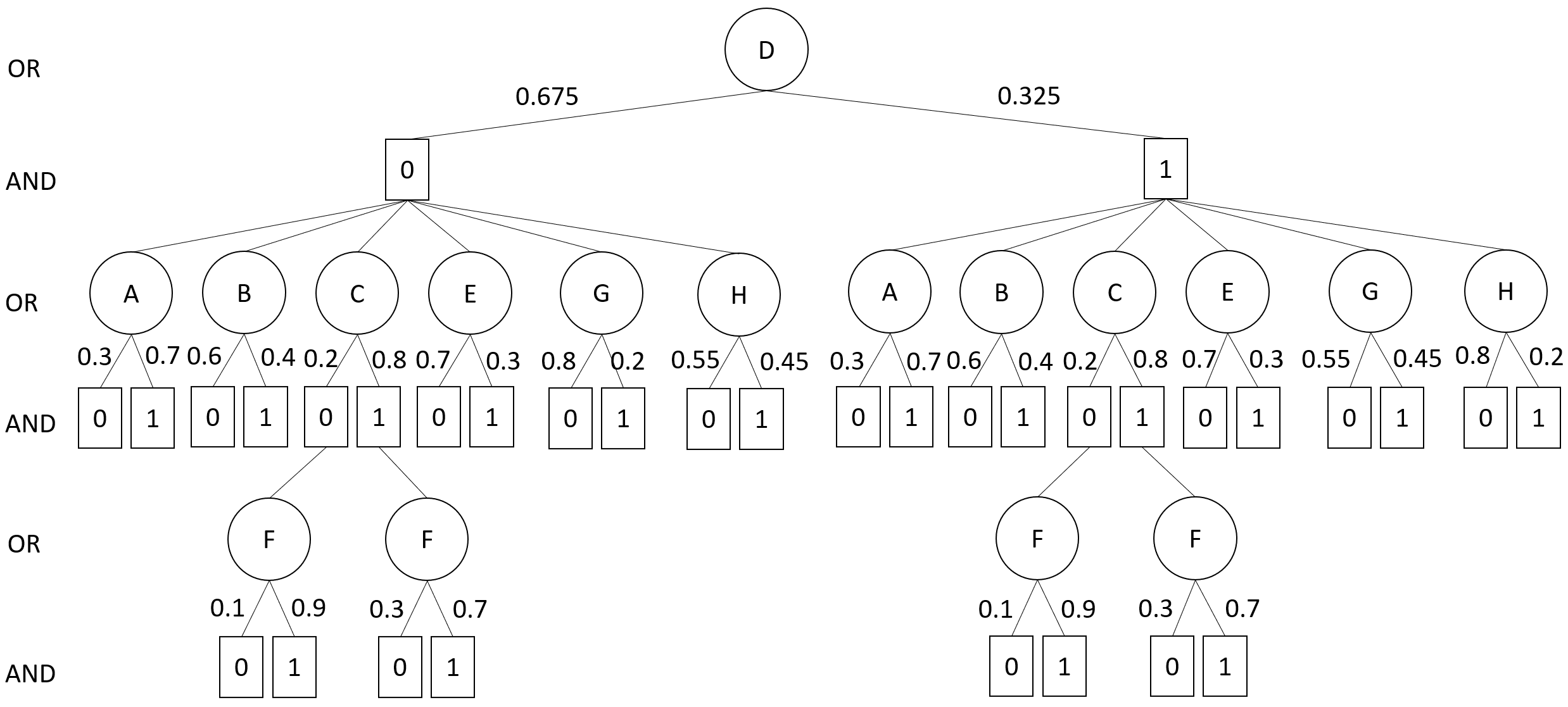
****

Second, we apply DFS on this AND/OR search tree. The visiting order will be . Let us denote as the domain size, as the number of variables, and as the induced-depth of the pseudo tree . There are variables in the network. The DFS algorithm will visit each node and arc once. Thus, the computational cost is In this case, the computational cost is .

* Yes, it can. We can merge those nodes that root identical subtree as shown below:



* The weights of arcs will change as shown below:



**Part 2: Inference [10 points]** Consider a chain Markov network . Provide an optimal algorithm which calculates for **all pairs** . Prove its optimality.

**Answer:**

According to the conditional probability theory, we know that . Where , . Thus, we can calculate by traversing from to using the chain dynamics in one pass. As we can calculate the probabilities in one pass, this algorithm is optimal.

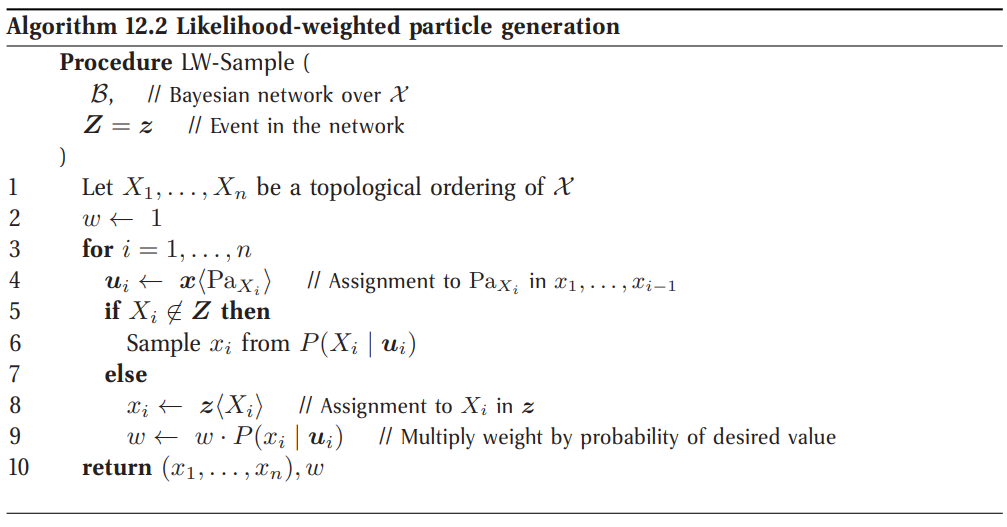
**Part 3: [20 points]** Exercise 12.9 from Koller and Friedman.

In this question, we consider the application of importance sampling to Markov networks.

1. Explain intuitively why we cannot simply apply likelihood weighting to Markov networks.
2. Show how likelihood weighting can be applied to chordal Markov networks. Is this approach interesting? Explain.
3. Provide a technique by which the more general framework of importance sampling can be applied to Markov networks. Be sure to define both a reasonable proposal distribution and an algorithmic technique for computing the weights.

**Answer:**

1. On one hand, because likelihood is a function P(A|B) which requires the dependence of data (e.g., A is dependent on B) while Markov networks are undirected and thus do not have such dependence. Thus, we cannot derive weights of samples from likelihood of evidence accumulated during sampling process. On the other hand, in Algorithm 12.2, line 4, we do not have parent-child relationship in Markov networks. Thus, we cannot simply apply likelihood weighting to Markov networks.



1. We can first convert chordal Markov networks into Bayesian networks and then apply likelihood weighting on the generalized Bayesian networks. Since for chordal graphs, a set of independences can be perfectly represented by both the Bayesian networks and the Markov networks, we can guarantee that the likelihood weighting on the Bayesian networks can have the same effect on the chordal Markov networks. Thus, this approach is interesting to implement.
2. In this problem, we provide with a normalized importance sampling on Markov networks. Let us denote and unnormalized version of . We want to sample from instead of and we do not know . while . We will generate samples from and assign weights , and normalize the estimator.

Algorithm:

For to do (sample n particles):

Sample from

Assign weight:

Approximation: . This is a biased estimator for finite (unbiased for ). Variance of estimator decreases linearly with sample size.

**Part 4: [10 points]** Exercise 12.13 from Koller and Friedman.

Show directly from equation (12.21) (without using the detailed balance equation) that the posterior distribution is a stationary distribution of the Gibbs chain (equation (12.22)).

**Answer:**

(12.21)

(12.22)

The distribution at time t can be represented as:

Equation (12.21) refers to the global transition model, which in the case of the Gibbs Chain corresponds to the probability of choosing a variable to flip and then applying the local transition model. The probabilities of flipping of all variables are the same. The variable is sampled from the distribution where is the assignment to all variables other than .

Notice in the case of the Gibbs chain, only one value is changed at a time.

**Part 5: [10 points]** In class, we saw that the Metropolis-Hastings algorithm converges to a stationary distribution if it satisfies the following detailed balance condition:

Show that if we use

then the detailed balance condition is satisfied.

**Answer:**

If we use , we will have .

Then,

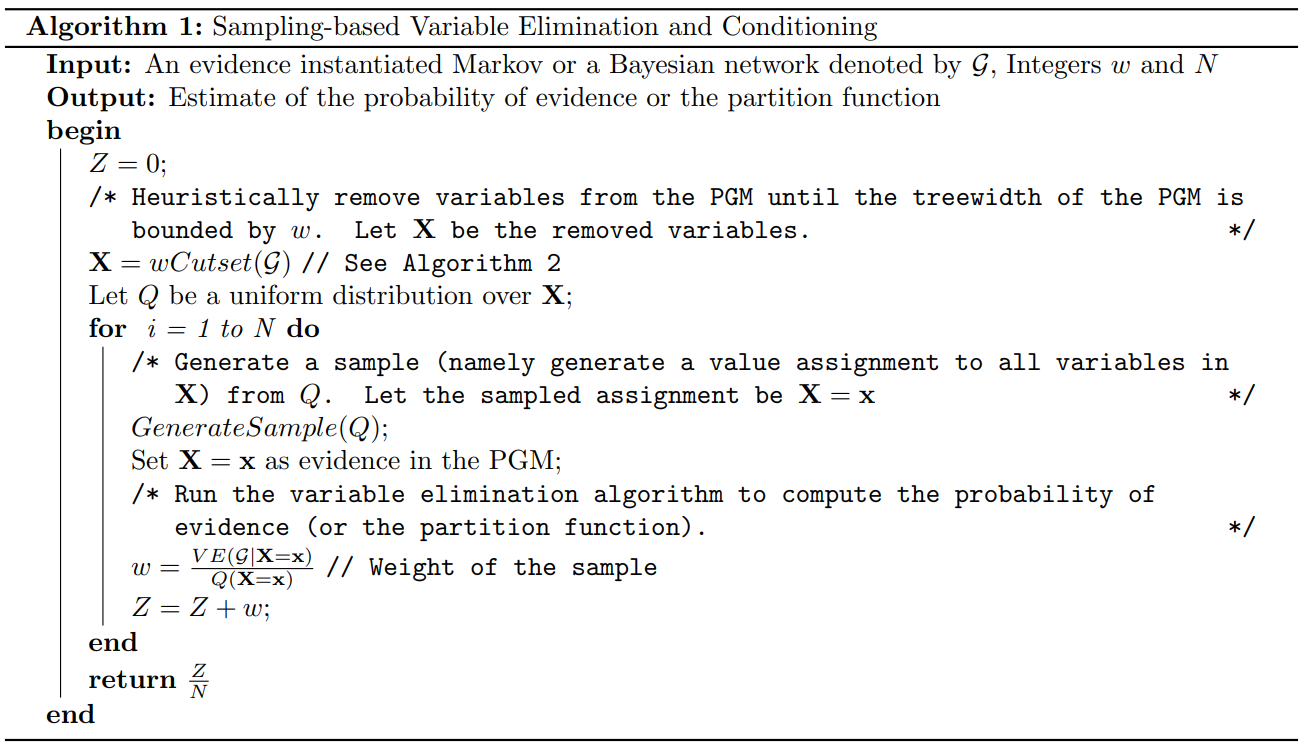
Thus, the detailed balance condition is satisfied.

**What to turn in for Parts 1-5?**

* A PDF containing your answers.

**Part 6: Programming [60 points]**

In this mini coding project, you will implement a “sampling-based version” of the variable elimination and conditioning algorithm (see Algorithm 1). The key idea in the algorithm is to compute the probability of evidence or the partition function by combining sampling with variable elimination (exact inference).



Assuming that you have already implemented the variable elimination algorithm, the hard part here is computing the w-cutset. You can use the following algorithm to compute the w-cutset.

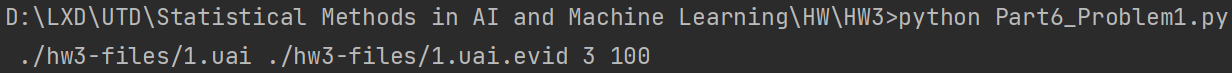
**What to do and turn in for Part 6?**

1. Write a program that implements Algorithm 1 and 2. The program should take the following inputs:
   * A Bayesian or Markov network in the UAI format
   * Evidence in the UAI format
   * An integer , which denotes the w-cutset bound
   * An integer N which denotes the number of samples

It should output an estimate of the partition function or the probability of evidence.

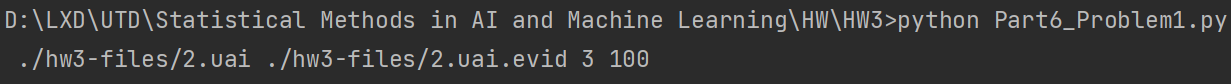
**Answer:** See Part6\_Problem1.py and README\_Problem1.txt

1.uai



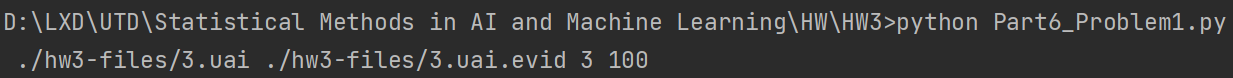


2.uai



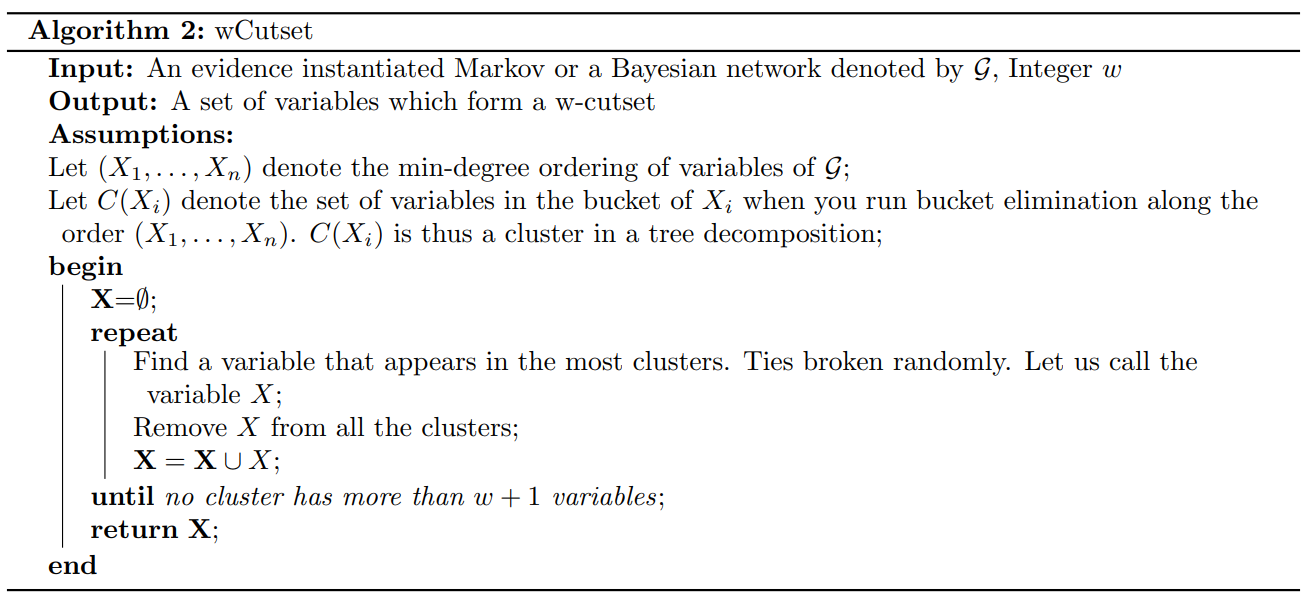


3.uai





1. Replace the proposal distribution by the following adaptive proposal distribution. Let the functional form of be . Namely, we assume that is an empty Bayesian network. Initialize to the uniform distribution and then after every 100 samples update each using the Monte Carlo estimate that we discussed in class. Formally, let be all the samples generated upto time

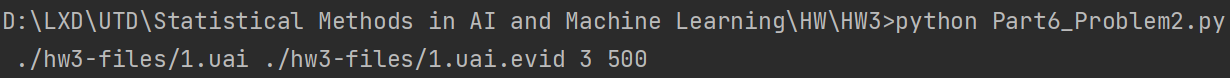


where is the dirac-delta function which is 1 when is present in and 0 otherwise and

Add an option to your program so that it will use the adaptive proposal distribution instead of the uniform distribution.

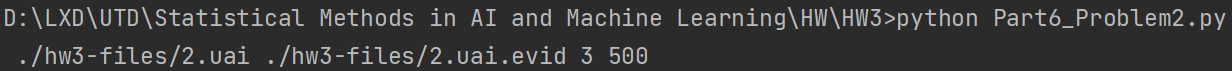
**Answer:** See Part6\_Problem2.py and README\_Problem2.txt

1.uai



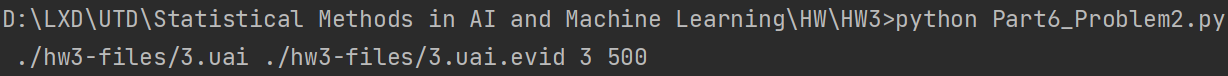


2.uai





3.uai





**For parts 1 and 2, turn in your source code and a readme file that describes how to compile and use your software.**

1. Run the two programs on the six PGMs given on the class web page. Try the following values for and and run each algorithm at least 10 times using different random seeds. For each run, compute the log-relative error between the exact partition function and the approximate one computed by your algorithm. Namely, compute

where is the exact answer and is the approximate answer.

For each PGM, report your results in a table such as the one given below and describe your findings in a few sentences (e.g., which method is better and why?; how and impact the accuracy, time-complexity and variance; etc.).

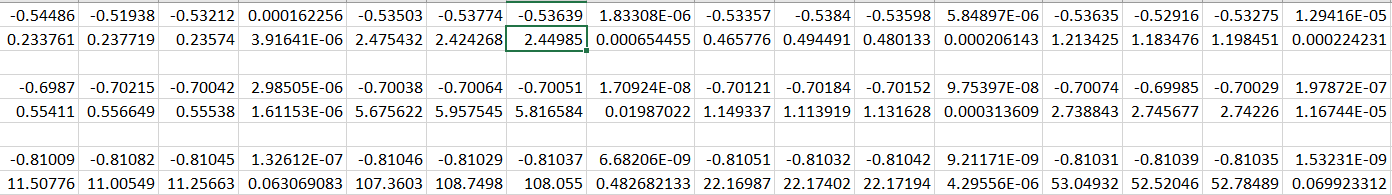
**Answer:** I cannot run my program on the given datasets because of the long running time and the overflow of the output. Thus, I run the experiments on previous datasets with 2 runs each experiment instead.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Problem-name |  |  | Uniform Proposal | | | | Adaptive Proposal | | | |
| N → | 100 | 1K | 200 | 500 | 100 | 1K | 200 | 500 |
| 1 | w=1  w=1 | Time  Error |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
| 2 | w=2  w=2 | Time  Error |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
| 3 | w=3  w=3 | Time  Error |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

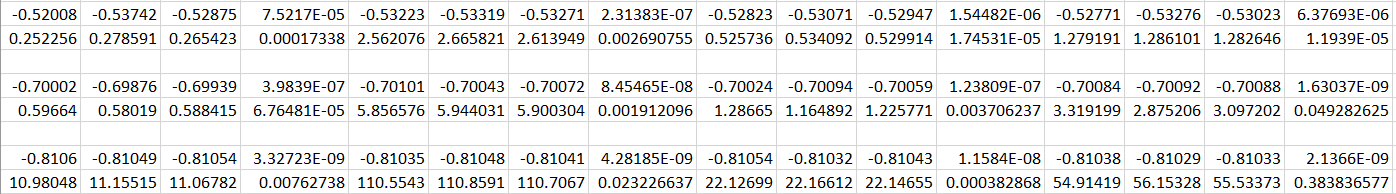
**See the table below:**

Order: Error, Time

Uniform Proposal:



Adaptive Proposal:



In the table, the time is in seconds. The quantity before is the average (e.g., and ) and the quantity after is the standard deviation (e.g., and ) over 10 runs (recall that you will run each algorithm 10 times using 10 different random seeds).

**For part 3, turn in a PDF or a Word file.**

**A useful tip.** Use a python script to set up your experiments. If you run each experiment individually by hand, you will take forever.