14.17 Consider the problem of generating a random sample from a specified distribution on a single variable. Assume you have a random number generator that returns a random number uniformly distributed between 0 and 1.
a. Let X be a discrete variable with P(X=xi)=pi for i ∈ {1, . . . , k}. The cumulative distribution of X gives the probability that X ∈ {x1, . . . , xj } for each possible j. (See also Appendix A.) Explain how to calculate the cumulative distribution in O(k) time and how to generate a single sample of X from it. Can the latter be done in less than O(k) time?
b. Now suppose we want to generate N samples of X, where N >> k. Explain how to do this with an expected run time per sample that is constant (i.e., independent of k).
c. Now consider a continuous-valued variable with a parameterized distribution (e.g., Gaussian). How can samples be generated from such a distribution?
d. Suppose you want to query a continuous-valued variable and you are using a sampling algorithm such as LIKELIHOODWEIGHTING to do the inference. How would you have to modify the query-answering process?

a. Let X be a discrete variable with P(X=xi)=pi for i ∈ {1, . . . , k}. The cumulative distribution of X gives the probability that X ∈ {x1, . . . , xj } for each possible j. (See also Appendix A.) Explain how to calculate the cumulative distribution in O(k) time and how to generate a single sample of X from it. Can the latter be done in less than O(k) time?

The probability distribution of variable X can be represented as: $< P(X = x_1),\ ...,\ P(X = x_k)\ >$
When the variable is discrete with ordered possible values X=xi with xi that belongs to {x1, x2, …, xk} (k possible values), then the cumulative distribution function could be defined as:

$$F_j = F(j) = P(X <= x_j) = \sum_{i=1}^{j} P(X = x_i)\ with\ 1 \leq j \leq k$$

The cumulative distribution of variable X can be represented as: $CD_X\ =\ < F_1,\ F_2,\ ...,\ F_k >$

If we pay attention, we can observe that:

1. $F_1 = P(X = x_1)$
2. $F_k = 1$ (by axiom I of probability theory)
3. $F_j = F_{j-1} + P(X = x_j)$ (for all other j where 1 < j <= k)

And that makes possible to calculate the cumulative distribution in O(k) time. We can build a loop that goes from $j = 2$ to $j = k - 1$ where we can obtain F_j in each step based on our already previously calculated or known values for F_{j-1} and $P(X = x_j)$.

$$CD_X =\ < F_1 = P_{x_1},\ F_2 = F_1 + P_{x_2},\ ...\ ,\ F_{k-1} = F_{k-2} + P_{x_{k-1}},\ F_k = 1 >$$

We can make another observation:

4. $F_j = F_{j+1} - P(X = x_j)$ (for all other j where 1 <= j < k)

And that makes possible to calculate the cumulative distribution in k/2 steps going from j=2 to k/2 and from j=k-1 to k/2 at the same time.

$$CD_X =\ < F_1 = P_{x_1},\ F_2 = F_1 + P_{x_2},\ ...\ ,\ F_{k-2} = F_{k-1} - P_{x_{k-2}},\ F_{k-1} = F_k - P_{x_{k-1}},\ F_k = 1 >$$

Other observation we can make is that:
$0 = F_0 < F_1 < F_2 < F_3 < \dots < F_k = 1$

And that the gap between two consecutive values F_j and F_{j-1} is the probability $P(X = x_j)$
$P(X = x_j) = F_j - F_{j-1}$

This gap let us take random samples of the distribution of variable X with a uniform random number generator. Suppose we generate a random number r between (0 and 1) using a uniform random number generator. We can take a sample using this approach:
$x_i = argmax(<1, 2, \dots, k> \ where\ F_{j-1} < r)$

We can use a simple but not optimized method that will loop from j=1 to j=k while the clause $F_{j-1} < r$ is true to generate the sample in O(k) time.

We can use a binary tree to improve the time performance of the sampling using $F_{k/2}$ as the root of the tree. Then the algorithm will require as much as $log_2(k)$ comparisons, so it will work in $O(log_2(k))$ time.

b. Now suppose we want to generate N samples of X, where N >> k. Explain how to do this with an expected run time per sample that is constant (i.e., independent of k).

We can use the uniform random generator to obtain a list with N random numbers r_i between 0 and 1.
$R_{samples} = [r_1, \dots, r_N]$
Then we create a list of tuples with using these random values and we reorder it in incremental order (based on the random values).
$O_{samples} = [(r_i, i), \dots, (r_j, j)]$
The r's subindex makes reference to the original position of r_i at list $R_{samples}$ and we save this index in the second term of the tuple. This job will be done in log(N) time, but we can ignore it.

Then we process the list $O_{samples}$ in incremental order one element at a time. We will read current random number in r_c that makes reference to the current random number being evaluated and at the same time the c value makes reference to the original order of the random number in the $R_{samples}$ list. We set the variable $i = 1$ at start. And then we loop: if current random number $r_c <= F_i$ we assign value x_i to the sample list $R_{samples}$ for element at position c. Then we increment variable i while current random number is bigger than the current cumulative value: $while\ i < k\ and\ r_c > F_i\ then\ i = i + 1$. This is repeated until all tuples in $O_{samples}$ is processed. This process will end with a list with N samples of variable X in random order.
$R_{samples} = [x_r, \dots, x_a, \dots, x_n, \dots, x_d, \dots, x_o, \dots, x_m]$

All these assignations will require of N steps. All the process for the N samples will work in O(N) time. If we use N steps to create N samples, we can conclude that the expected run time per sample is constant, then we can say that this algorithm will work in O(1) time to generate one sample.

c. Now consider a continuous-valued variable with a parameterized distribution (e.g., Gaussian). How can samples be generated from such a distribution?
If X is a continuous variable, then we know the probability density function at any fixed value is zero.

$P(X = x) = 0$ for all possible x that belongs to the continuous range of variable X.

The cumulative distribution function of variable X can be defined using the probability density function:

$F_X(x) = P(X \le x)$

So the CDF of variable X can be defined as the integral of the probability density function in range $(-\infty, x)$:

$$F_X(x) = \int_{-\infty}^{x} P(z)dz.$$

And if F_X is a continuous and strictly increasing function, then we can use its inverse F_X^{-1} to obtain samples with uniform random generated values. This implies that If r is a uniform random generated value, then the sample x is the resulting value of doing:

$x = F_X^{-1}(r)$ with $r \in (0, 1)$

F_X^{-1} is also known as a quantile function (special case).

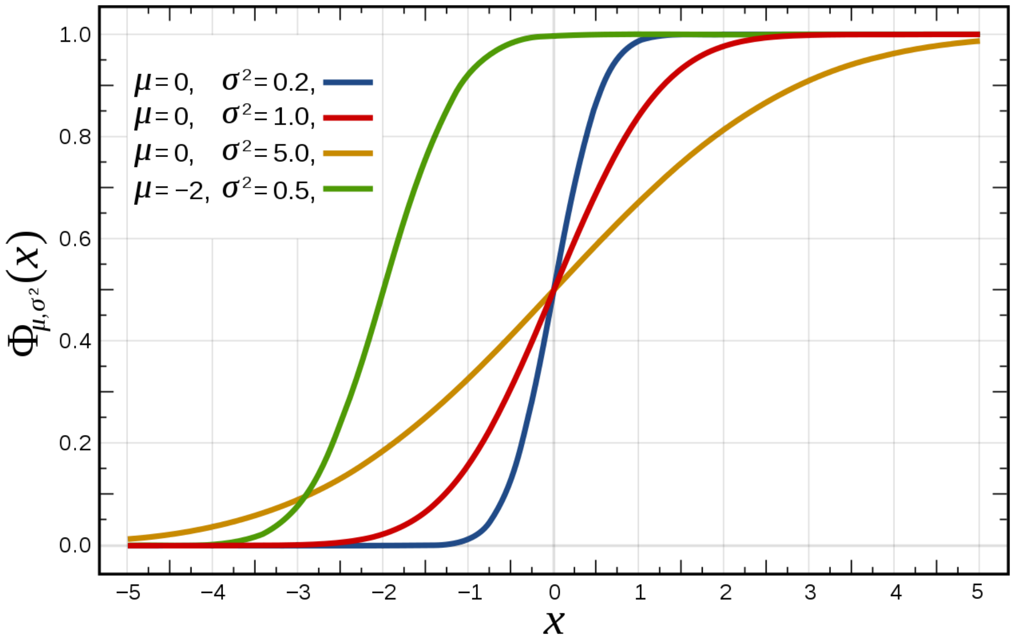
This is not so easy if we do not know a simple representation of the function F_X^{-1}. The exercise says to consider a Gaussian distributed variable.

Gaussian Probability Distribution

$$P(x) = \frac{1}{\sigma\sqrt{2\pi}} \times e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Gaussian Cumulative Probability Distribution

$$F(x) = \int_{-\infty}^{x} P(z)dz = \frac{1}{2} \times (1 + erf(\frac{(z-\mu)}{\sigma\sqrt{2}}))$$



So, we can write the inverse of a Gaussian Cumulative Probability distribution as:

$$F_X{}^{-1}(z) = \frac{(z-1/2)}{1+erf(\frac{(z-\mu)}{\sigma\sqrt{2}}))}$$

We can see that knowing the Gaussian parameters of the variable’s distribution we can obtain the Inverse of the Gaussian Cumulative Probability Distribution. But to be able to use this function we need to have access to another function: the Error Function.

This function is also known as the Gauss Error Function:

$$erf(x) = \frac{1}{\sqrt{\pi}} \times \int_{-x}^{x} e^{-t^2} dt \; = \frac{2}{\sqrt{\pi}} \times \int_{0}^{x} e^{-t^2} dt$$

(returns values between -1 and 1 for all x)

The error function is not an elementary function. We could make some approximations using the Maclaurin Series for e^{-t^2}.

$$erf(x) = \frac{2}{\sqrt{\pi}} \times \sum_{n=0}^{\infty} \frac{(-1)^n \times z^{2n+1}}{n! \times (2n+1)}$$

Another option is to use its Bürmann series to calculate it, but the important things is that if we resolve to get access to the error function, then we can easily use it to calculate the $F_X{}^{-1}(z)$ that will help at the same time to generate samples that we were looking for.

The question is what happens if we do not have access to the error function

We could build a discrete probability distribution using the Gaussian parameters. And then we can reuse the procedure described in exercise items a) and b) for generating samples. Bigger the k (the number of bins or the pieces in which we distribute infinites values :D) bigger the space we will require to generate our samples, but better the its accuracy too.

d. Suppose you want to query a continuous-valued variable and you are using a sampling algorithm such as LIKELIHOODWEIGHTING to do the inference. How would you have to modify the query-answering process?

“Each non-evidence variable is sampled according to the conditional distribution given the values already sampled for the variable’s parents”. This implies that we will need access to a distribution with no elements at the start, a distribution that does not exist. So, we would require to “bring it to real life”. If the variable has no parents, we already see that we could simple use the inverse cumulative distribution function to generate the sample. But if the variable presents at least one parent or its CDF is not invertible, then it could be better to work with a previously generated discrete distribution based on the continuous variable.

```
function LIKELIHOOD-WEIGHTING($X$, $\mathbf{e}$, $bn$, $N$) returns an estimate of $\mathbf{P}(X|\mathbf{e})$
  inputs: $X$, the query variable
          $\mathbf{e}$, observed values for variables $\mathbf{E}$
          $bn$, a Bayesian network specifying joint distribution $\mathbf{P}(X_1, \ldots, X_n)$
          $N$, the total number of samples to be generated
  local variables: $\mathbf{W}$, a vector of weighted counts for each value of $X$, initially zero

  for $j$ = 1 to $N$ do
      $\mathbf{x}, w \leftarrow$ WEIGHTED-SAMPLE($bn$, $\mathbf{e}$)
      $\mathbf{W}[x] \leftarrow \mathbf{W}[x] + w$ where $x$ is the value of $X$ in $\mathbf{x}$
  return NORMALIZE($\mathbf{W}$)
```

```
function WEIGHTED-SAMPLE($bn$, $\mathbf{e}$) returns an event and a weight

  $w \leftarrow 1$; $\mathbf{x} \leftarrow$ an event with $n$ elements initialized from $\mathbf{e}$
  foreach variable $X_i$ in $X_1, \ldots, X_n$ do
      if $X_i$ is an evidence variable with value $x_i$ in $\mathbf{e}$
          then $w \leftarrow w \times P(X_i = x_i \mid parents(X_i))$
          else $\mathbf{x}[i] \leftarrow$ a random sample from $\mathbf{P}(X_i \mid parents(X_i))$
  return $\mathbf{x}$, $w$
```

Figure 14.15 The likelihood-weighting algorithm for inference in Bayesian networks. In WEIGHTED-SAMPLE, each nonevidence variable is sampled according to the conditional distribution given the values already sampled for the variable's parents, while a weight is accumulated based on the likelihood for each evidence variable.