1. Prove that \( L \) is closed under Kleene star if and only if \( L = \text{NL} \).

**Solution** We (you) proved in the homework that \( \text{NL} \) is closed under Kleene star which gives us the direction \( L = \text{NL} \Rightarrow L \) closed under Kleene star. For the other direction, consider the following \( \text{NL} \)-complete language:

\[
\text{ORDERED-PATH} = \{ (G = (V,E), s,t) : \text{there is a path from } s \text{ to } t \text{ and } (v_i,v_j) \in E \Rightarrow i < j \}.
\]

The \( \text{NL} \)-hardness of this language is visible from the proof of \( \text{NP} \)-hardness of \( \text{PATH} \): a configuration graph is acyclic and therefore for each new node we add to the graph the given condition would hold.

Let’s now define

\[
B := \{ Gv_{i+1}Gv_{i+2}...Gv_j : (v_i,v_j) \in E(G) \}.
\]

\( B \in L \) (we can check whether \( G \) repeats checking it bit-by-bit without using any additional space). Furthermore, the string \( r = Gv_{s+1}Gv_{s+2}...Gv_t \in B^* \) if and only if there is a path from \( s \) to \( t \). (Indeed, if there is a path from \( s \) to \( t \), it is of the form \( v_i,v_{i+k_1},v_{i+k_2},...,v_j \), because of the ordered condition on graph \( G \). If \( r \in B^* \), there is a decomposition of \( r \) into

\[
(Gv_{s+1},Gv_{s+k_1},Gv_{s+k_2}...Gv_{s+k_3},...,Gv_{s+k_m}...Gv_{s+k_j})
\]

such that \( (v_i,v_{i+k_1}),(v_{i+k_1},v_{i+k_2}),...(v_{s+k_m},v_j) \in E \).) Therefore, given an instance of \( \text{ORDERED-PATH} \), we can transform it to an instance of \( B^* \), which is logspace computable (by the assumption of \( L \) being closed on Kleene star). This gives us \( L = \text{NL} \).

2. Define the class \( \text{PP}' \) similarly to \( \text{PP} \): \( L \in \text{PP}' \) if there exists a PPT \( N \) such that if \( \alpha \in L \), \( \text{Pr}[N(\alpha) = 1] > \frac{1}{2} \), but for \( \alpha \notin L \), \( \text{Pr}[N(\alpha) = 0] \geq \frac{1}{2} \) (in other words, \( \text{PP}' \) accepts by majority, but rejects by non-minority). Show that \( \text{PP} = \text{PP}' \).

**Solution** The direction \( \text{PP} \subset \text{PP}' \) follows from the definitions (a \( \text{PP} \) machine is at the same time \( \text{PP}' \) machine). Let \( L \in \text{PP}' \) and \( M' \) a \( \text{PP}' \) machine for \( L \). Let \( M' \) run in time \( \geq f(n), \forall(n) \). Now let’s design \( M \), a \( \text{PP} \) machine for \( L \) that accepts input \( \alpha \)

- toss a coin \( f(n) + 1 \) times
- if all tosses give tail then REJECT
- else
- simulate \( M' \) on \( \alpha \)
- end if
Consider both cases

- if $\alpha \in L$, $\mathbb{P}[M(\alpha) = 1] = \mathbb{P}[M'(\alpha) = 1] \cdot (1 - \frac{1}{2^{2^k-1}})$. Since $M'$ accepts with probability greater than 0.5, and there are $k$ branches, $k \leq f(n)$, we have the following: $\mathbb{P}[M(\alpha) = 1] \geq (\frac{1}{2} + \frac{1}{2^k}) \cdot (1 - \frac{1}{2^{2^k-1}}) = \frac{1}{2} - \frac{1}{2^{2^k-1}} + \frac{1}{2} - \frac{1}{2^{2^k-1}} > \frac{1}{2}$.

- if $\alpha \notin L$: $\mathbb{P}[M(\alpha) = 0] = \frac{1}{2^{2^k-1}} + \mathbb{P}[M'(\alpha) = 0] \cdot (1 - \frac{1}{2^{2^k-1}}) \geq \frac{1}{2^{2^k-1}} + \frac{1}{2} \cdot (1 - \frac{1}{2^{2^k-1}}) > \frac{1}{2}$.

Therefore, the constructed machine $M$ is also an PP-machine.

3. Define yet another similar class - PP''. $L \in \text{PP}''$ if there exists a PPT $N$ such that for $\alpha \in \Sigma^*$, $\mathbb{P}[N(\alpha) = L(\alpha)] \geq \frac{1}{2}$. Show that PP'' is not so similar to PP, rather PP'' = $\Sigma^*$.

**Solution** Take any language and let’s design machine $M$ for it. On input $\alpha$, simply toss one coin. If it is head, accept, else reject. $\mathbb{P}[L(\alpha) = M(\alpha)] = \frac{1}{2}$. We see that $L \in \text{PP}''$

4. For all $k \geq 1$, define class PP$_k$ such that $L \in \text{PP}_k$ if there exists a PPT $N$ such that

- $\alpha \in L \Rightarrow \mathbb{P}[N(\alpha) = 1] > 2^{-k}$
- $\alpha \notin L \Rightarrow \mathbb{P}[N(\alpha) = 0] \geq 1 - s^{-k}$

To phrase it differently, $N$ accepts $\alpha$ if and only if more than a fraction of $2^{-k}$ branches are accepting. Show that PP$_k$ = PP

**Solution** Note that PP$_1$ = PP According to previous exercise, It is enough to show PP$_k = \text{PP}$, for $k > 1$. In order to prove equality, we have to prove two inclusions.

- PP$_k \subset$ PP': Let $M_k$ be a PP$_k$ machine for a language $L$. We will try to describe a (PP') machine $M'$ on input $\alpha$.

  make $k+1$ coin tosses and treat them as binary representation of integer $r$, $0 \leq r \leq 2^{k+1} - 1$
  
  if $0 \leq r \leq 2^k - 2$ then
  accept
  else if $r = 2^k - 1$ then
  reject
  else if $2^k \leq r \leq 2^{k+1} - 1$ then
  simulate $M_k$ on $\alpha$
  end if
For \( \alpha \in L \),
\[
\mathbb{P}[M'(\alpha) = 1] = \frac{2^k - 1}{2^{k+1}} + \frac{2^{k+1} - 2^k}{2^{k+1}} \cdot \mathbb{P}[M_k(\alpha) = 1] \\
= \frac{1}{2} - \frac{1}{2^{k+1}} + \frac{1}{2} \cdot \mathbb{P}[M_k(\alpha) = 1] \\
> \frac{1}{2} - \frac{1}{2^{k+1}} + \frac{1}{2} \cdot \frac{1}{2^k} \\
= \frac{1}{2}
\]

For \( \alpha \notin L \),
\[
\mathbb{P}[M'(\alpha) = 0] = \frac{1}{2^{k+1}} + \frac{1}{2} \cdot \mathbb{P}[M_k(\alpha) = 0] \\
\geq \frac{1}{2^{k+1}} + \frac{1}{2} \cdot (1 - \frac{1}{2^k}) \\
= \frac{1}{2}
\]

• \( \text{PP}' \subseteq \text{PP}_k \) Now let \( M' \) be a \( \text{PP}' \) machine for language \( L \), and we’ll design a \( \text{PP}_k \) machine \( M_k \) for \( L \) (for any given \( k \)).

  make \( k \)-1 coin tosses
  if all outcomes are heads then simulate \( M' \) on \( \alpha \)
  else reject
  end if

Similarly as in the previous case we first observe the case when input \( \alpha \in L \)
\[
\mathbb{P}[M_k(\alpha) = 1] = \frac{1}{2^{k+1}} \cdot \mathbb{P}[M'(\alpha) = 1] \\
\geq \frac{1}{2^{k+1}} \cdot \frac{1}{2}
\]

Conversely, if \( \alpha \notin L \)
\[
\mathbb{P}[N_k(\alpha) = 1] = (1 - \frac{1}{2^{k+1}}) + \frac{1}{2^{k+1}} \cdot \mathbb{P}[M'(\alpha) = 0] \\
\geq 1 - \frac{1}{2^{k+1}} + \frac{1}{2^k} \\
= 1 - \frac{1}{2^k}
\]

5. Prove that \( \text{PP} \subseteq \text{PSPACE} \)

Let’s start from a language \( L \in \text{PP} \) and \( N \), a nondeterministic algorithm. We will try to create a deterministic version of the same algorithm that
uses only poly-space. Let \( p(n) \) be a polynomial bound on running time of \( N \). This also means that in the run of algorithm there are at most \( p(n) \) coin tosses. The algorithm works as follows

\[
\begin{align*}
c_0, c_1 & \leftarrow 0 \\
\text{for each outcome of } p(n) \text{ tosses do} \\
& \quad \text{run algorithm } N \text{ with current state of the tosses} \\
& \quad \text{if } N \text{ accepts then} \\
& \quad \quad c_0 \leftarrow c_0 + 1 \\
& \quad \text{else} \\
& \quad \quad c_1 \leftarrow c_1 + 1 \\
& \quad \text{end if} \\
\text{end for} \\
\text{if } c_0 > c_1 \text{ then} \\
& \quad \text{accept} \\
\text{else} \\
& \quad \text{reject} \\
\text{end if}
\end{align*}
\]

All runs of \( N \) reuse the space. \( c_0, c_1 < 2^{p(n)} \), which means that their binary representation is polynomial. Therefore, the whole algorithm runs in polynomial space. Notes

**Note** Define RP

6. Show that RP is closed under union and intersection

**Solution** Let \( M_1 \) and \( M_2 \) be the two RP machines deciding \( L_1 \) and \( L_2 \).

- **Intersection**: consider the following algorithm \((M_{\cap})\)
  
  \[
  \begin{align*}
  & \text{Simulate } N_1 \text{ on input } \alpha \\
  & \quad \text{if } N_1 \text{ rejects then} \\
  & \quad \quad \text{reject} \\
  & \quad \text{else} \\
  & \quad \quad \text{Simulate } N_2 \text{ on } \alpha \\
  & \quad \quad \text{if } N_2 \text{ accepts then} \\
  & \quad \quad \quad \text{accept} \\
  & \quad \quad \text{else} \\
  & \quad \quad \quad \text{reject} \\
  & \quad \text{end if} \\
  & \text{end if}
  \end{align*}
  \]

If \( \alpha \in M_1 \cap M_2 \), \( M_{\cap} \) accepts with probability \( \frac{1}{4} \). If \( \alpha \notin M_1 \cap M_2 \), \( M_{\cap} \) it would reject with probability 1.

- **Union**: Consider the following algorithm, \( M_{\cup} \):
  
  \[
  \begin{align*}
  & \text{toss a coin} \\
  & \quad \text{if head then} \\
  & \quad \quad \text{simulate } M_1 \text{ on the input } \alpha \\
  & \quad \text{else} \\
  & \quad \quad \text{simulate } M_2 \text{ on the input } \alpha
  \end{align*}
  \]
end if

In the case that $\alpha \in L_1 \cup L_2$, the probability of accepting equals
\[ \frac{1}{2} \cdot \frac{1}{2} + \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{2}. \]
Otherwise, $\alpha$ is not in $L_1$ nor in $L_2$, therefore there are no accepting branches of either $M_1$ or $M_2$ and the probability of accepting is again 0.

NOTE: Sources of the problems and more problems of the same kind:

- [http://blog.computationalcomplexity.org/2015/04/is-logarithmic-space-closed-under.html](http://blog.computationalcomplexity.org/2015/04/is-logarithmic-space-closed-under.html)
- [http://cse.iitkgp.ac.in/~abhij/course/theory/CC/Spring04/chap5.pdf](http://cse.iitkgp.ac.in/~abhij/course/theory/CC/Spring04/chap5.pdf)