Problem

Show that if NP $\subseteq$ BPP, then NP = RP.

Step-by-step solution

Step 1 of 2

As $RP \in NP$, it is sufficient to show $NP \subseteq RP$. It can be prove by showing that if $NP \subseteq BPP$, then $SAT \in RP$. Consider a formula ϕ with n variables $x_1, x_2, \ldots, x_n$ be the input. ϕ is satisfied iff $\exists$ truth assignment for $x_1, x_2, \ldots, x_n$ so that $\phi(x_1, x_2 \ldots x_n) = 1$.

• Assume A be a BPP algorithm with error probability at most 2^{-k} for SAT, where $k = |\phi|$ is the length of formula ϕ. Such A exist because of the assumption that $SAT \in BPP$.

• First run A on ϕ. If A rejects, then reject, otherwise try to satisfy assignment for ϕ on variable at a time.

• Initialize x_1 to 0 and then call A to determine that if the formula resulting is satisfiable: if A returns "accept" then permanently set x_1 to 0; otherwise set x_1 to 1. Then proceed with x_2 similarly.

• If manage to construct a satisfying assignment at end then verify this assignment for ϕ. If $\phi(x_1, \ldots, x_n) = 1$, then accept; otherwise reject.

• Here is the analysis. If ϕ is unsatisfiable, then always reject either because A rejects in the process or do not arrive at a satisfying assignment at the end.

Comment

Step 2 of 2

On the other hand, suppose ϕ is satisfiable. Proceed to show that it accept with probability at least $\frac{1}{2}$.

• Invoke A a total of $n+1$ times. If ϕ is satisfiable and A returns "accept" each time only for an assignment for variable x_i which is part of a satisfying assignment, then end up with a satisfying assignment.

• Now show that the probability that at least one of the $n+1$ invocations returns "reject" for an assignment for variable x_i which is part of a satisfying assignment is at most $\frac{1}{2}$.

• The probability that an invocation of A returns does so is at most $2-k$. So probability that encounter it is at most $(n+1).2^{-k}$, which is at most $\frac{1}{2}$ because $n+1 \le k$.

Since both the algorithm A and the construction of satisfying assignment run in polynomial time, the whole procedure clearly runs in polynomial time. Hence, $SAT \in RP$ and therefore $NP = RP$.

Comment