4/26/19

HomeWork 6

4. Given Hamiltonian path in a graph G. and it contains all vertices of G. To prove Ham-Path $\leq_p$ Ham-Cycle.

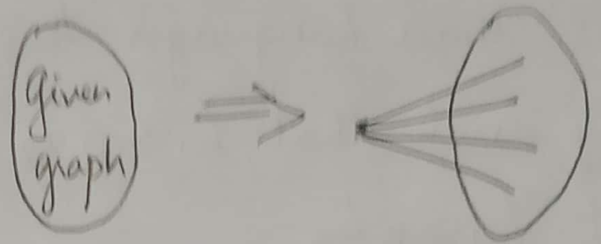
Given an undirected graph G we need to convert it to an undirected graph G' s.t. G contains a Hamiltonian path iff G'' contains a hamiltonian cycle.

- G'' consists of all nodes of G and an additional node v.

- G'' contains all edges G and additional edges of the form (x, v) for every vertex x of G.

If G contains m vertices and n edges then G will consist of m+1 vertices and 2m + n edges which are clearly polynomial in the input size.

Example of this algorithm can be seen as follows.



Now. we prove that the above reduction works.

- If G contains a hamiltonian path then G' consists of a hamiltonian cycle.

Let the Hamiltonian path of G be of form $(x1, x2)$ $(x2, x3), \ldots (x_{m-1}, x_m)$ then the hamiltonian cycle of G" will be of the form $(v, x1), (x1, x2), (x2, x3), \ldots$ $(x_{m-1}, x_m), (x_m, v)$.

- If G' contains a hamitonian cycle then G consists of a hamiltonian path.

Let the hamiltonian cycle of G' be of the form $(v, x1), (x1, x2), (x2, x3), \ldots (x_{m-1}, x_m), (x_m, v)$ then the hamiltonian path of G will be of the form $(x1, x2), (x2, x3) \ldots (x_{m-1}, x_m)$.

Thus we have reduced hampath to ham cycle and shown that ham-path $\leq_p$ ham-cycle.

Given that three languages, L_1, L_2 and L_3 over $\Sigma = \{0, 1\}$ such that $L_1 \leq_p L_2 \leq_p L_3$.

To prove that $L_1 \leq_p L_3$

Sol: ~~If problem~~ By definition, there are poly-time functions f and g such that $x \in L_1 \Leftrightarrow f(x) \in L_2$ and $y \in L_2 \Leftrightarrow g(y) \in L_3$, thus $x \in L_1 \Leftrightarrow f(x) \in L_2 \Leftrightarrow g(f(x)) \in L_3$. Obviously, $g(f(\cdot))$ is poly time (since $|f(x)|$ is polynomial in $|x|$).

If a polynomial time reduction from L_1 to L_2 running in time n^k and g is a polynomial time reduction from L_2 and L_3 computed in time n^l, then go from L_1 to L_3 and can be computed in time $O(n^k + (n^k)^l) = O(n^{kl})$.

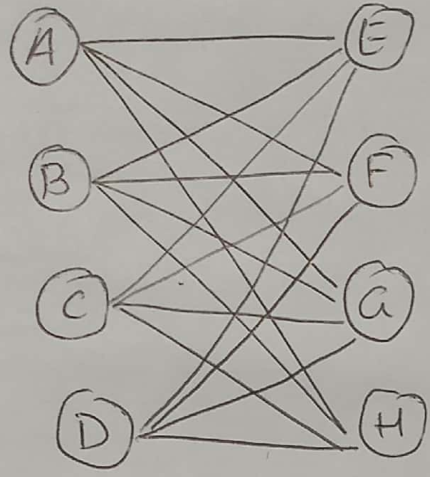
2.(a) Bipartite graph of 8 vertices:

Hamiltonian cycle is a cycle that visits each vertex exactly once but the start and end vertex is same. Thus the start vertex is visited twice.

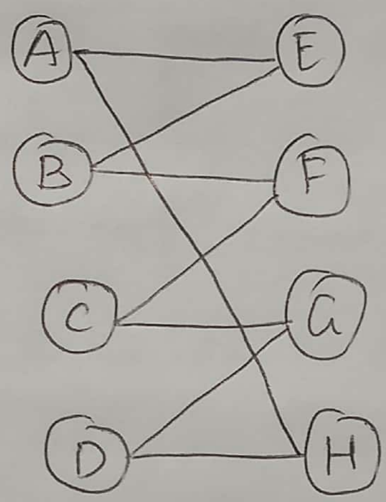
The graph that contains a hamiltonian cycle is called as hamiltonian graphs.

A bipartite graph is a graph whose vertices can be divided into two disjoint sets, such that there is edge joining every vertex of one set to every vertex of the other set. But there is no edge connecting vertices of the same set.

Given vertices = 8.



The corresponding hamiltonian graph starting from vertex 'A' is:

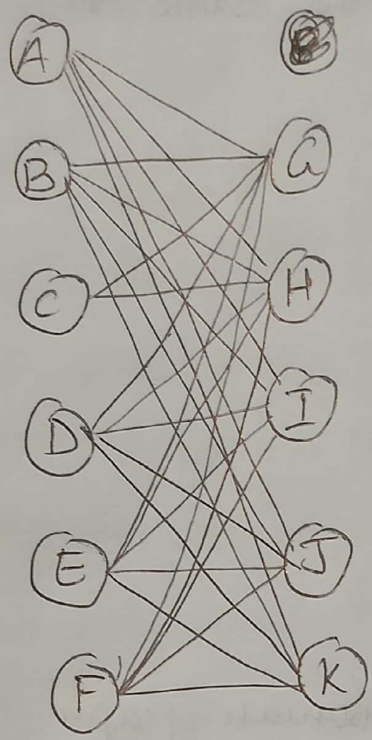


$A \rightarrow E \rightarrow B \rightarrow F \rightarrow C \rightarrow G \rightarrow D \rightarrow H \rightarrow A$

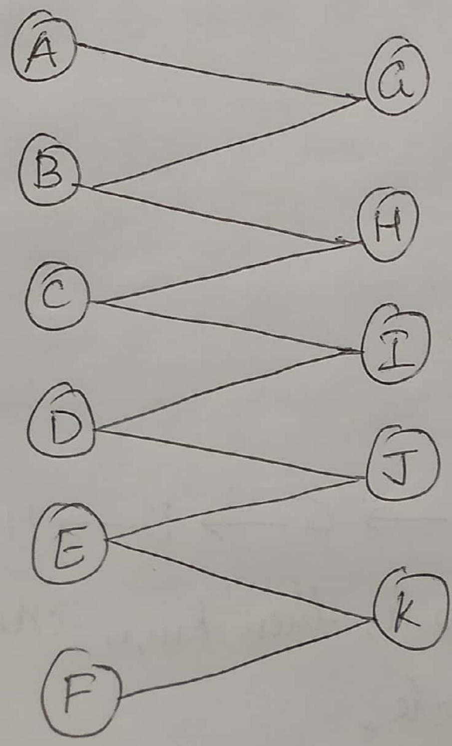
If m is not equal to n, then $K_{m,n}$ cannot have a hamiltonian cycle.

(b) Given vertices = 11,

If we take $m = 6$, $n = 5$



The hamiltonian graph is not possible as shown below.



The vertex starts with 'A' there is no way to come from 'K'. Thus there's no hamiltionian cycle.

3. a) Prove that Approx-SAT is in NP.

Given a CNF formula ϕ with $k \geq 2$ clauses $C_1 \ldots C_k$ and n variables $x_1 \ldots x_n$

The algorithm takes the inputs encoding and number of clauses i.e., ϕ, k and boolean assignment to ϕ.

⇒ The algorithm checks whether $k \geq 2$ and ϕ is in CNF algorithm then evaluates all the clauses. Algorithm is evaluated as true if and only if one clause is false, otherwise it returns false.

→ The algorithm runs in polynomial time.

Hence the approx-SAT is in class NP.

b) Let there be k-clauses in 3-SAT. This problem returns true if all k-clauses are true.

* Consider adding $k+1^{th}$ clause to ϕ, so that $k+1^{th}$ is a dummy clause and is always false.

* Approx-SAT remains true if k-1 clauses out of k are true. Now ϕ has k+1 clause after you added k+1th clause. When ϕ ~~ϕ~~ is added to approx-SAT, it will return ~~twice~~ true if k clauses are true and one is false.

* when comparing with 3-SAT, if variable returns true for APProx-SAT, then there is only one false clause, which implies all k-clauses are true.

* If approx-SAT returns false for some assignment then there are two or more claus that are false → one is dummy, other is in k-clauses.

* 3-SAT also returns false on this assignment of variables.

* We can say that if approx-SAT returns true 3-SAT is also true and if approx-SAT returns false, 3-SAT also returns false.

Hence 3-SAT $\leq_p$ Approx-SAT

5. Travelling Sales man problem.

Given: TSP for undirected graphs in NP-C

To prove: TSP for directed graphs is also NPC.

proof: 1. TSP for directed graph is in NP.

Where there exists a verification algorithm m

that decides TSP in polynomial time.

Given graph G(V,E) with k vertices cost

function f.

* Take the tour with each k' ∈ k & check whether cost of that path is equal to cost function given or not.

* Above algorithm goes towards every vertex k' ∈ k, hence is polynomial time in k.

② TSP for directed is NP-HARD. i.e., ∓

TSP (undirected) $\leq_p$ TSP (directed)

we take input of TSP (undirected) & convert it to the inputs of TSP

for TSP (undirected) to TSP (directed)

change all the edges like above

Now because, TSP(undirected) is NP-Hard so TSP (directed) is also more or equally hard than TSP (undirected)

Hence, TSP(undirected) $\leq_p$ TSP(directed)

So, TSP for directed is in NPC.