Weekly Assignment 11

Munia Humaira
Student ID: 21116435
Algorithm Design and Analysis

November 28, 2024

Solution 2:

The pseudocode for a non-deterministic polynomial-time algorithm for HamPath is:

```
HamPath(G, s, t):
path = [s]
current = s

for i = 1 to |V(G)| - 1:
    next = non_deterministically_choose(adjacent_vertices(G, current))
    if next is already in path:
        reject
    path.append(next)
    current = next

if current != t or |path| != |V(G)|:
    reject
accept
```

This algorithm runs in non-deterministic polynomial time.

Solution 3:

A polynomial-sized certificate for a true instance of HamPath would be the Hamiltonian path itself. Specifically, it would be a sequence of vertices $(v1, v2, ..., vn)$ where: $v1 = s$ (start vertex), $vn = t$ (end vertex) and $n = |V(G)|$ (number of vertices in G).

The verification algorithm would be:

```
Verify_HamPath(G, s, t, certificate):
if certificate[0] != s or certificate[-1] != t:
    reject

if len(certificate) != |V(G)|:
    reject

for i = 0 to |V(G)| - 2:
    if (certificate[i], certificate[i+1]) not in E(G):
        reject

if len(set(certificate)) != |V(G)|:
    reject

accept
```

This verification algorithm runs in polynomial time. It checks that the path starts at s, ends at t, contains all vertices exactly once, and that each consecutive pair of vertices in the path is connected by an edge in G.

Solution 4:

The pseudocode:

```
Algorithm HamPathConstruct(G):
n := number of vertices in G
V := list of vertices in G

if not H(G) then
    return string(epsilon)

for i := 0 to n-1 do
    for j := 0 to n-1 do
        if i not equal j and H(G, V[i], V[j]) then
            path := [V[i], V[j]]
            goto ConstructPath

ConstructPath:
for k := 2 to n-1 do
    for each v in V not in path do
        if H(G, path[0], v) and H(G, v, path[last]) then
            for i := 1 to length(path)-1 do
                if H(G, path[0], v) and H(G, v, path[i]) then
                    insert v into path at position i
                    break
            break

return path
```

The algorithm makes $\mathcal{O}(n^3)$ calls to the oracle H, where n is the number of vertices in G. Each call takes constant time $\theta(1)$, and the rest of the operations are also polynomial in n. Therefore, the overall time complexity is polynomial in the size of the input graph G.

Solution 5:

From textbook's claim 51: "**co-NP** is closed under $\leq_k$. That is, if $B \in$ **co-NP** and $A \leq_k B$, then $A \in$ **co-NP**."

Proof: Given: $B \in$ **co-NP** and $A \leq_k B$. Since $B \in$ **co-NP** , is complement B' is in **NP**. This means there exists a polynomial-time verifier V and a polynomial $p(n)$ such that for all x :

$x \in B'$ iff $\exists\ y$ ($|y| \leq p(|x|)$ and $V(x, y)$ accepts)

Let f be the polynomial-time reduction function from A to B. So, $x \in A$ iff $f(x) \in B$. We can construct a verifier V_A for A' as follows:

$V_A(x, y) = V(f(x), y)$

Now, $x \in A'$ iff $f(x) \in B'$ iff $\exists\ y$ ($|y| \leq p(|f(x)|)$ and $V(f(x), y)$ accepts) iff $\exists\ y$ ($|y| \leq p(|f(x)|)$ and $V_A(x, y)$ accepts)

V_A runs in polynomial time because f is a polynomial-time function and V runs in polynomial time.

The length of the witness y is polynomial in $|x|$ because $|y| \leq p(|f(x)|)$.

f is a polynomial-time function, so $|f(x)|$ is polynomial in $|x|$. The composition of two polynomials is still a polynomial. Therefore, we have shown that $A' \in$ **NP**, which means $A \in$ **co-NP**. This proves that **co-NP** is closed under Karp reductions ($\leq_k$).