HQ 20

(a) Reduction

```
def reduce ( G) :
    G' = empty graph
    for every vertex v in G:
        copy v in G'
    for every edge e in G:
        copy e in G'.
    add a vertex v' in G'
    return G'
```

(b) Complexity analysis

As we are creating a new graph G' from G by copying its vertices (in 1st for loop) and its edges (in 2nd for loop) we require $O(V+E)$ time complexity. Assuming $G(V,E)$ has $|V|$ vertices and $|E|$ edges.

(c) Proof of correctness

Lemma

we will get a Yes instance of UHAMPATH iff we get a Yes instance of UHPBUTT1.

proof

Let $v_1 - v_2 - v_3 - \cdots v_k$ is a hamiltonian path in G, then $v_1 - v_2 - v_3 \cdots v_k$ is a path in G' that also visits each vertex from v_1 to v_k exactly once but leaves out one vertex v', which is not visited. It is so because v' is not connected by any edge to the other vertices in G'. Thus a yes instance of UHAMPATH gives a YES instance of UHPBUTTI

Let $v_1 - v_2 - \cdots v_k$ be a path in G' s.t v' is ~~never~~ left unvisited (and rest all vertices are visited exactly once during the path. Let k total no. of vertices). Then $v_1 - v_2 - \cdots v_k$ is a hamiltonian path in G since v' was simply removed we can traverse all the vertices from v_1 to v_k exactly once in G as well, as G' is a copy of G except one vertex. Thus yes instance of UHPBUTTI gives a yes instance of UHPBUTTI

(b) The reduction of UHPBUTI to UHAMPATH is not a polynomial time reduction.

As we are taking permutation of all vertices (except one vertex) the complexity reaches to O(n!), where n= no. of vertices in graph G'.

The reduction of UHPBUTI to UHAMPATH is an incorrect one.

This is so because in the reduction algorithm when P forms a valid path we return a triangle graph and if it doesn't we return a graph with 3 edges and 4 vertices {a-b, b-c, c-d}. Now both these graphs will always give a ~~valid~~ hamiltonian path. Thus UHAMPATH problem will never have a NO instance. Thus the lemma : ~~To~~ we get a Yes instance of UHPBUTI iff we get a yes instance of UHAMPATH, is incorrect.

~~Running time complexity of reduction algorithm is~~

Running time complexity of the reduction algorithm is as follows —

$O(v)$ — ~~for~~ for loop running v times

~~$O(v!)$~~ $O((v-1)!)$ — for permutation of the vertices, except one

$O(v)$ — ~~for~~ to check if there is an edge between subsequent pair of vertices

$O(V+E)$ — to copy G ~~to~~ for constructing G' ie copying all vertices and edges.

time complexity ⇒ $\cancel{O(v) + O(v!) + O(v) + O(V+E)}$

$\cancel{\approx O(v}$

$V\left[O(V+E) + O((v-1)!) + O(v)\right]$

$\Rightarrow \cancel{O(V\,V!)} \quad \underline{\underline{O(v(v-1)!)}}$