**Problem 1**

If Prob1 Prob2 and Prob2 Prob3, then Prob1 Prob2

Let p(y) be a polynomial and C an algorithm, I1 is an instance of Prob1 with input data X1 of size n, C produces in O(p(n)) time input data X2 for I2 of Probl2. So I1 has a solution iff I2 has a solution.

Same with q(y) be a polynomial and D an algorithm, J2 is an instance of Prob2 with input data Y2 of size n, D produces in O(p(n)) time input data X3 for J3 of Probl3. So J2 has a solution iff J3 has a solution.

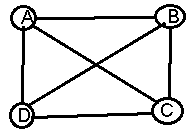
Let r(y) = q(p(y), with r(y) is a polynomial.

Define an algorithm E that accepts input data for instances of Prob1

For each input data X1 of I1 of Probl1, E performs the steps of C to produce input data X2 for instance I2 so that I1 has a solution iff I2 has a solution. And so on, I1 has a solution iff I3 has a solution.

**Problem 2**

From HC instance G = (V, E), adds missing edges to G to produce a complete graph H = (VH, EH), and then define c: H N by c(e) = 0 if e E, but c(e) = 1 if e E. Finally, let k = 0. Our TSP instance is (H, c, k).



1. So now solution for HC —> will be the solution for TSP instance

2. Solution for TSP —> will be the solution for HC instance

**Problem 3**

Suppose R is an NP problem.

We must show that R TSP.

We have R HC TSP

The first is because HC is NP-complete; the second is because of the fact shown previously.  
  
It now follows from Problem 1 that R TSP.

Alternatively, just show HC TSP and show that TSP in in NP.

Since HC is an NP-Complete problem and it can be reduced to TSP and TSP belongs to NP, TSP is an NP-Complete problem.

**Problem 4**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| N[0] |  |  |  |  |  |  |  |  |  |  |  |
| N[1] |  |  |  |  |  |  |  |  |  |  |  |
| N[2] |  |  |  |  |  |  |  |  |  |  |  |
| … |  |  |  |  |  |  |  |  |  |  |  |
| N[n] |  |  |  |  |  |  |  |  |  |  |  |

O (n) algorithm

* Using the memorization table for the SubsetSum problem given by N array and the number sum k = 10
* So the table will be filled from row N[0] to N[N.length - 1]
* Continue that way and look at the cell to see if there is any cell that contains the number whose sum is 10. Then we found the solution for it
* The running time to fill in to the data table is O(kn) which is O(10n) —> O(n)

**Problem 5**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 |
| (3) S0 | {} | NULL | NULL | {3} | NULL |
| (2) S1 | {} | NULL | {2} | {3} | NULL |
| (1) S2 | {} | {} | {2} | {3} | {3,1} |
| (5) S3 | {} | {1} | {2} | {3} | {3,1} |