FIT1045 Algorithmic Problem Solving – Tutorial 12.

Objectives

The **objectives of this tutorial** are:

- To be able to find vertex covers and cliques in graphs.
- To show that a given decision problem is in **P** or **NP**.
- To understand the **P** vs **NP** problem.
- To discuss how programs can be encoded in bits.
- To discuss the problem of deciding if a given algorithm terminates.

Task 1

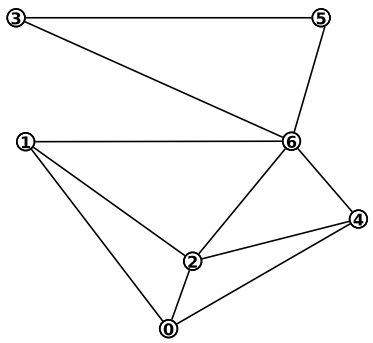

Figure 1: Graph H

Part A

Find a vertex cover for the graph in Figure 1.

Part B

What is the size of a largest clique in the graph in Figure 2?
What is the size of a smallest clique in this graph?

Part C

The graph $\overline{H}$ is the complement of the graph H if $\overline{H}$ and H have the same vertices and every distinct pair of vertices, u and v, are adjacent in $\overline{G}$ if and only if they are not adjacent in G. For example, the graph in Figure 2 is the complement of the graph in Figure 1.

Consider the following decision problems:
VERTEX COVER
Input: Graph G and number k
Question: Does G have a vertex cover of size k?

CLIQUE
Input: Graph G and number k
Question: Does G have a clique of size k?

These problems are in **NP**.

Suppose you had an algorithm to solve **VERTEX COVER**. Explain how you could solve **CLIQUE** using this algorithm.

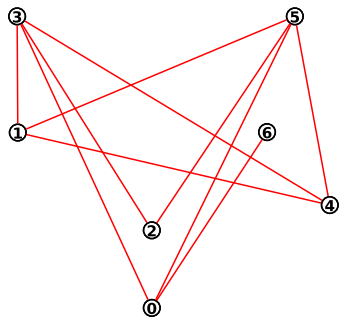

Figure 2: Complement of Graph H

Task 2

Part A

Give examples of decision problems in **P** or **NP**.

Part B

Discuss the **P** vs **NP** problem.

Draw a diagram that portrays our current view of this problem.

Part C

A decision problem is in **P** if it can be *solved* in polynomial time. A decision problem is in **NP** if any "yes"-answer can be *verified* in polynomial time.

Show that the following decision problem is in **P**:
SEARCH
Input: List L and target item
Question: Is the target item in the list L?

Show that the decision problem **CLIQUE** is in **NP**.

Part D

Give the complexity class of the decision version of the Knapsack, n-Queens and Travelling Salesperson problems.

Task 3

Table 1 gives part of the ASCII table. We can encode words by encoding each character by its ASCII value. Numbers can be encoded in their binary format. Instructions in the computer can also be encoded in this way. This gives us a way of representing a program as a list of bits.

Part A

Write an algorithm to convert an integer to binary.

Part B

Write an algorithm to convert an ASCII number to characters. [Refer Table 1].

Part C

Using Part B convert the following into characters.

```
65 108 103 111 114 105 116 104 109 32 67 111 108 108 97 116 122 40 110 41 10
119 104 105 108 101 32 40 110 62 49 41 32 32 123 10
32 112 114 105 110 116 32 110 10
32 105 102 32 40 110 32 37 32 50 32 61 61 32 48 41 32 123 110 60 45 110 47 50 125 10
32 101 108 115 101 32 123 110 60 45 51 110 43 48 125 10
125
```

Can you decide if this process will always terminate?

Symbol	ASCII	Binary Representation
A	65	01000001
B	66	01000010
C	67	01000011
D	68	01000100
E	69	01000101
F	70	01000110
...	...	...
Z	90	01011010
a	97	01100001
b	98	01100010
c	99	01100011
d	100	01100100
e	101	01100101
f	102	01100110
g	103	01100111
h	104	01101000
...	...	...
z	122	01111010
(	40	00101000
)	41	00101001
>	62	00111110
<	60	00111100
{	123	01111011
}	125	01111101
\	92	01011100
+	43	00101011
%	37	00100101
-	45	00101101
0	48	00110000
1	49	00110001
2	50	00110010
...	...	...
9	57	00111001
LF	10	00001010
Space	32	00100000

Table 1: Some of the ASCII table.

Puzzle of the week

Suppose that we have a collection of small villages in a remote part of the world. We would like to locate radio stations in some of these villages so that messages can be broadcast to all of the villages in the region. Since

each radio station has a limited broadcasting range, we must use several stations to reach all villages. But since radio stations are costly, we want to locate as few as possible so that all the villages can receive the broadcasts. How can we solve this problem?