Vertex Cover

Vincenzo Petrolo

November 16, 2019

Chapter 1

Data Structures

```
typedef struct {
int u;
int v;
}arco_non_orientato_t;

typedef enum {FALSE,TRUE} boolean;
```

Chapter 2

Prototypes

```
boolean vertex_cover(arco_non_orientato_t **,int **,int **,int ,int );
void powerset(int , int *, int *, int ,arco_non_orientato_t **,int );
void display_set(int **,int ,int **);
void leggi_archi(arco_non_orientato_t **,int *,int *);
boolean appartiene_a_W(arco_non_orientato_t ,int *,int ,int *);
```

2.1 Functions description

`vertex_cover(4);`

This is the main function of the code, it calls the functions

`appartiene_a_W(4)`

to compare arches to the subset generated from

`powerset(6).`

`powerset(6);`

This function has the purpose to create recursively vertexes subsets such that for every subset it is verified if it is a vertex cover.

`void display_set(3);`

The scope of this functions is to print in STDOUT the vertex subset which verfies the vertex cover.

```
void leggi_archi(3);
```

The aim of this functions is to read from a file which name is stored in FILENAME and to store datas in the data structure.

```
boolean appartiene_a_W(arco_non_orientato_t ,int *,int ,int *);
```

This function returns TRUE if given an arch and a subset, the arch belongs to the subset.

Chapter 3

Worst case complexity analysis

```
vertex_cover(4);
```

The function that verifies wether the given subset of vertexes is a vertex cover or not. The functions has inside itself two for loops one inside the other, this causes complexity of this function being $T(n) = O(n^2)$, where n is the size of the array containing vertexes