Universidad Nacional del Altiplano

Facultad de Ingeniería Mecánica Eléctrica, Electrónica y de Sistemas

Escuela Profesional de Ingeniería de Sistemas

Programación Competitiva

Coloración de Grafos (Welsh- Powell, Matula y Greedy)

Presenta: Jorge G. Olarte

Código: 215167

Docente: Ing. Alodia Flores Arnao

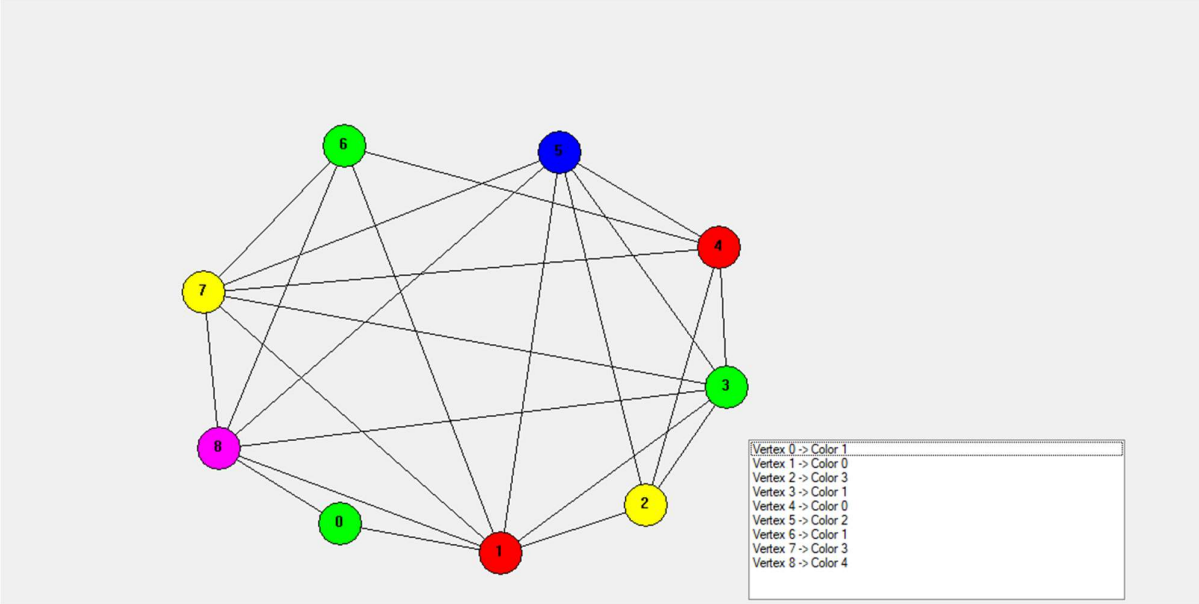
Puno-Perú

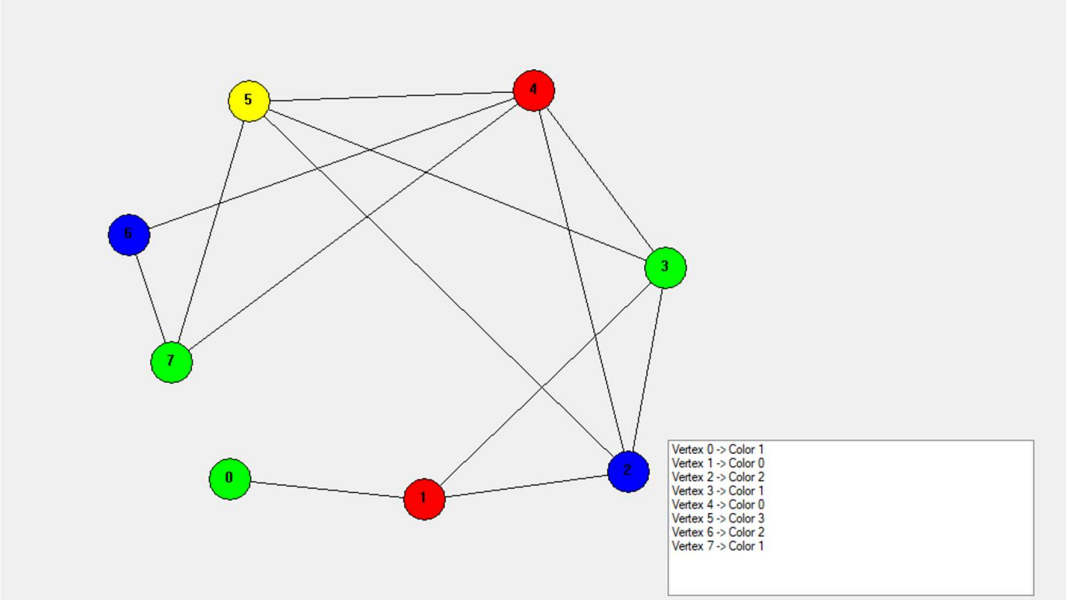
2024

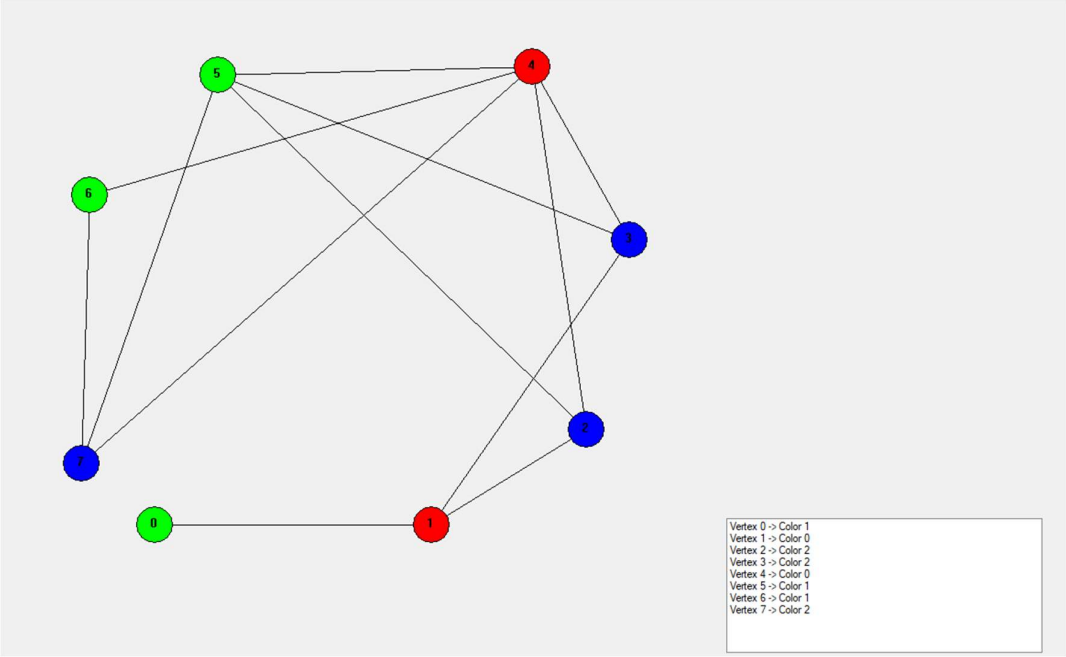
Link del Repositorio: https://github.com/ArtStyle19/Coloracion-de-grafos.git

Hecho con C++ (winAPI).

Se puede arrastrar los (estos aparecen de Manera aleatoria):






5
4
6
3
2
7
0
1
Vertex 0 -> Color 1
Vertex 1 -> Color 0
Vertex 2 -> Color 2
Vertex 3 -> Color 2
Vertex 4 -> Color 0
Vertex 5 -> Color 1
Vertex 6 -> Color 1
Vertex 7 -> Color 2

```
68  void WelshPowell() {
69      int degrees[MAX_VERTICES];
70      int order[MAX_VERTICES];
71      for (int i = 0; i < numVertices; i++) {
72          degrees[i] = 0;
73          for (int j = 0; j < numVertices; j++) {
74              if (graph[i][j]) degrees[i]++;
75          }
76          order[i] = i;
77      }
78      for (int i = 0; i < numVertices - 1; i++) {
79          for (int j = i + 1; j < numVertices; j++) {
80              if (degrees[order[i]] < degrees[order[j]]) {
81                  int temp = order[i];
82                  order[i] = order[j];
83                  order[j] = temp;
84              }
85          }
86      }
87
88      for (int i = 0; i < numVertices; i++) {
89          colors[i] = -1;// sin color
90      }
91
92      int currentColor = 0;
93      for (int i = 0; i < numVertices; i++) {
94          int u = order[i];
95          if (colors[u] == -1) {
96              colors[u] = currentColor;
97              for (int j = i + 1; j < numVertices; j++) {
98                  int v = order[j];
99                  if (colors[v] == -1) {
100                     int canColor = 1;
101                     for (int k = 0; k < numVertices; k++) {
102                         if (graph[v][k] && colors[k] == currentColor) {
103                             canColor = 0;
104                             break;
105                         }
106                     }
107                     if (canColor) {
108                         colors[v] = currentColor;
109                     }
110                 }
111             }
112             currentColor++;
113         }
114     }
```

Matula

```
91  int minDegreeVertex(int degrees[], int colored[]) {
92      int minDegree = MAX_VERTICES;
93      int minVertex = -1;
94      for (int i = 0; i < numVertices; i++) {
95          if (!colored[i] && degrees[i] < minDegree) {
96              minDegree = degrees[i];
97              minVertex = i;
98          }
99      }
00      return minVertex;
```

```
void MatulaMarbleIsaacson() {
    int degrees[MAX_VERTICES];
    int colored[MAX_VERTICES] = {0};
    for (int i = 0; i < numVertices; i++) {
        degrees[i] = 0;
        for (int j = 0; j < numVertices; j++) {
            if (graph[i][j]) degrees[i]++;
        }
    }

    int currentColor = 0;
    for (int i = 0; i < numVertices; i++) {
        int u = minDegreeVertex(degrees, colored);
        if (u == -1) break;

        colors[u] = currentColor;
        colored[u] = 1;
        for (int j = 0; j < numVertices; j++) {
            if (graph[u][j]) degrees[j]--;
        }

        for (int j = 0; j < numVertices; j++) {
            if (!colored[j] && !graph[u][j]) {
                int canColor = 1;
                for (int k = 0; k < numVertices; k++) {
                    if (graph[j][k] && colors[k] == currentColor) {
                        canColor = 0;
                        break;
                    }
                }
                if (canColor) {
                    colors[j] = currentColor;
                    colored[j] = 1;
                    for (int k = 0; k < numVertices; k++) {
                        if (graph[j][k]) degrees[k]--;
                    }
                }
            }
        }

        currentColor++;
    }
}
```

\

Greedy O voraz

```
50  void GreedyColoring() {
51      for (int i = 0; i < numVertices; i++) {
52          colors[i] = -1;
53      }
54
55      colors[0] = 0;
56
57      bool available[MAX_VERTICES];
58      for (int i = 0; i < MAX_VERTICES; i++) {
59          available[i] = true;
60      }
61
62      for (int u = 1; u < numVertices; u++) {
63          for (int i = 0; i < numVertices; i++) {
64              if (graph[u][i] && colors[i] != -1) {
65                  available[colors[i]] = false;
66              }
67          }
68
69          int cr;
70          for (cr = 0; cr < MAX_VERTICES; cr++) {
71              if (available[cr]) {
72                  break;
73              }
74          }
75
76          colors[u] = cr;
77
78          for (int i = 0; i < numVertices; i++) {
79              if (graph[u][i] && colors[i] != -1) {
80                  available[colors[i]] = true;
81              }
82          }
83      }
84  }
85
```