Examen Programación Para Competición Intermedio: Pregunta 2
Estudiante: Wilson Joel Valeriano Quispe

DIJSKTRA

```cpp
//Algoritmo de dijkstra para el grafo
//Compilador en linea: https://www.onlinegdb.com/online_c++_compiler
#include <iostream>
#include <vector>
#include <queue>
#include <limits>

using namespace std;

const int INF = numeric_limits<int>::max();

void dijkstra(int inicio, vector<vector<pair<int, int>>>& grafo,
vector<int>& distancias) {
    int n = grafo.size();
    priority_queue<pair<int, int>, vector<pair<int, int>>,
greater<pair<int, int>>> pq;

    distancias[inicio] = 0;
    pq.push({0, inicio}); // {distancia, nodo}

    while (!pq.empty()) {
        int distancia_actual = pq.top().first;
        int nodo_actual = pq.top().second;
        pq.pop();

        if (distancia_actual > distancias[nodo_actual])
            continue;

        for (auto& vecino : grafo[nodo_actual]) {
            int nodo_vecino = vecino.first;
            int peso_arista = vecino.second;

            if (distancias[nodo_actual] + peso_arista <
distancias[nodo_vecino]) {
                distancias[nodo_vecino] = distancias[nodo_actual] +
peso_arista;
                pq.push({distancias[nodo_vecino], nodo_vecino});
            }
        }
    }
}
```

Examen Programación Para Competición Intermedio: Pregunta 2
Estudiante: Wilson Joel Valeriano Quispe

```
int main() {
    int nodos = 7;
    vector<vector<pair<int, int>>> grafo(nodos + 1);

    // Definir el grafo dirigido con las distancias
    grafo[1].push_back({2, 10});
    grafo[1].push_back({5, 6});
    grafo[2].push_back({4, 2});
    grafo[2].push_back({3, 5});
    grafo[4].push_back({1, 5});
    grafo[3].push_back({4, 4});
    grafo[5].push_back({6, 9});
    grafo[5].push_back({7, 1});
    grafo[6].push_back({7, 8});
    grafo[6].push_back({4, 3});
    grafo[6].push_back({3, 2});

    vector<int> distancias(nodos + 1, INF);

    dijkstra(1, grafo, distancias);

    for (int i = 1; i <= nodos; ++i) {
        cout << "Distancia minima desde el Nodo 1 hacia el Nodo " << i
<< ": " << (distancias[i] == INF ? -1 : distancias[i]) << endl;
    }

    return 0;
}
```

Examen Programación Para Competición Intermedio: Pregunta 2
Estudiante: Wilson Joel Valeriano Quispe

```cpp
//Algoritmo de dijkstra para el grafo
//Compilador en linea: https://www.onlinegdb.com/online_c++_co
#include <iostream>
#include <vector>
#include <queue>
#include <limits>

using namespace std;

const int INF = numeric_limits<int>::max();
```

input

```
Distancia minima desde el Nodo 1 hacia el Nodo 1: 0
Distancia minima desde el Nodo 1 hacia el Nodo 2: 10
Distancia minima desde el Nodo 1 hacia el Nodo 3: 15
Distancia minima desde el Nodo 1 hacia el Nodo 4: 12
Distancia minima desde el Nodo 1 hacia el Nodo 5: 6
Distancia minima desde el Nodo 1 hacia el Nodo 6: 15
Distancia minima desde el Nodo 1 hacia el Nodo 7: 7


...Program finished with exit code 0
Press ENTER to exit console.
```

KRUSKAL

```cpp
//Algoritmo de Kruskal para el grafo
//Compilador en linea: https://www.onlinegdb.com/online_c++_compiler
#include <iostream>
#include <vector>
#include <algorithm>

using namespace std;

struct Arista {
    int origen, destino, peso;
};

bool compararAristas(Arista a, Arista b) {
    return a.peso < b.peso;
}

int encontrar(int nodo, vector<int>& padre) {
    if (padre[nodo] != nodo)
        padre[nodo] = encontrar(padre[nodo], padre); // Compresion de
camino
    return padre[nodo];
```

Examen Programación Para Competición Intermedio: Pregunta 2
Estudiante: Wilson Joel Valeriano Quispe

```
}

void unir(int nodo1, int nodo2, vector<int>& padre, vector<int>& rango)
{
    int raiz1 = encontrar(nodo1, padre);
    int raiz2 = encontrar(nodo2, padre);

    if (raiz1 != raiz2) {
        if (rango[raiz1] < rango[raiz2]) {
            padre[raiz1] = raiz2;
        } else if (rango[raiz1] > rango[raiz2]) {
            padre[raiz2] = raiz1;
        } else {
            padre[raiz2] = raiz1;
            rango[raiz1]++;
        }
    }
}

int main() {
    int nodos = 7;
    vector<Arista> aristas = {
        {1, 2, 10}, {1, 5, 6}, {2, 4, 2}, {2, 3, 5},
        {4, 1, 5}, {3, 4, 4}, {5, 6, 9}, {5, 7, 1},
        {6, 7, 8}, {6, 4, 3}, {6, 3, 2}
    };

    // Ordenar las aristas por peso
    sort(aristas.begin(), aristas.end(), compararAristas);

    vector<int> padre(nodos + 1), rango(nodos + 1, 0);
    for (int i = 1; i <= nodos; ++i)
        padre[i] = i;

    vector<Arista> mst;
    int peso_total = 0;

    for (Arista& arista : aristas) {
        if (encontrar(arista.origen, padre) !=
encontrar(arista.destino, padre)) {
            unir(arista.origen, arista.destino, padre, rango);
            mst.push_back(arista);
            peso_total += arista.peso;
```

Examen Programación Para Competición Intermedio: Pregunta 2
Estudiante: Wilson Joel Valeriano Quispe

```
        }
    }

    cout << "Arbol de expansion minima (Kruskal):\n";
    for (Arista& arista : mst) {
        cout << arista.origen << " - " << arista.destino << " = " <<
arista.peso << endl;
    }
    cout << "Peso total del MST: " << peso_total << endl;

    return 0;
}
```

```
1  //Algoritmo de Kruskal para el grafo
2  //Compilador en linea: https://www.onlinegdb.com/online_c++_co
3  #include <iostream>
4  #include <vector>
5  #include <algorithm>
6
7  using namespace std;
8
9  struct Arista {
10     int origen, destino, peso;
11 };
```

input

```
Arbol de expansion minima (Kruskal):
5 - 7 = 1
2 - 4 = 2
6 - 3 = 2
6 - 4 = 3
4 - 1 = 5
1 - 5 = 6
Peso total del MST: 19


...Program finished with exit code 0
Press ENTER to exit console.
```