TALLER PROGCOMP: TRACK GRAFOS

MINIMUM SPANNING TREE

Gabriel Carmona Tabja

Universidad Técnica Federico Santa María,
Università di Pisa

October 6, 2024

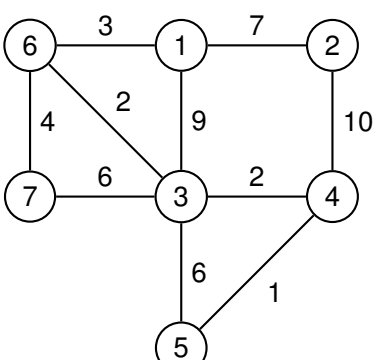
Part I

Minimum Spanning Tree

Minimum Spanning Tree

Definición

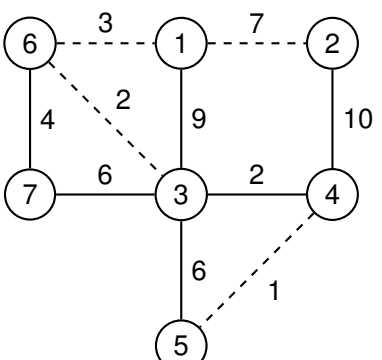
Un subárbol de un grafo que conecta todos los nodos con el menor costo posible, sin crear ciclos.



Minimum Spanning Tree

Definición

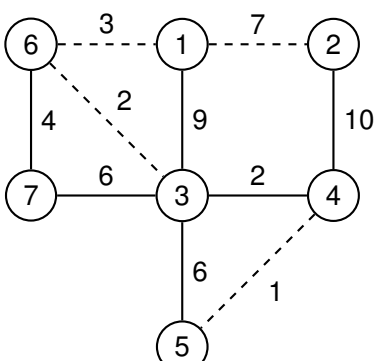
Un subárbol de un grafo que conecta todos los nodos con el menor costo posible, sin crear ciclos.



Minimum Spanning Tree

Definición

Un subárbol de un grafo que conecta todos los nodos con el menor costo posible, sin crear ciclos.

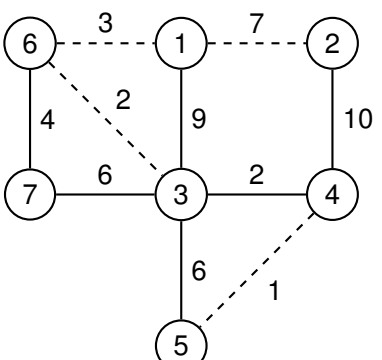


Este subárbol tiene costo 37.

Minimum Spanning Tree

Definición

Un subárbol de un grafo que conecta todos los nodos con el menor costo posible, sin crear ciclos.

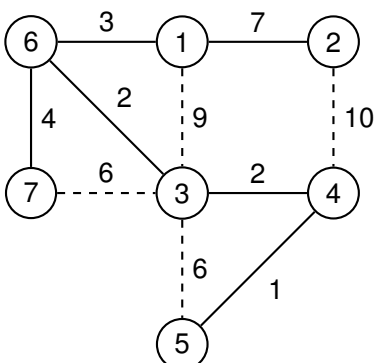


Este subárbol tiene costo 37.
¿Será el mínimo?

Minimum Spanning Tree

Definición

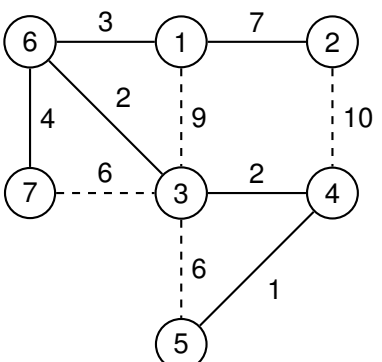
Un subárbol de un grafo que conecta todos los nodos con el menor costo posible, sin crear ciclos.



Minimum Spanning Tree

Definición

Un subárbol de un grafo que conecta todos los nodos con el menor costo posible, sin crear ciclos.

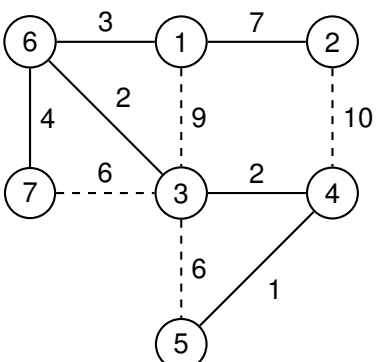


El costo de este subárbol es 19.

Minimum Spanning Tree

Definición

Un subárbol de un grafo que conecta todos los nodos con el menor costo posible, sin crear ciclos.

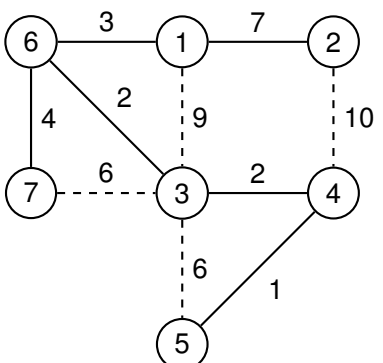


El costo de este subárbol es 19.
Este es el *Minimum Spanning Tree*.

Minimum Spanning Tree

Definición

Un subárbol de un grafo que conecta todos los nodos con el menor costo posible, sin crear ciclos.



El costo de este subárbol es 19.
Este es el *Minimum Spanning Tree*.
¿Cómo resolver esto?

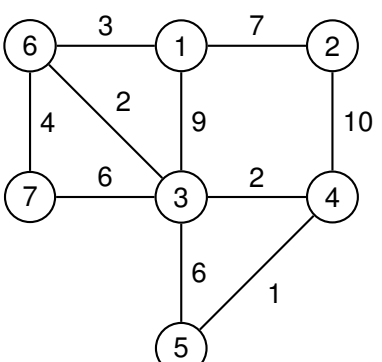
Part II

ALGORITMO DE KRUSKAL

ALGORITMO DE KRUSKAL

Definición

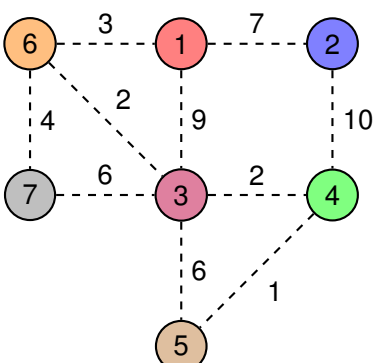
- Algoritmo descrito por Joseph Bernard Kruskal, Jr en 1956



ALGORITMO DE KRUSKAL

Definición

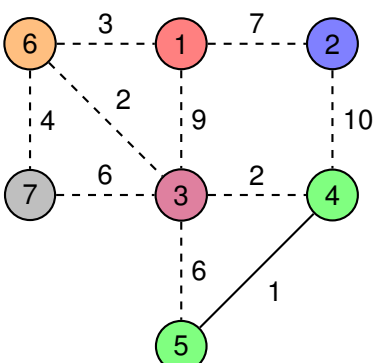
- Algoritmo descrito por Joseph Bernard Kruskal, Jr en 1956



ALGORITMO DE KRUSKAL

Definición

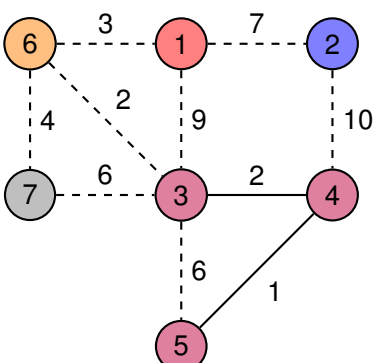
- Algoritmo descrito por Joseph Bernard Kruskal, Jr en 1956



ALGORITMO DE KRUSKAL

Definición

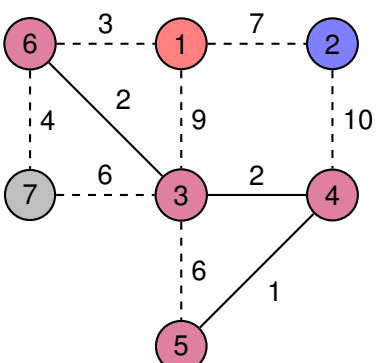
- Algoritmo descrito por Joseph Bernard Kruskal, Jr en 1956



ALGORITMO DE KRUSKAL

Definición

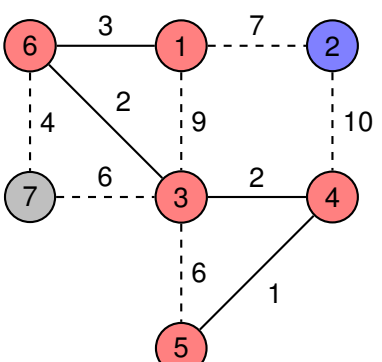
- Algoritmo descrito por Joseph Bernard Kruskal, Jr en 1956



ALGORITMO DE KRUSKAL

Definición

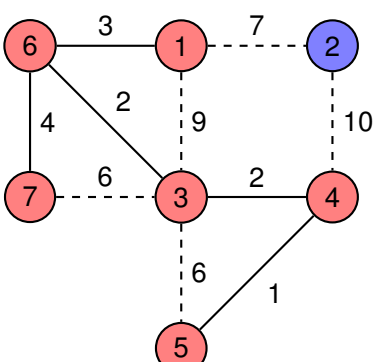
- Algoritmo descrito por Joseph Bernard Kruskal, Jr en 1956



ALGORITMO DE KRUSKAL

Definición

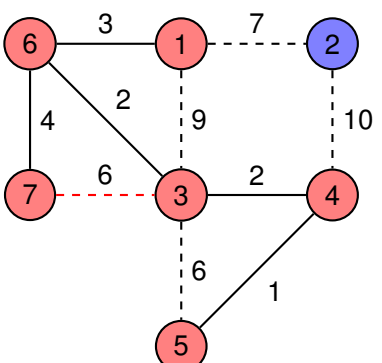
- Algoritmo descrito por Joseph Bernard Kruskal, Jr en 1956



ALGORITMO DE KRUSKAL

Definición

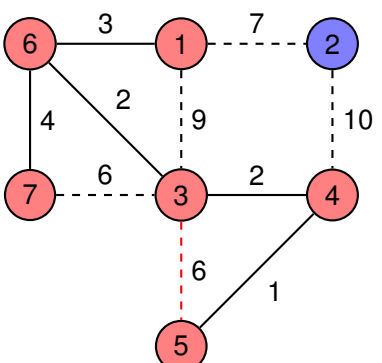
- Algoritmo descrito por Joseph Bernard Kruskal, Jr en 1956



ALGORITMO DE KRUSKAL

Definición

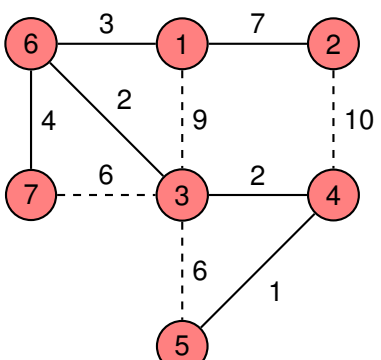
- Algoritmo descrito por Joseph Bernard Kruskal, Jr en 1956



ALGORITMO DE KRUSKAL

Definición

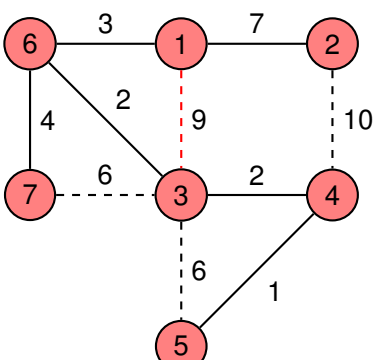
- Algoritmo descrito por Joseph Bernard Kruskal, Jr en 1956



ALGORITMO DE KRUSKAL

Definición

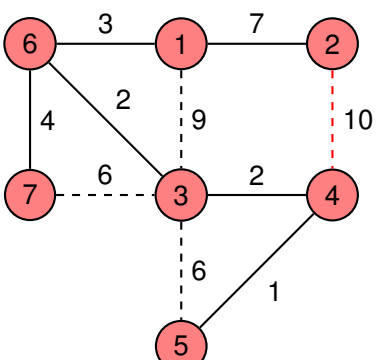
- Algoritmo descrito por Joseph Bernard Kruskal, Jr en 1956



ALGORITMO DE KRUSKAL

Definición

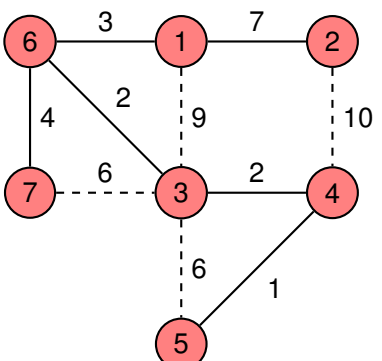
- Algoritmo descrito por Joseph Bernard Kruskal, Jr en 1956



ALGORITMO DE KRUSKAL

Definición

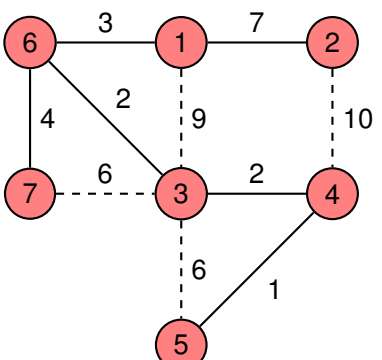
- Algoritmo descrito por Joseph Bernard Kruskal, Jr en 1956



ALGORITMO DE KRUSKAL

Definición

- Algoritmo descrito por Joseph Bernard Kruskal, Jr en 1956

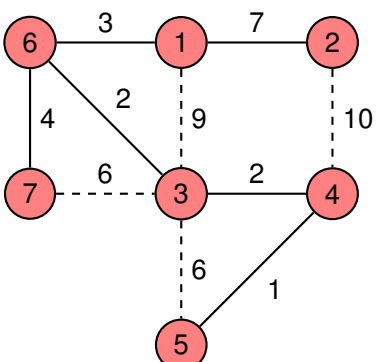


¿Cómo podemos saber si dos nodos hacen parte de un mismo grupo o no de manera eficiente?

Algoritmo de Kruskal

Definición

- Algoritmo descrito por Joseph Bernard Kruskal, Jr en 1956

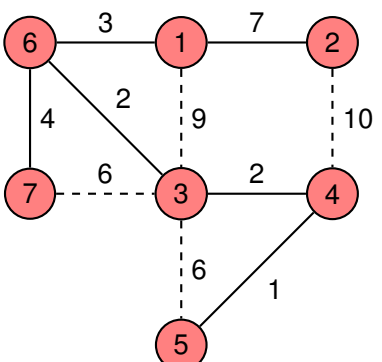


¿Cómo podemos saber si dos nodos hacen parte de un mismo grupo o no de manera eficiente?
¡Disjoint Union Find!

Algoritmo de Kruskal

Definición

- Algoritmo descrito por Joseph Bernard Kruskal, Jr en 1956



¿Cómo podemos saber si dos nodos hacen parte de un mismo grupo o no de manera eficiente?
¡Disjoint Union Find!
Complejidad $O(a \log n)$.

Código

```
int n;
vector< vector< pair< int, int > > > lista;
struct Edge {
  int a; int b; int w;
  Edge(int a_, int b_, int w_) : a(a_), b(b_), w(w_) {}
}
bool c_edge(Edge &a, Edge &b) { return a.w < b.w;  }
int Kruskal() {
  UnionFind sets(n);
  vector< Edge > edges;
  for(int i = 0; i < n; i++) {
    for(pi eg : G[i]) {
      Edge e(i, eg.first, eg.second);
      edges.push_back(e);
    }
  }
  sort(edges.begin(), edges.end(), c_edge);
  int min_cost = 0;
  for(Edge e : Edges) {
    if(sets.find(e.a, e.b) != true) {
      tree.push_back(Edge(e.a, e.b, e.w));
      min_cost += e.w;
      sets.union(e.a, e.b);
    }
  }
  return min_cost;
}
```

Una implementación de Union Find la pueden encontrar en el Handbook de la UTFSM.

REFERENCES I