INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
SUL-RIO-GRANDENSE
Campus Passo Fundo

Ciência da
Computação

ESTRUTURA DE DADOS II

Prof. Adilso Nunes de Souza

KRUSKAL

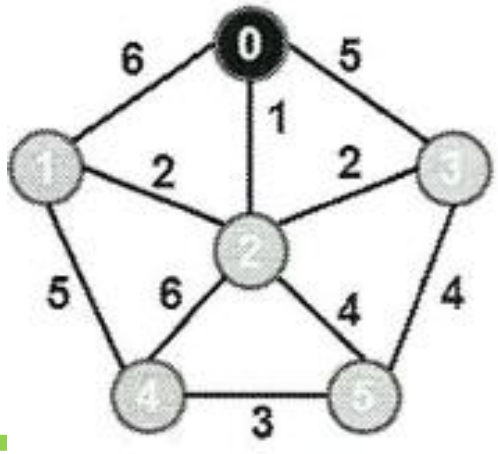
- O algoritmo de KRUSKAL é outro algoritmo clássico capaz de obter uma solução ótima para o problema da árvore geradora mínima.
- Considerando cada vértice uma árvore independente, o algoritmo procura a aresta de menor peso que conecta duas árvores diferentes. Os vértices das árvores selecionadas passam a fazer parte de uma mesma árvore.
- O processo se repete até que todos os vértices façam parte de uma mesma árvore.

KRUSKAL

- O algoritmo de KRUSKAL se inicia com uma floresta, várias árvores. A cada iteração, duas árvores são selecionadas para ser unidas em uma mesma árvore.
- Esse algoritmo pode ser implementado com uma função que recebe três parâmetros: o grafo, o vértice que será ponto de partida e um array para marcar que é o pai de cada vértice. Também será necessário um array auxiliar para gerenciar a qual árvore cada vértice pertence.

KRUSKAL EXEMPLO

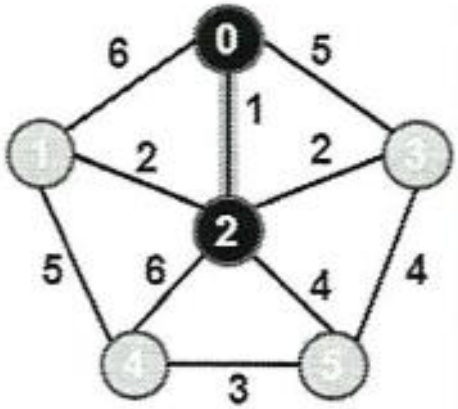
- Inicia o cálculo com o vértice 0
- Atribui seu próprio índice como pai
- O restante dos vértices recebem pai igual -1 (sem pai).
- Inicializa o chefe com o índice do vértice.



pai	arv
0	0
-1	1
-1	2
-1	3
-1	4
-1	5

KRUSKAL EXEMPLO

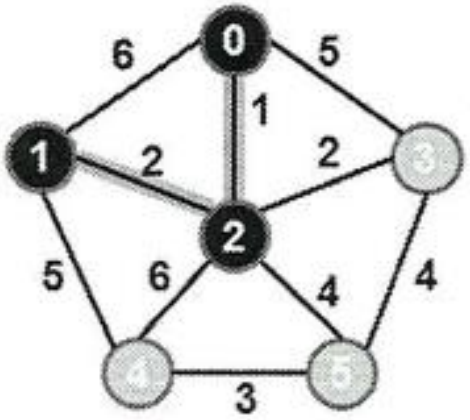
- Procura a aresta com menor peso conectando vértices com chefes diferentes: vértice 0 e 2
- Atribui vértice 0 como pai do vértice 2
- Todos que possuem chefe igual ao vértice 2 passam a ter chefe igual ao vértice 0.



pai	arv
0	0
-1	1
0	0
-1	3
-1	4
-1	5

KRUSKAL EXEMPLO

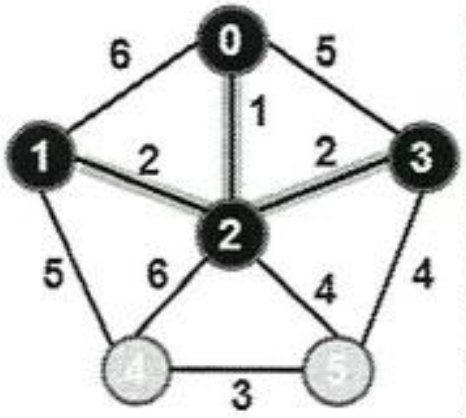
- Procura a aresta com menor peso conectando vértices com chefes diferentes: vértice 1 e 2
- Atribui vértice 2 como pai do vértice 1
- Todos que possuem chefe igual ao vértice 2 passam a ter chefe igual ao vértice 1.



pai	arv
0	1
2	1
0	1
-1	3
-1	4
-1	5

KRUSKAL EXEMPLO

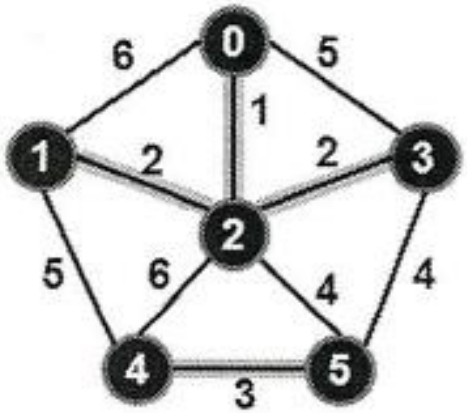
- Procura a aresta com menor peso conectando vértices com chefes diferentes: vértice 2 e 3
- Atribui vértice 2 como pai do vértice 3
- Todos que possuem chefe igual ao vértice 3 passam a ter chefe igual ao vértice 2.



pai	arv
0	1
2	1
0	1
2	1
-1	4
-1	5

KRUSKAL EXEMPLO

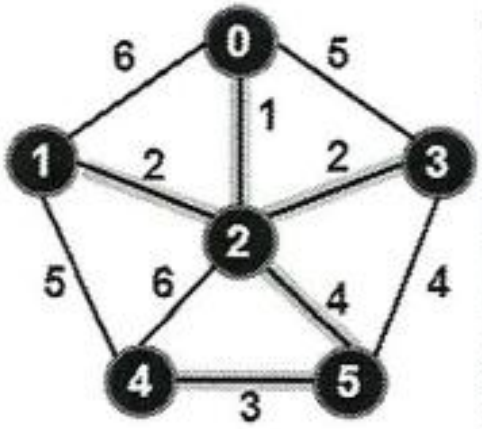
- Procura a aresta com menor peso conectando vértices com chefes diferentes: vértice 4 e 5
- Atribui vértice 5 como pai do vértice 4
- Todos que possuem chefe igual ao vértice 5 passam a ter chefe igual ao vértice 4.



pai	arv
0	1
2	1
0	1
2	1
5	4
-1	4

KRUSKAL EXEMPLO

- Procura a aresta com menor peso conectando vértices com chefes diferentes: vértice 2 e 5
- Atribui vértice 2 como pai do vértice 5
- Todos que possuem chefe igual ao vértice 5 passam a ter chefe igual ao vértice 2.



pai	arv
0	1
2	1
0	1
2	1
5	1
2	1

KRUSKAL

- Uma vez que se tenha percorrido todas as arestas do grafo, verifique se foi possível achar uma aresta de menor peso, caso não tenha sido possível, o processo termina.
- Ao final o array pai irá conter os antecessores de cada vértice do grafo na árvore montada a partir do vértice inicial.
- A eficiência do algoritmo de KRUSKAL depende da forma usada para encontrar a aresta de menor peso.

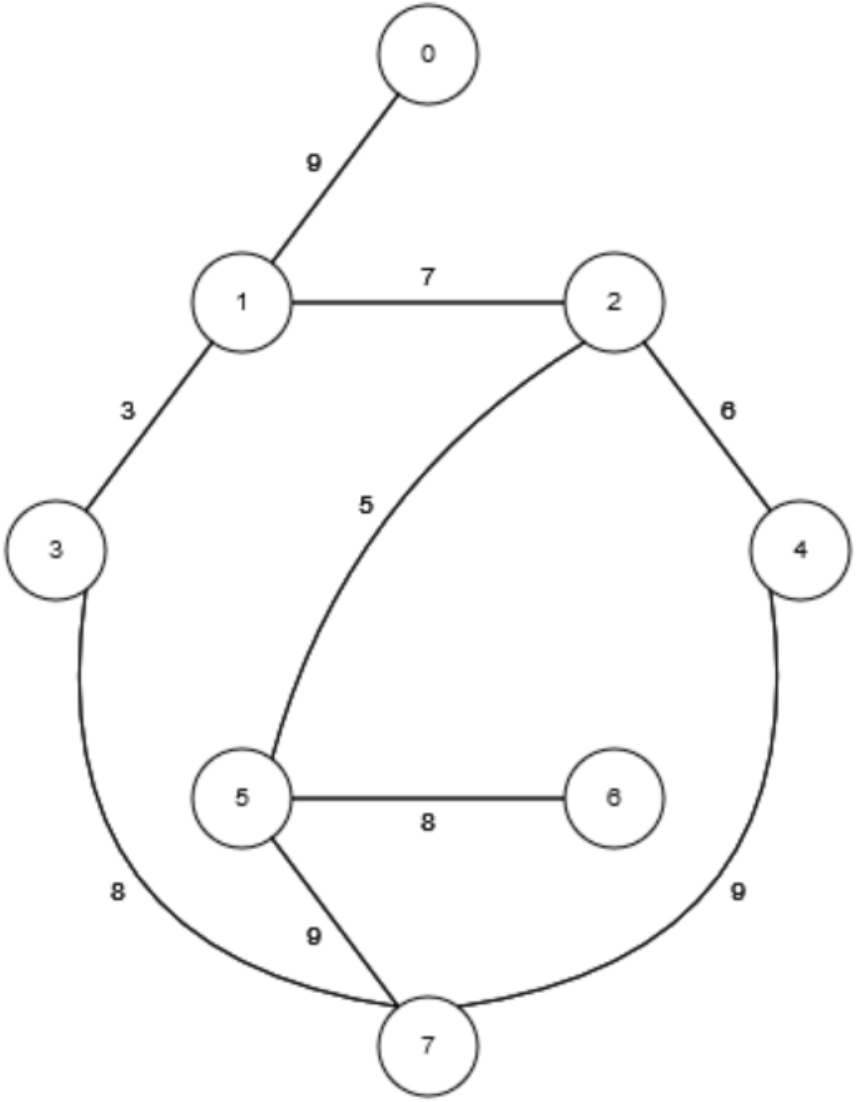
```
01  void algoritmoKruskal_Grafo(Grafo *gr,int orig,int *pai){
02      int i, j, dest, NV, primeiro, *arv;
03      double menorPeso;
04      NV = gr->nro_vertices;
05      arv = (int*) malloc(NV * sizeof(int));
06      for(i=0; i < NV; i++){
07          arv[i] = i;
08          pai[i] = -1;// sem pai
09      }
10      pai[orig] = orig;
11      while(1){
12          primeiro = 1;
13          for(i=0; i < NV; i++){
14              for(j=0; j<gr->grau[i]; j++){
15                  if(arv[i] != arv[gr->arestas[i][j]]){
16                      if(primeiro){
17                          menorPeso = gr->pesos[i][j];
18                          orig = i;
19                          dest = gr->arestas[i][j];
20                          primeiro = 0;
21                      }else{
22                          if(menorPeso > gr->pesos[i][j]){
23                              menorPeso = gr->pesos[i][j];
24                              orig = i;
25                              dest = gr->arestas[i][j];
26                          }
27                      }
28                  }
29              }
30          }
31          if(primeiro == 1)
32              break;
33
34          if(pai[orig] == -1)
35              pai[orig] = dest;
36          else
37              pai[dest] = orig;
38
39          for(i=0; i < NV; i++)
40              if(arv[i] == arv[dest])
41                  arv[i] = arv[orig];
42
43      }
44      free(arv);
45  }
```

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
SUL-RIO-GRANDENSE
Campus Passo Fundo

KRUSKAL

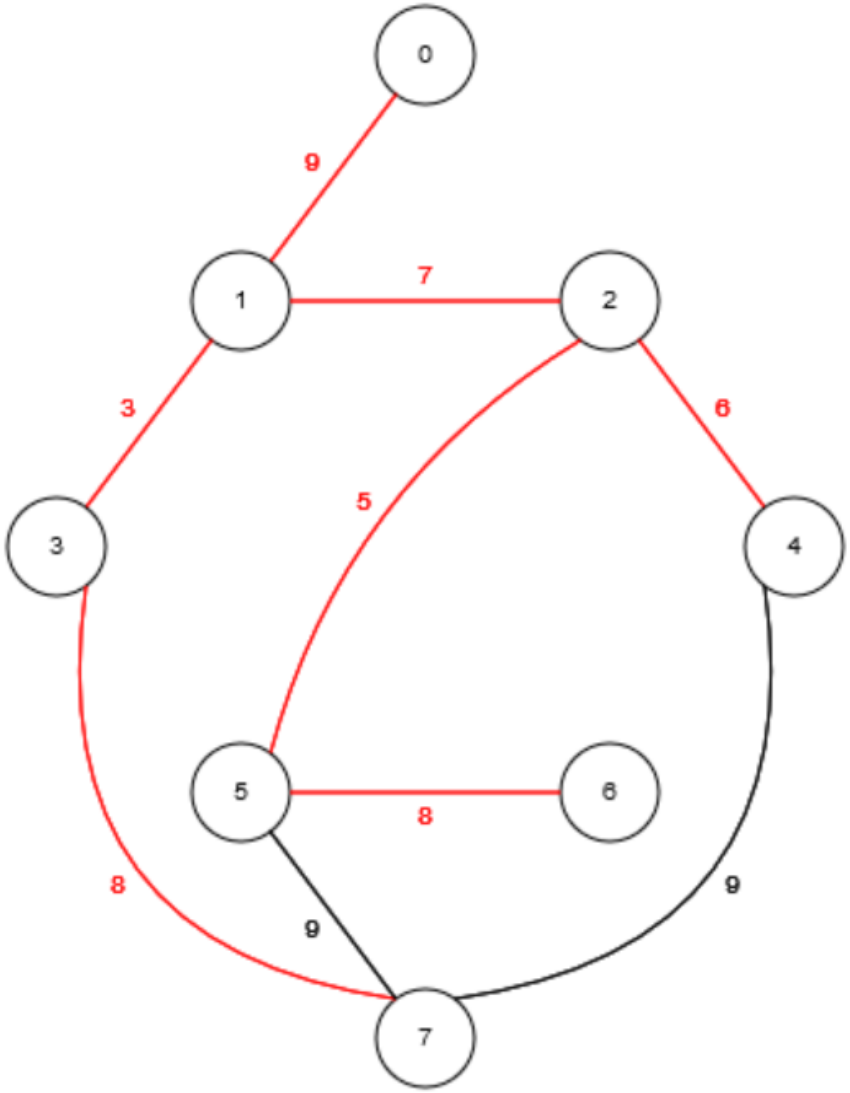
- Dado o grafo aplique o algoritmo de KRUSKAL, tendo como vértice inicial o 1.

Apresente a solução.



KRUSKAL

- Solução final



REFERÊNCIAS

- PEREIRA, Silvio do Lago. Estrutura de Dados Fundamentais: Conceitos e Aplicações, 12. Ed. São Paulo, Érica, 2008.
- BACKES, André Ricardo, Estrutura de dados descomplicada: em linguagem C, 1 Ed. – Rio de Janeiro: Elsevier, 2016.
- ROCHA, Anderson, Grafos – Representações, buscas e Aplicações.
- SENGER, H., Notas de Aula, Universidade de São Judas Tadeu, 1999.
- WALDEMAR Celes, Renato Cerqueira, José Lucas Rangel, Introdução a Estruturas de Dados, Editora Campus (2004).
- VELOSO, Paulo. SANTOS, Celso dos. AZEVEDO, Paulo. FURTADO, Antonio. Estrutura de dados. Rio de Janeiro: Ed. Elsevier, 1983 27ª reimpressão.
- https://www.cs.usfca.edu/~galles/visualization/Algorithms.html
- ZIVIANI, Nivio. Projeto de Algoritmos com implementações em Java e C++, 2007.