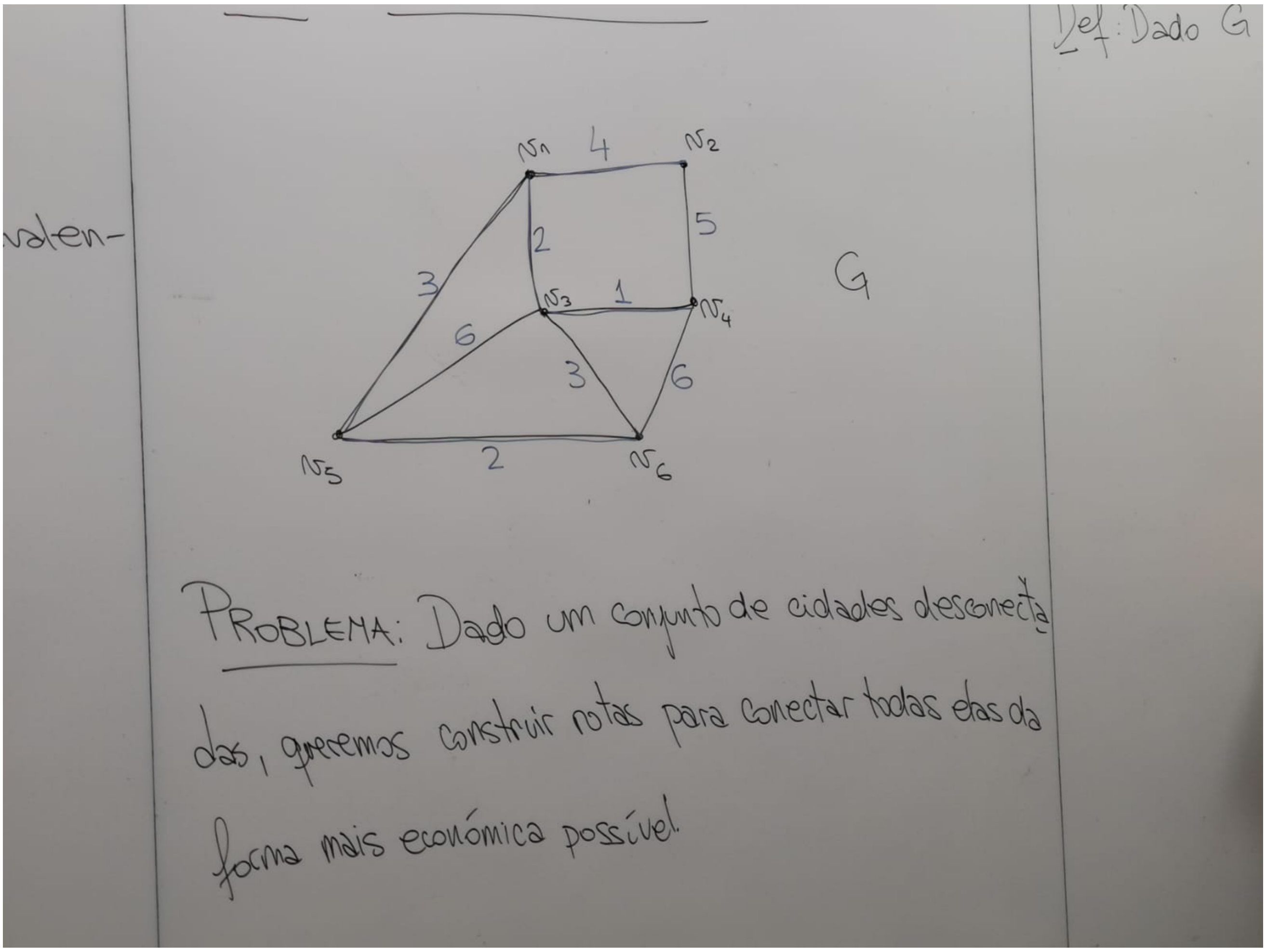
Árvore Geradora Minimal

Descrição

Dado G um grafo com peso, dizemos que T é uma ÁRVORE GERADORA MINIMAL de G se T é uma árvore geradora de peso total* mínimo.

$*$ Soma dos pesos das arestas envolvidas.

Exemplo Motivacional



Algoritmo Prim

Descrição

Encontra a árvore geradora minimal de um grafo.

Input

Grafo conexo com pesos (w) com vértices 1,...,n, e vértice inicial S.
$w(i,j)$: peso de (i,j) se (i,j) é aresta de G.
$w(i,j)$: $+\infty$ se (i,j) não é aresta de G.
(computacionalmente escolher um valor maior a max $w(i,j),(i,j)$: aresta)

Output

$E =$ conjunto de arestas da árvore geradora minimal (mst = minimal spanning tree)

Pseudocódigo

```
prim(w,n,s){
//v(i) = 1 se foi adicionado à mst
//v(i) = 0 se não foi adicionado à mst
//mst: minimal spanning tree

1. for i=1 to n{
2.      v(i) = 0
}

3. v(s) = 1
4. E = NULL

5. for i=1 to n-1{ //para cada i o algoritmo busca uma aresta com um vértice mst (v(j)
= 1) e um vértice k fora da mst (v(k)=0) com peso mínimo
6.      min = + ∞
7.      for j=1 to n
8.              if v(j) = 1 { //j está em mst
9.                      for k=1 to n:
10.                             if (v(k) = 0 and w(j,k) < min){
11.                                     add_vertex = k
12.                                     e = (j,k)
13.                                     min = w(j,k)
14.                             }
                        }
15.             v(add_vertex) = 1
16.             E = E ∪ {e}
17.     }
}
return E
}
```

$(*)$ Para cada $i = 1, \ldots, n-1$, o algoritmo busca a aresta 'e' de menor peso com um vértice na árvore gerada até esse momento e um vértice fora da árvore. Logo adiciona essa aresta 'e' à árvore.

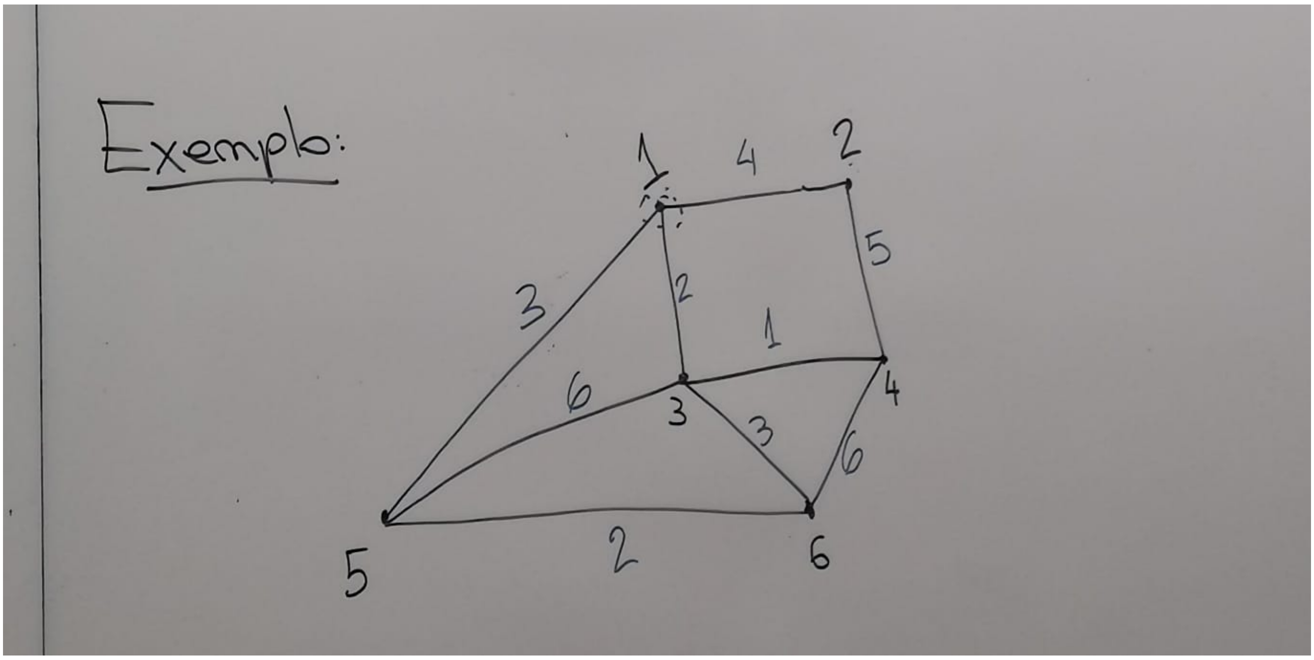
Inicialização

- Defina o vértice inicial igual a 1
- Defina todos os outros vértices iguais a 0
- Inicializa lista de arestas E vazia

Iterações

- O algoritmo busca uma aresta com um vértice nst ($v(j) = 1$) e um vértice k fora da nst ($v(k) = 0$) com peso mínimo
- Define esse vértice igual a 1
- Adiciona a aresta à lista de arestas E

Exemplo



s = 1
v(s) = 1
v(i) = 0, para $i \neq s$
$E = \phi$

5. i = 1 // começo do for

aresta	peso
(1,2)	4
(1,3)	2
(1,5)	3

Escolhemos (1,3) --> 2 //menor peso

15. v(3) = 1

16. $E = \{(1,3)\}$

17. i=2

aresta	peso
(1,2)	4
(1,5)	3
(3,4)	1
(3,5)	6
(3,6)	3

Escolhemos (3,4) --> 1 //menor peso
15. $v(4) = 1$
16. $E = \{(1,3), (3,4)\}$

5. i = 3

aresta	peso
(1,2)	4
(1,5)	3
(3,5)	6
(3,6)	3
(4,2)	5
(4,6)	6

$v(5) = 1$
$E = \{(1,3), (3,4), (1,5)\}$

i = 4
.
.
.

$E = (1,3), (3,4), (1,5), (5,6)$

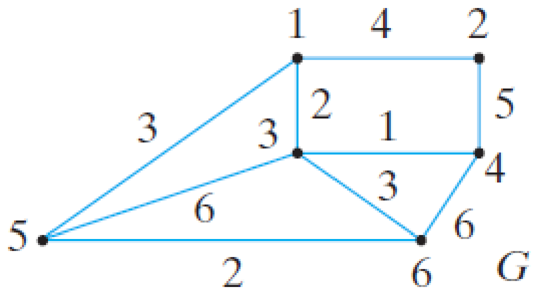
i = 5
.
.
.

$E = \{(1,3), (3,4), (1,5), (5,6), (1,2)\}$ // arestas da árvore geradora minimal (mst)
end

Observação

O algoritmo Prim é GULOSO ("greedy"), dado que escolhe a melhor opção em cada iteração.

Exemplo 9.4.1



Minimal Spanning Tree: