# Ant Colony Optimization

Cassio Conti
cassio@inf.ufsc.br

# Roteiro

1. Problemas de otimização
    - Menor Caminho
    - Ideia do Ant Colony Optimization (ACO)
    - Caixeiro Viajante
2. Ant Colony Optimization (ACO)
    - Ant System
3. Aplicação
    - Exemplo
    - Código
    - Heurística

# Problemas de Otimização

Os problemas de otimização combinatória possuem a qualidade da solução ligada a ordem em que os elementos que compõem a solução aparecem. Quanto mais elementos, mais combinações são possíveis e maior é o tempo de processamento necessário para encontrar a solução ótima.

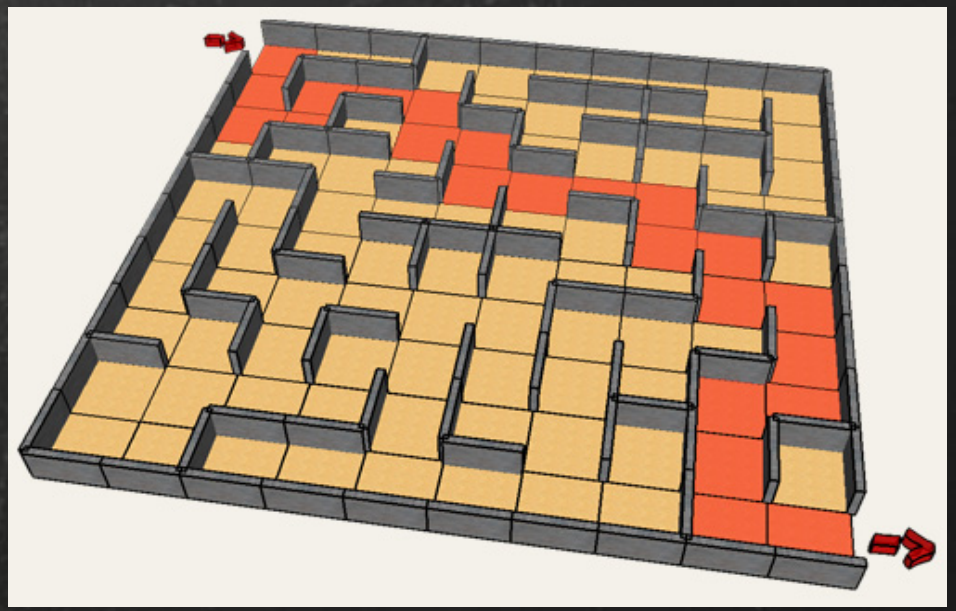
- Sequência desejada:
  - 2-3-1
- Combinações possíveis:
  - 1-2-3
  - 1-3-2
  - 2-1-3
  - 2-3-1
  - 3-1-2
  - 3-2-1

# Problemas de otimização

3 elementos: 6 combinações
4 elementos: 24 combin.
5 elementos: 120 combin.
6 elementos: 720 combin.
7 elementos: 5040 combin.
8 elementos: 40320 combin.
9 elementos: 362880 comb.
n elementos: n! combinações

Algoritmos de otimização utilizam técnicas para buscar a melhor solução de um problema em situações onde seria necessário muito tempo computacional para analisar todos os possíveis valores.

# Menor Caminho



Dado um ambiente com obstáculos e diversos possíveis caminhos de ir de um ponto A até um ponto B, encontrar o melhor (menor) caminho entre A e B.

# Ideia do ACO

Baseia-se no comportamento de formigas reais.

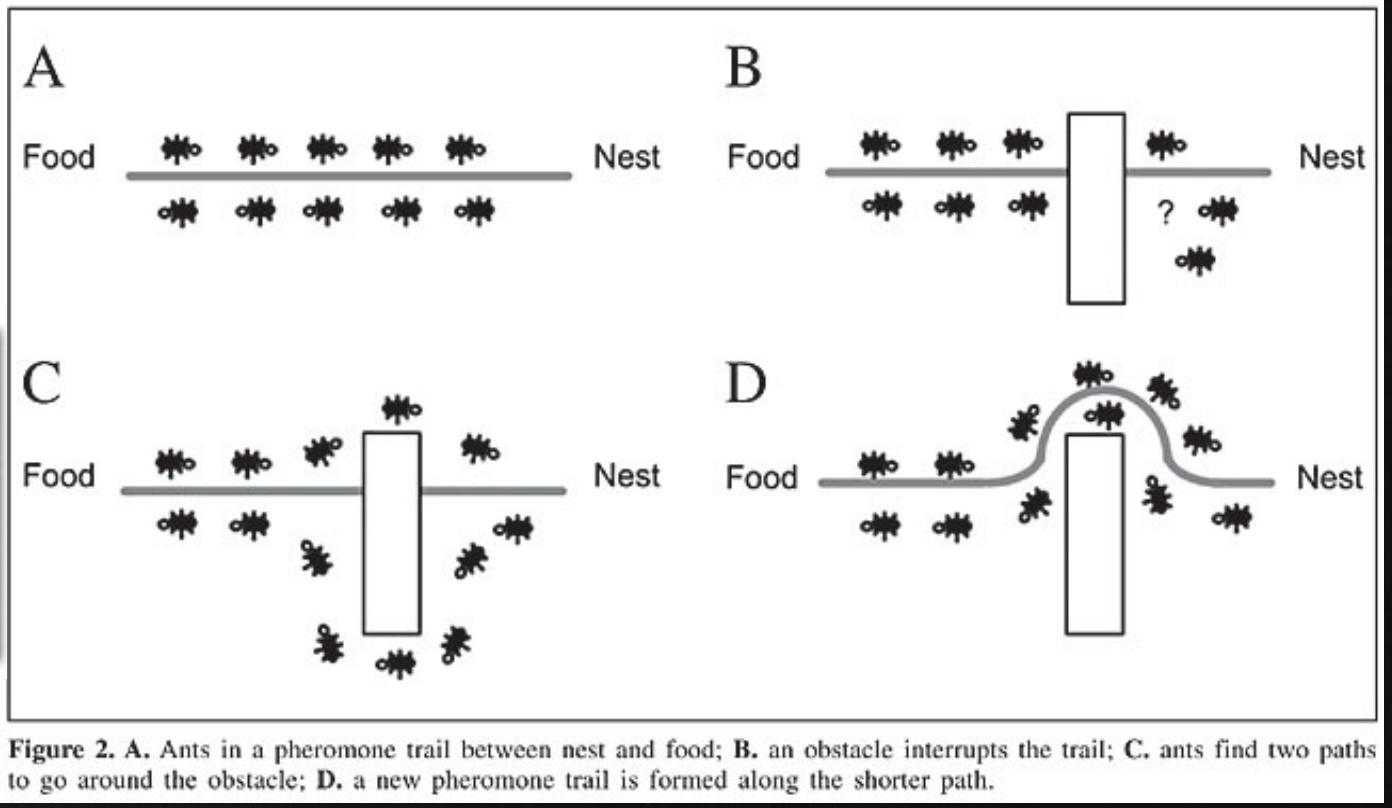
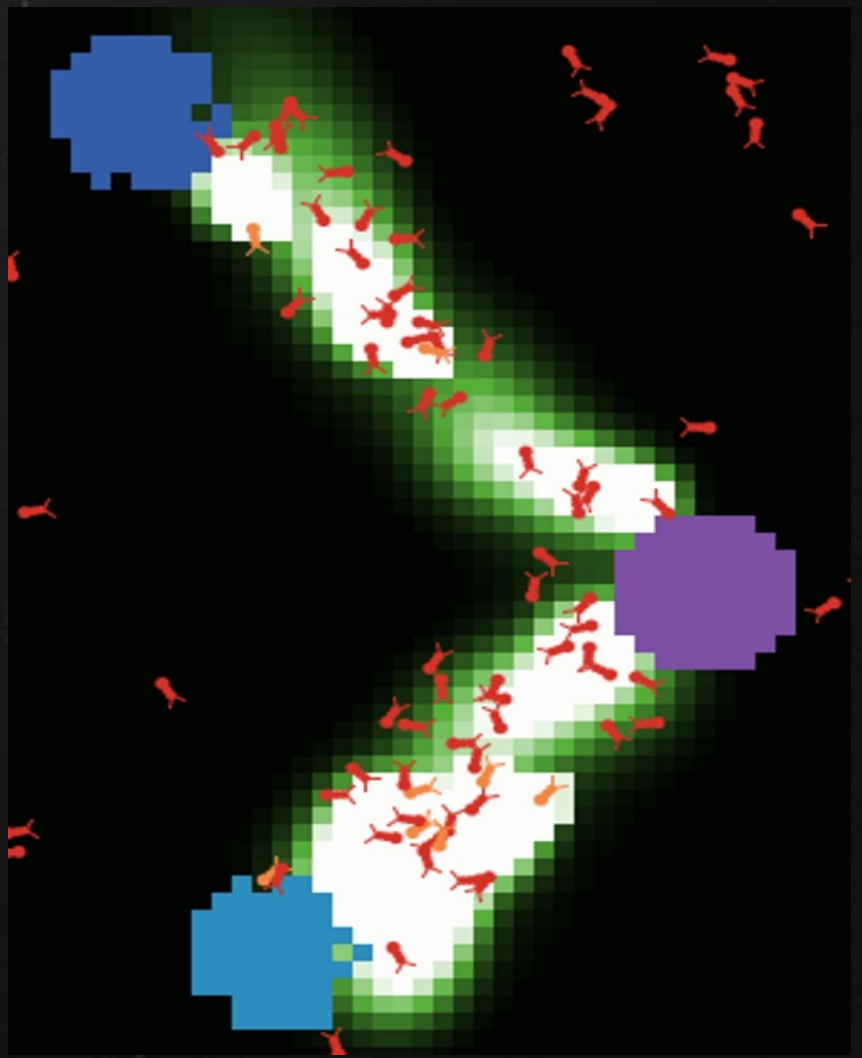


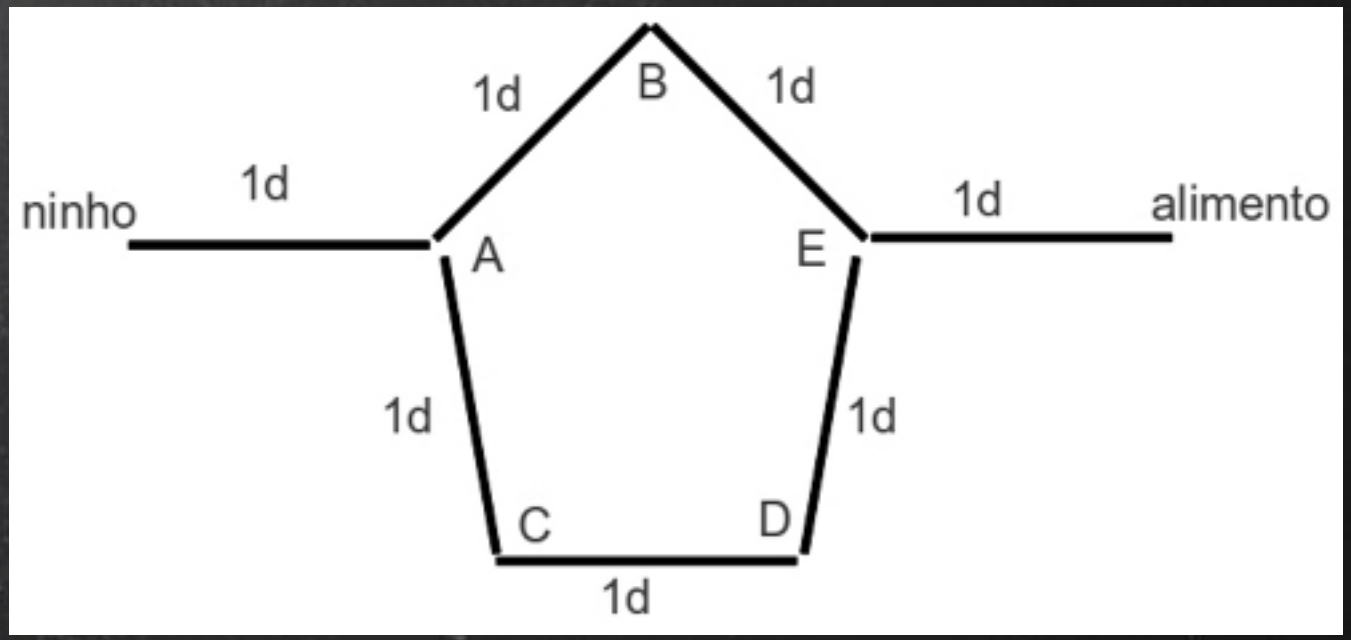

**Figure 2. A.** Ants in a pheromone trail between nest and food; **B.** an obstacle interrupts the trail; **C.** ants find two paths to go around the obstacle; **D.** a new pheromone trail is formed along the shorter path.

# Ideia do ACO

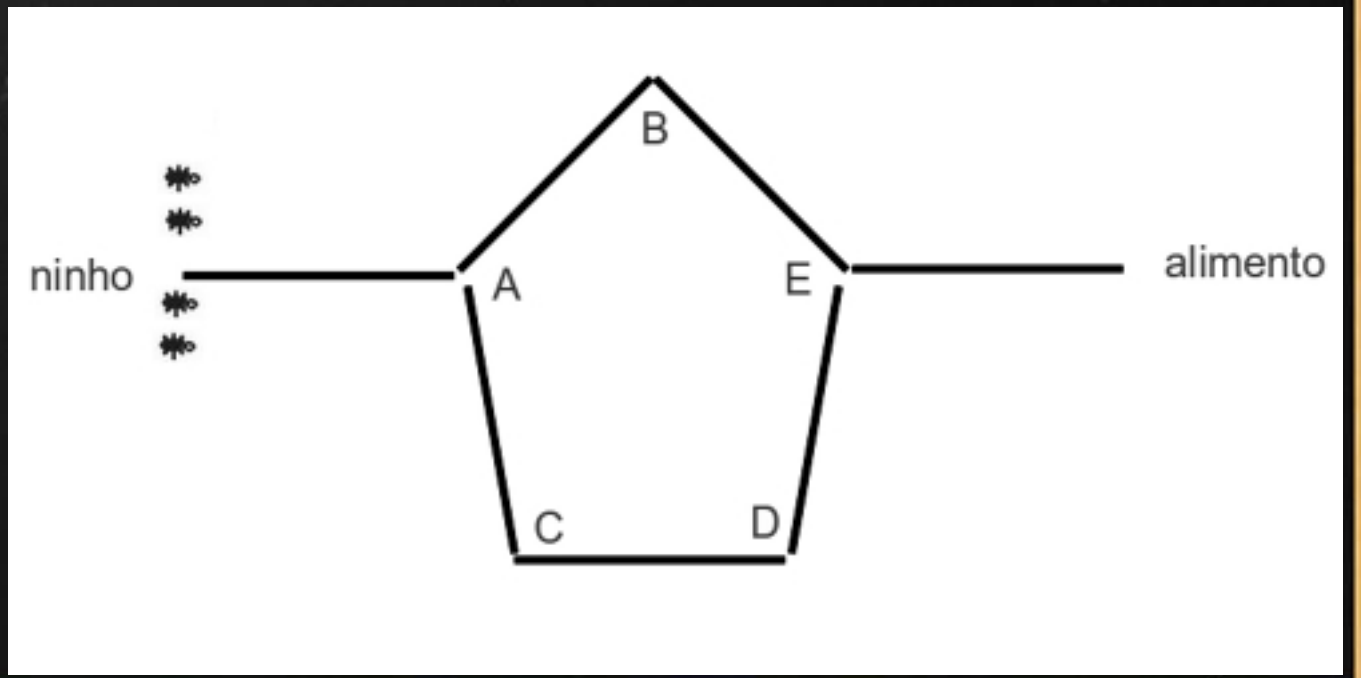
- As formigas, tanto as reais quanto as do ACO baseiam seu "conhecimento" do ambiente nas trilhas de feromônio.
- A tendência é escolher a trilha com mais feromônio.
- Feromônio evapora com o tempo.



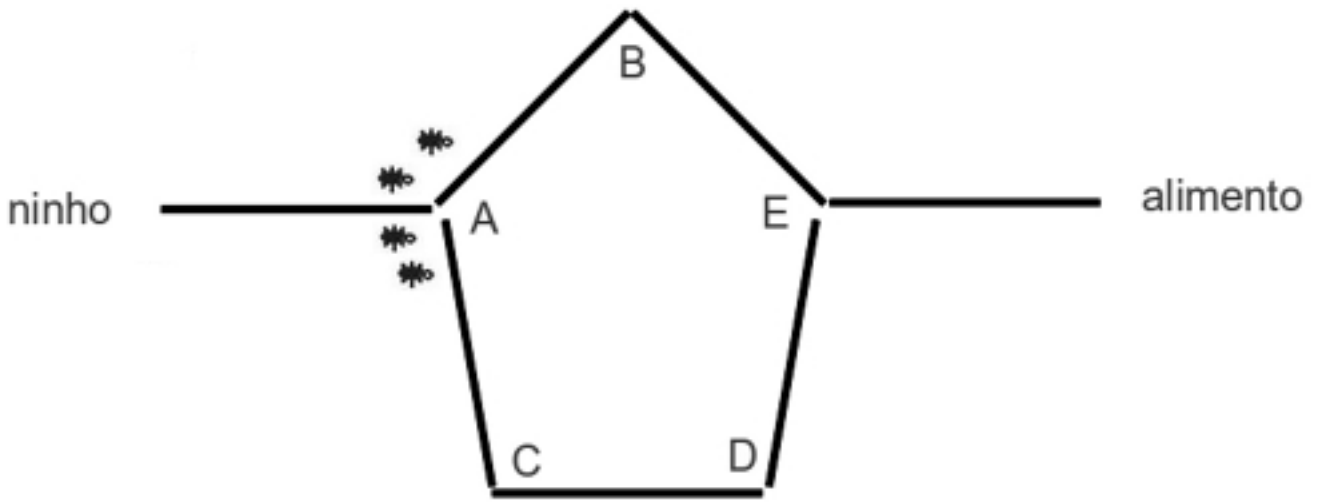
# Ideia do ACO




Passo 1

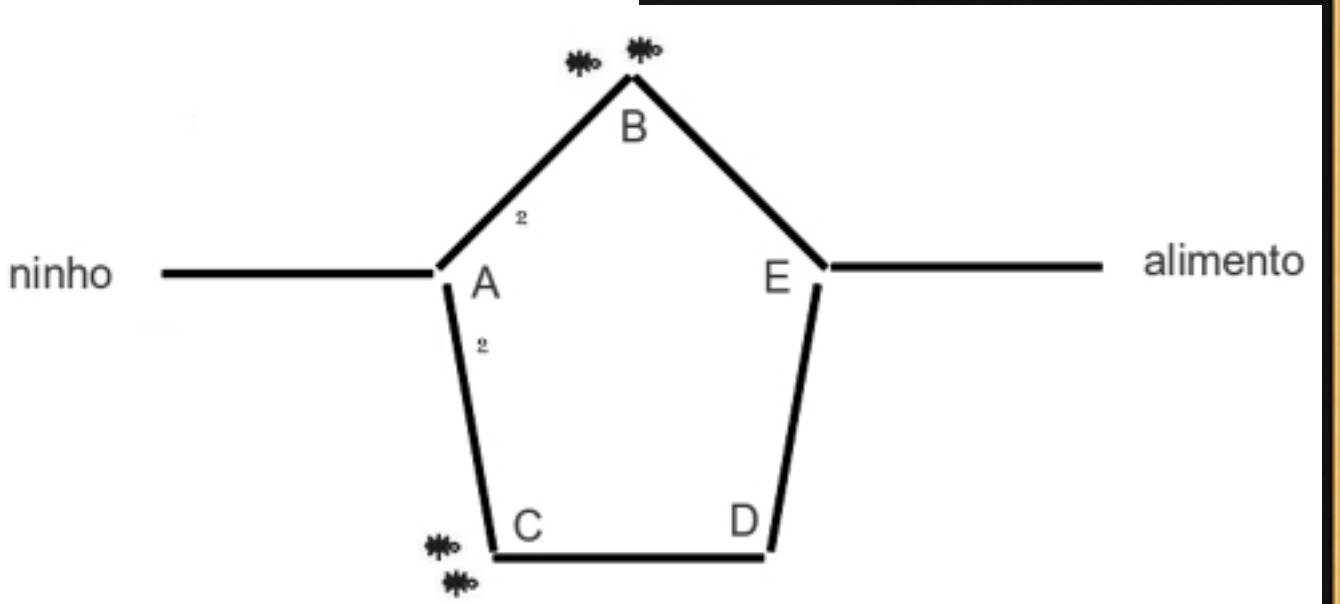


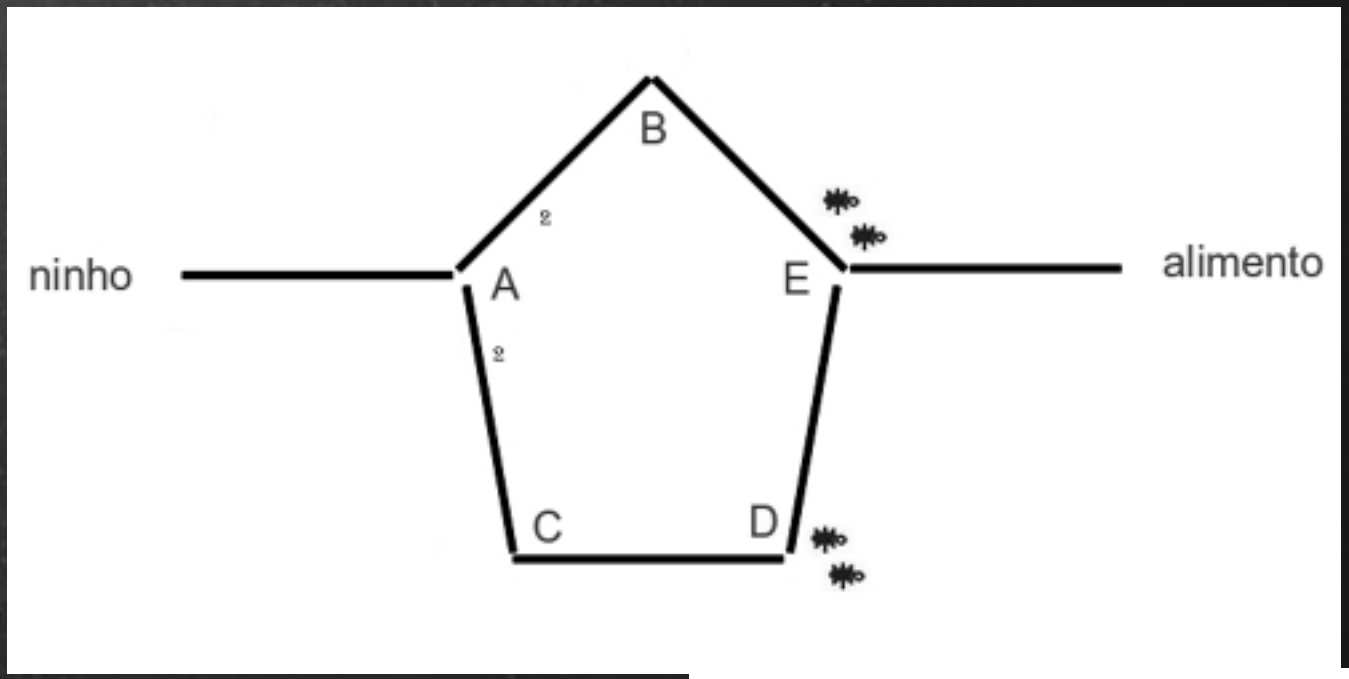
# Ideia do ACO




Passo 2

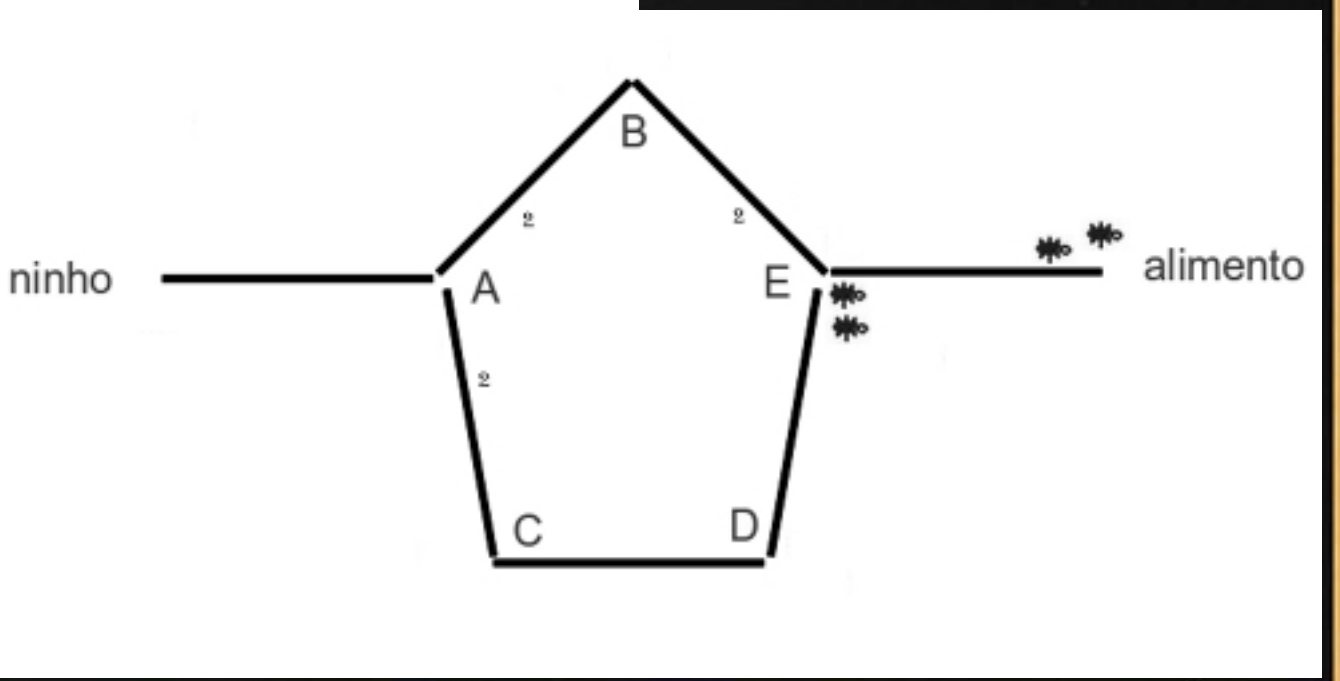
Passo 3

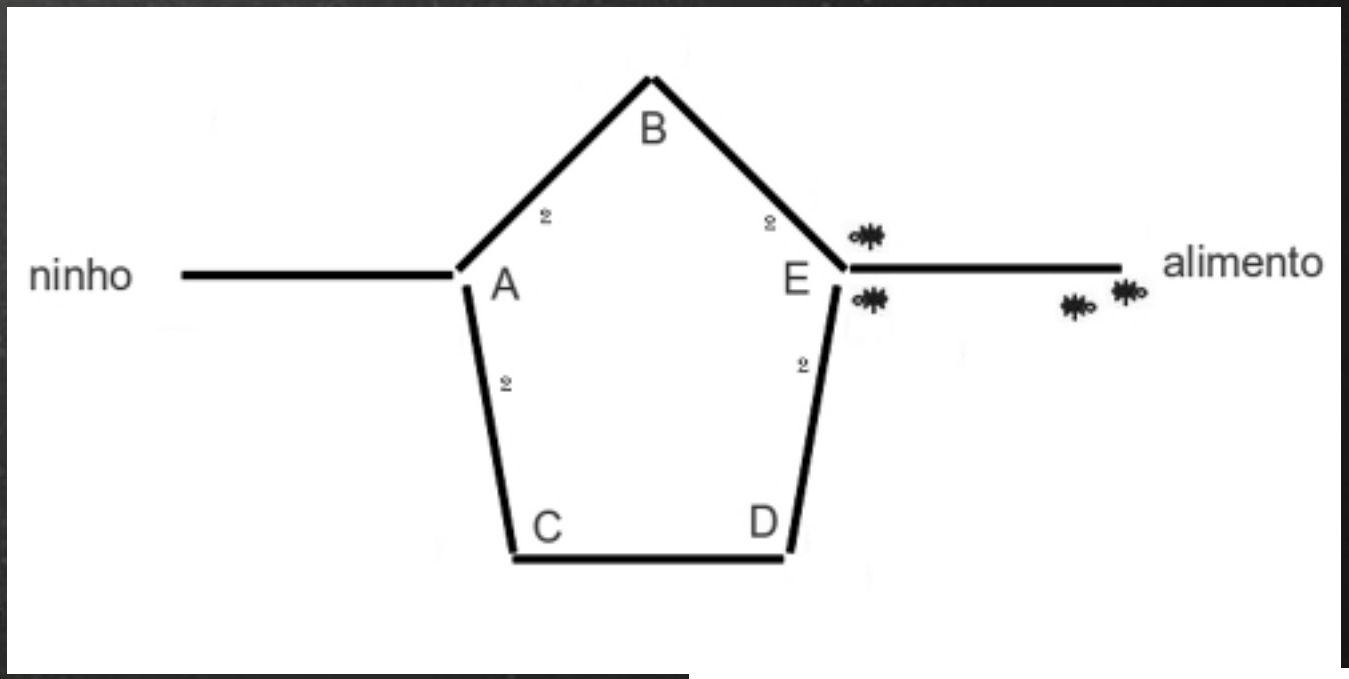





Passo 4

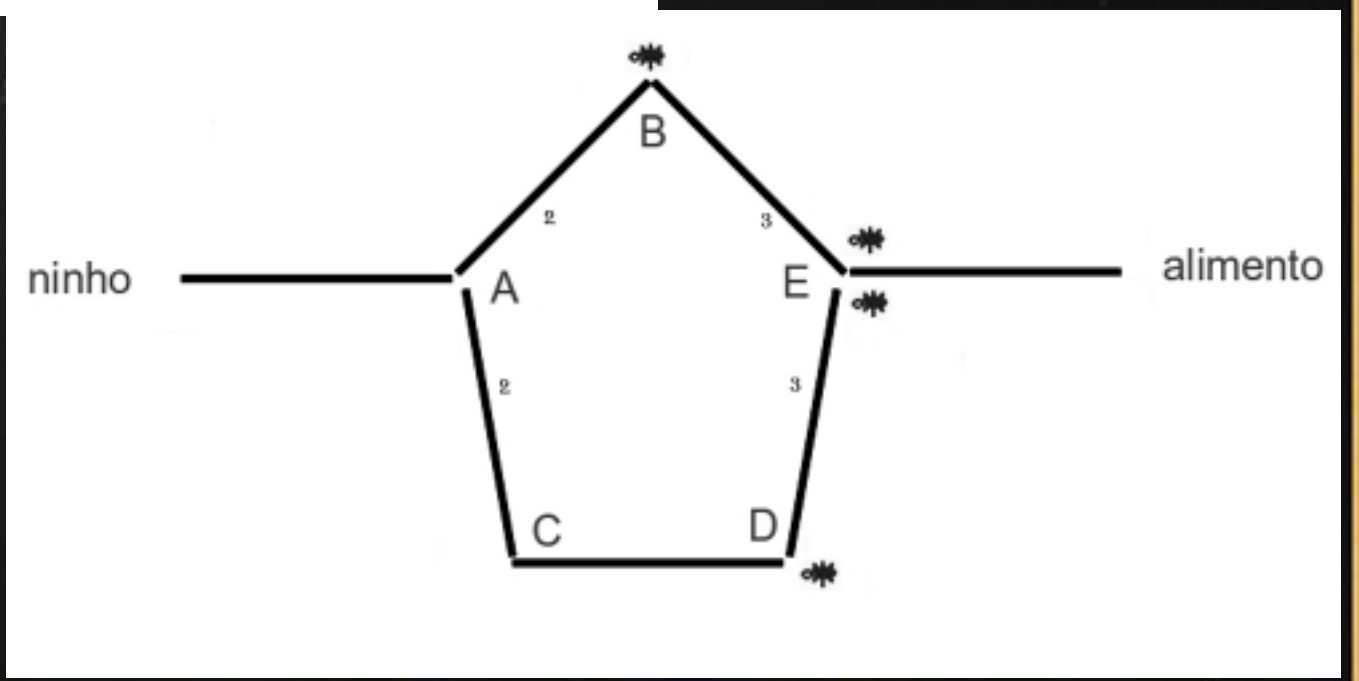
Passo 5

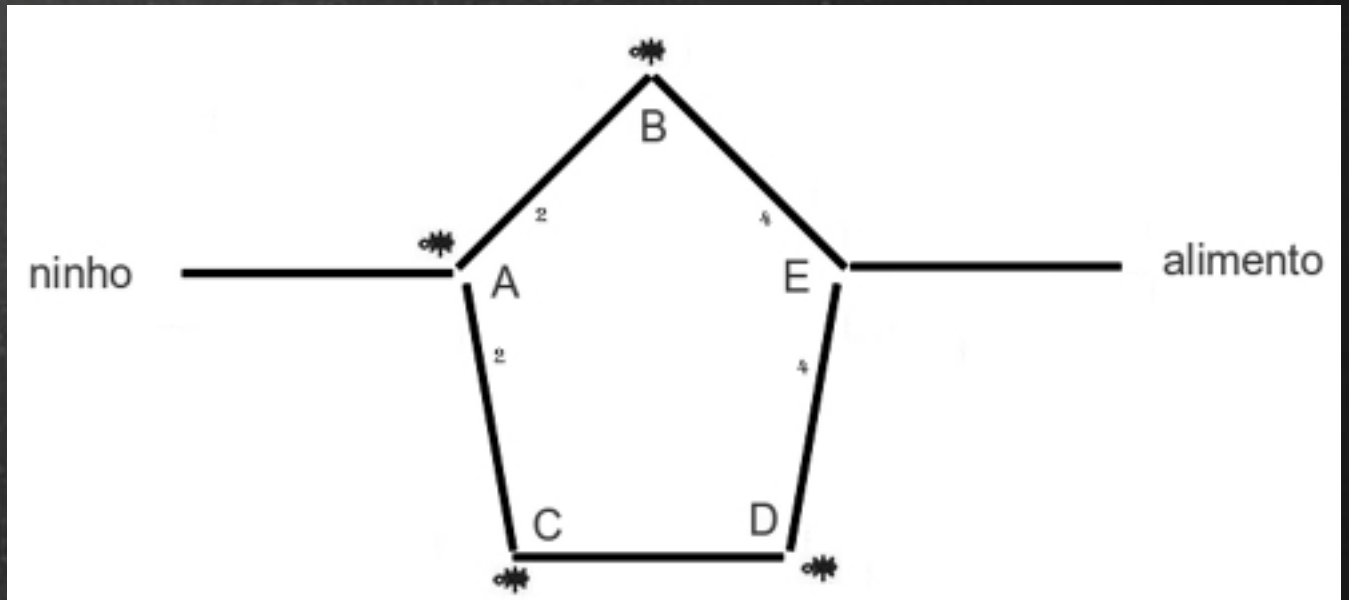





Passo 6

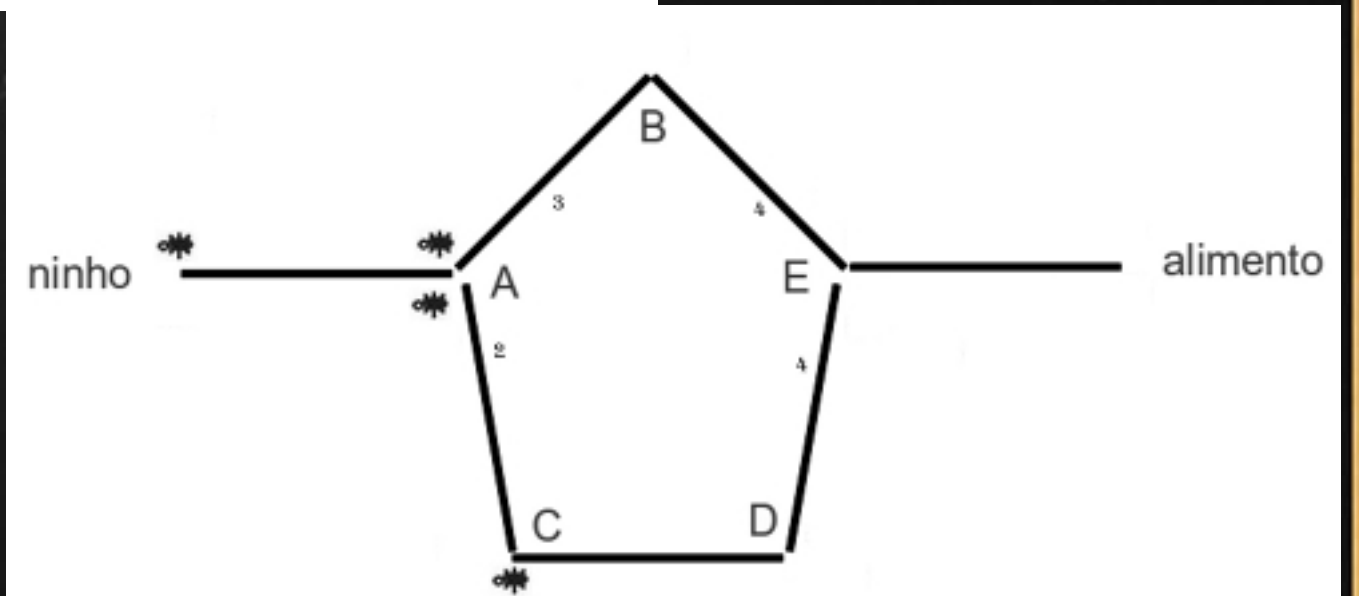
Passo 7

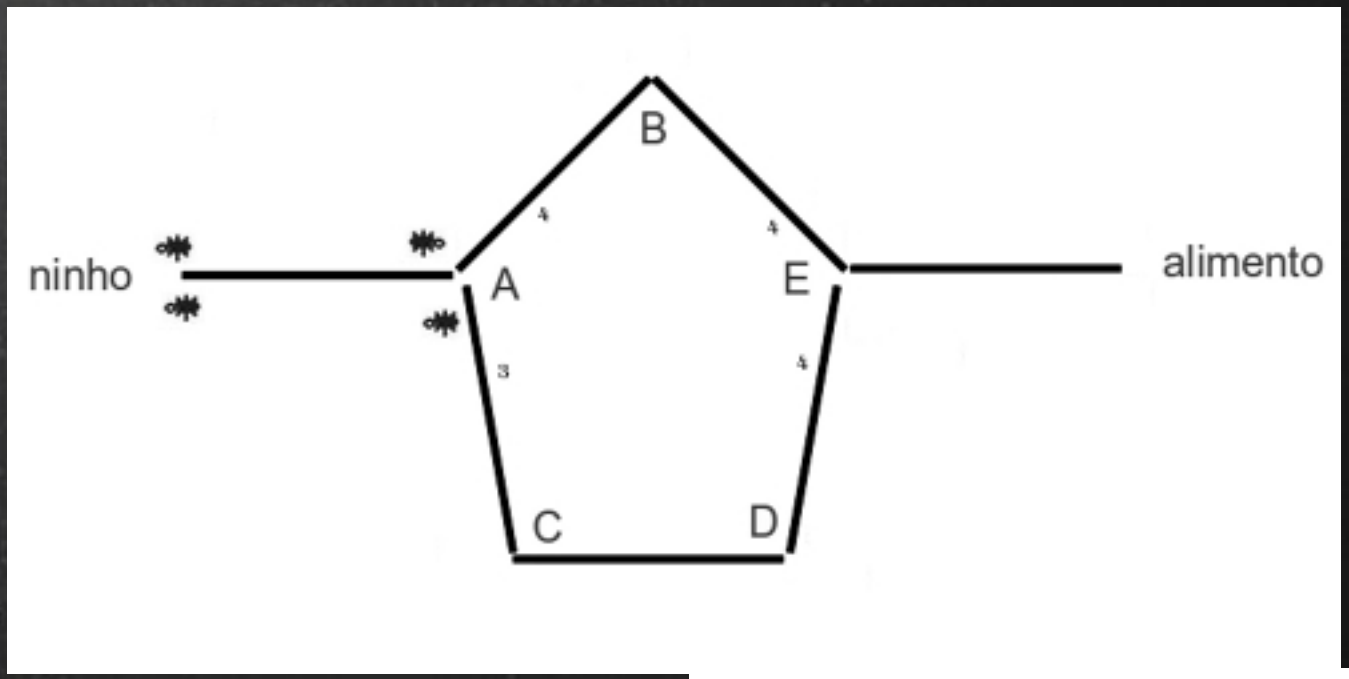





Passo 8

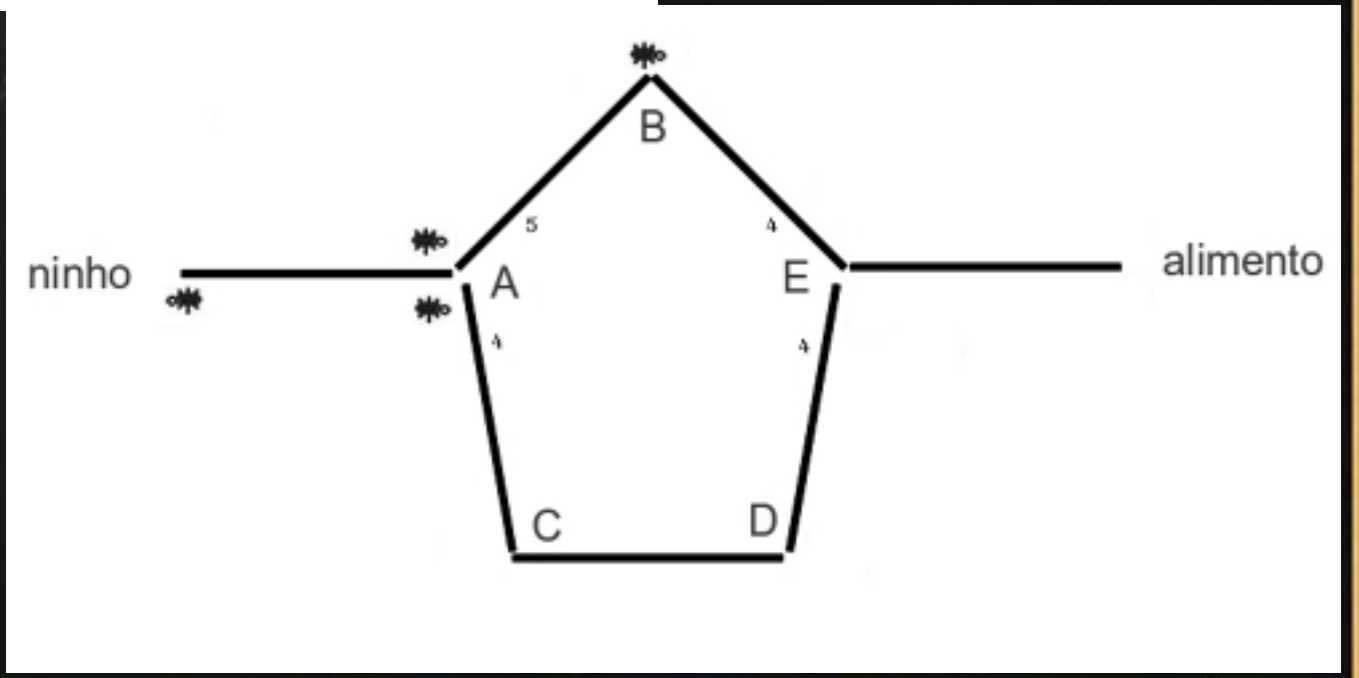
Passo 9

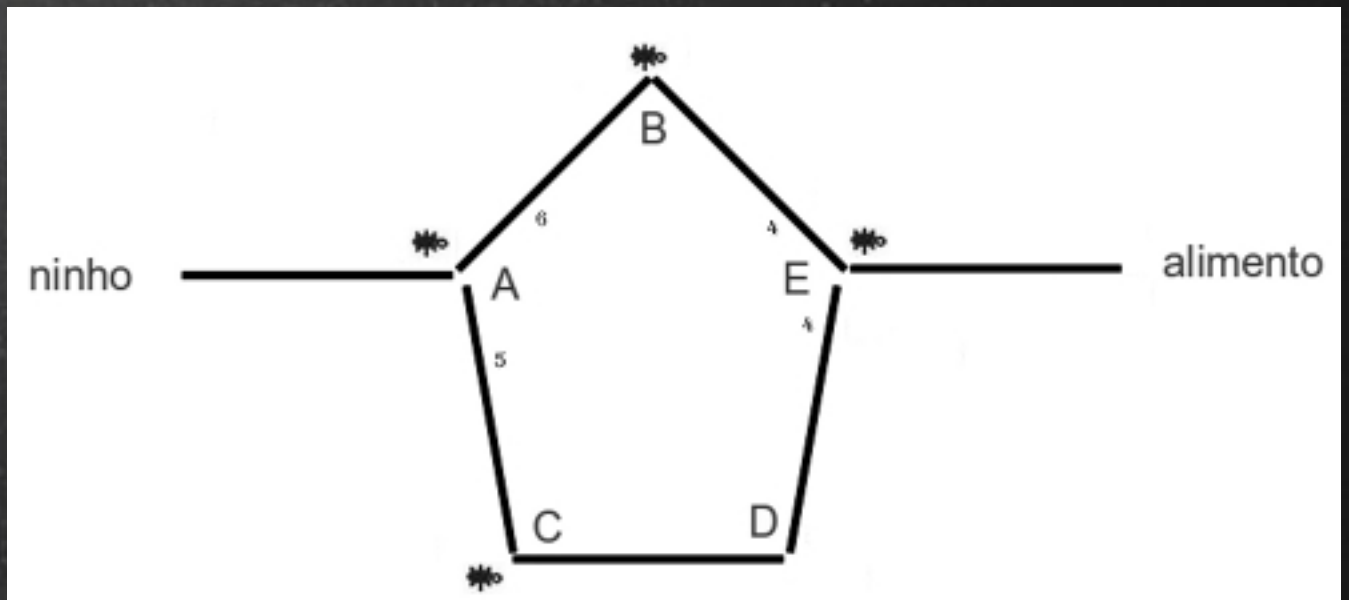





Passo 10

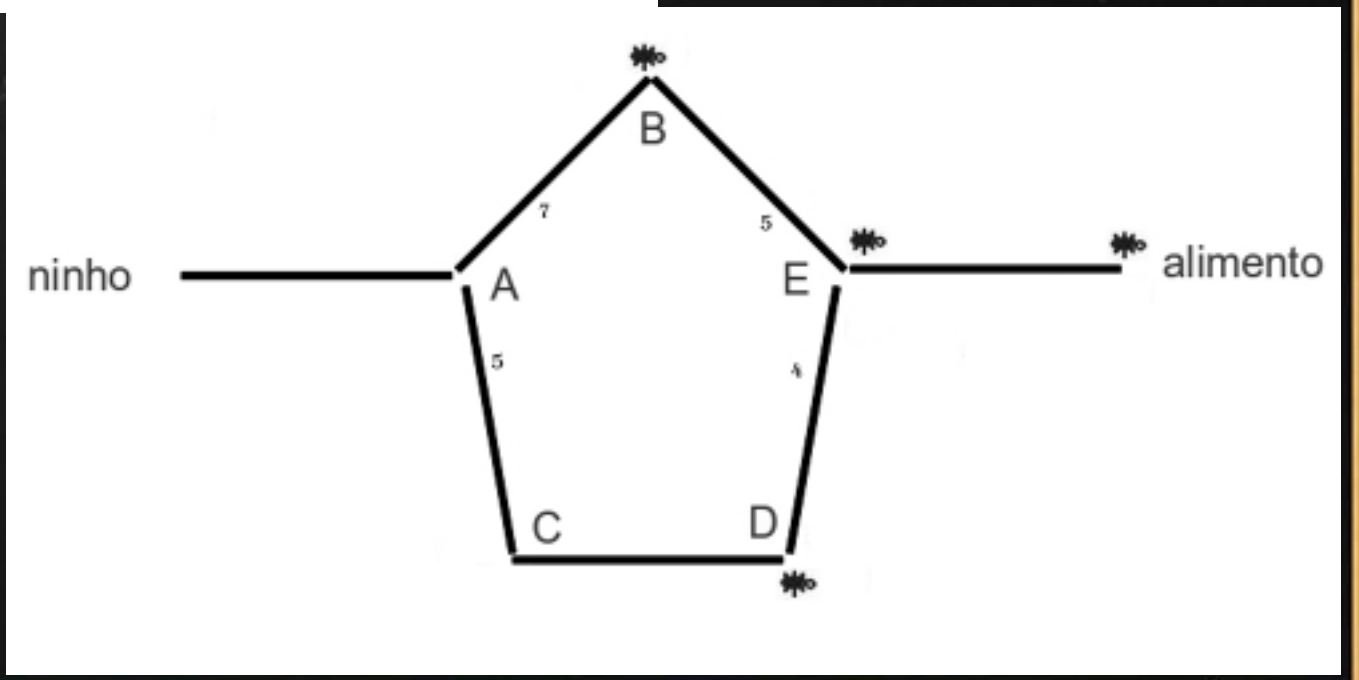
Passo 11

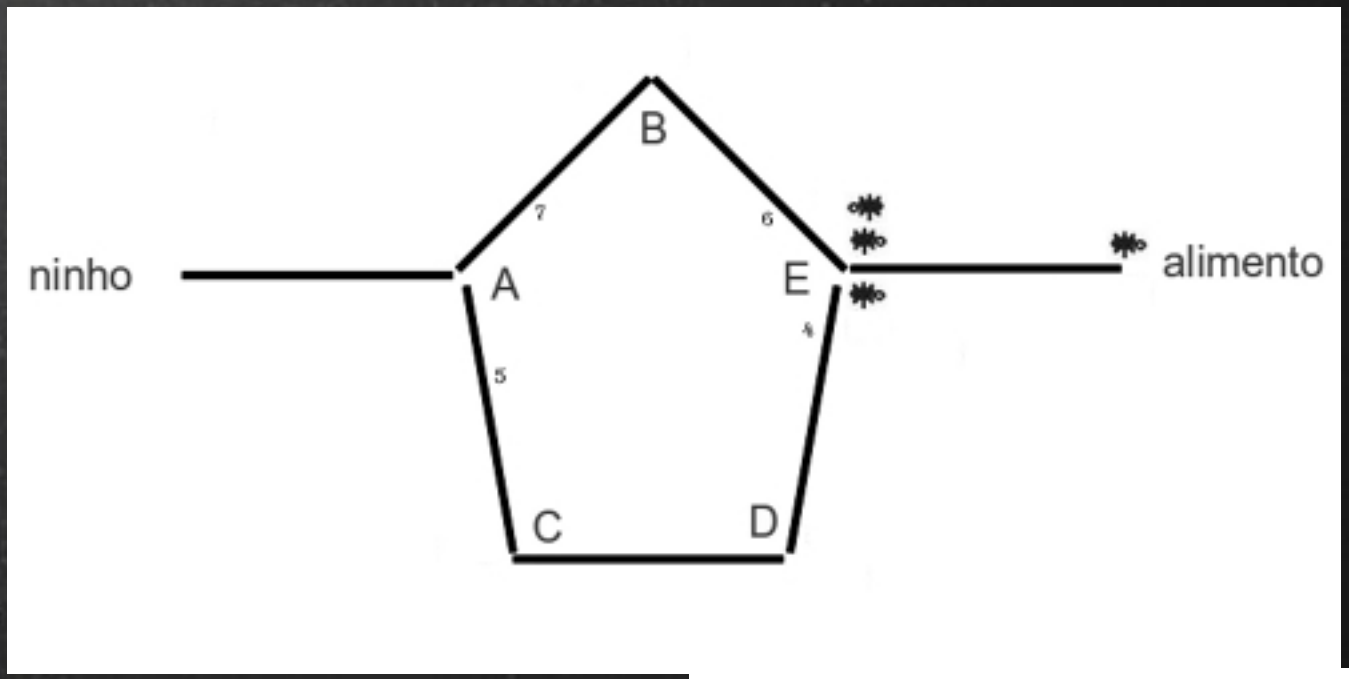





Passo 12

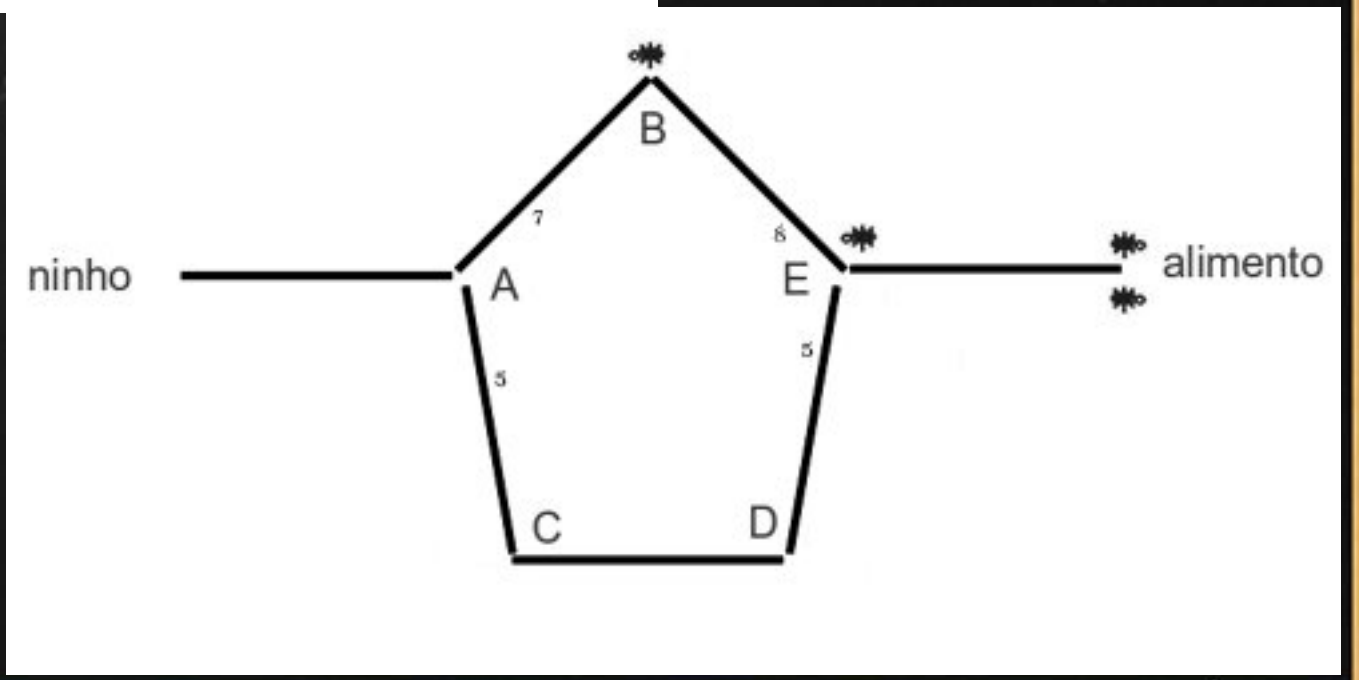
Passo 13



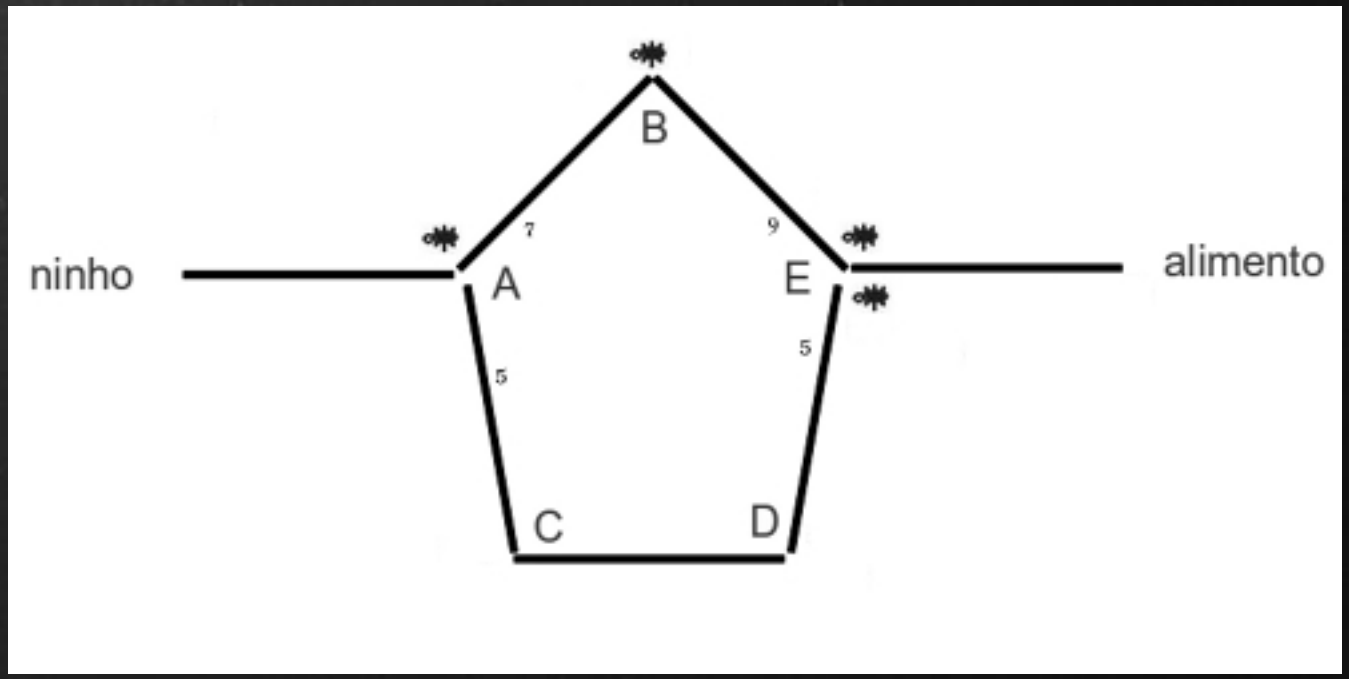



Passo 14

Passo 15



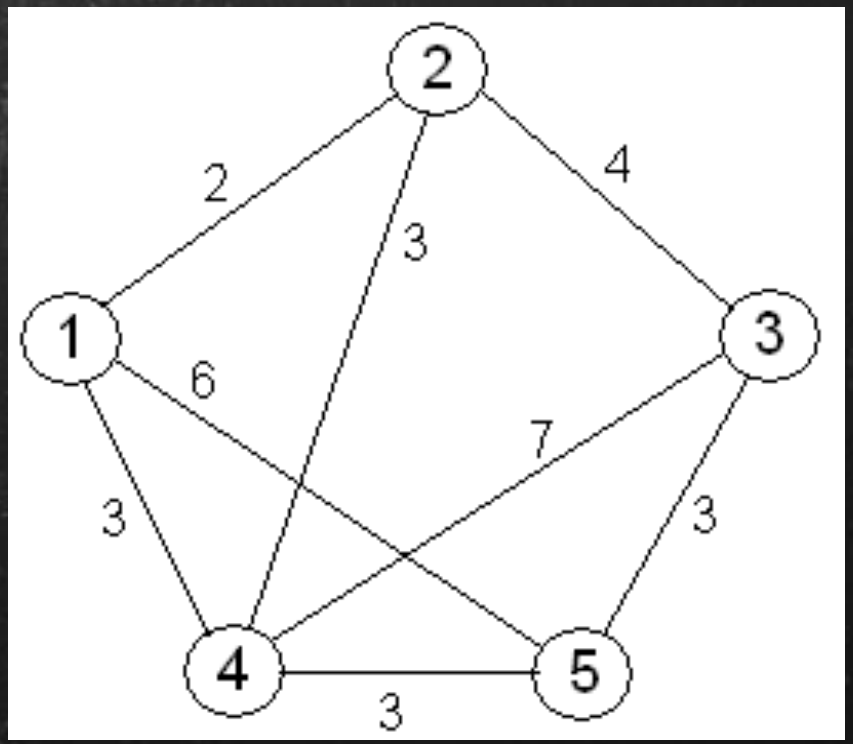
# Ideia do ACO




Passo 16

# Caixeiro Viajante

Traveling Salesman Problem (TSP)




Se existem 2 cidades, A e B, considerando que o caixeiro inicialmente está em A, tem-se apenas 1 solução possível que é a distancia entre A e B. Se existem 3 cidades, têm-se 2 caminhos possíveis, ABC ACB. 4 cidades indicam 6 caminhos possíveis, ABCD ABDC ACBD ACDB ADBC ADCB, ou seja, (n – 1)! possíveis caminhos diferentes.

# Ant Colony Optimization (ACO)

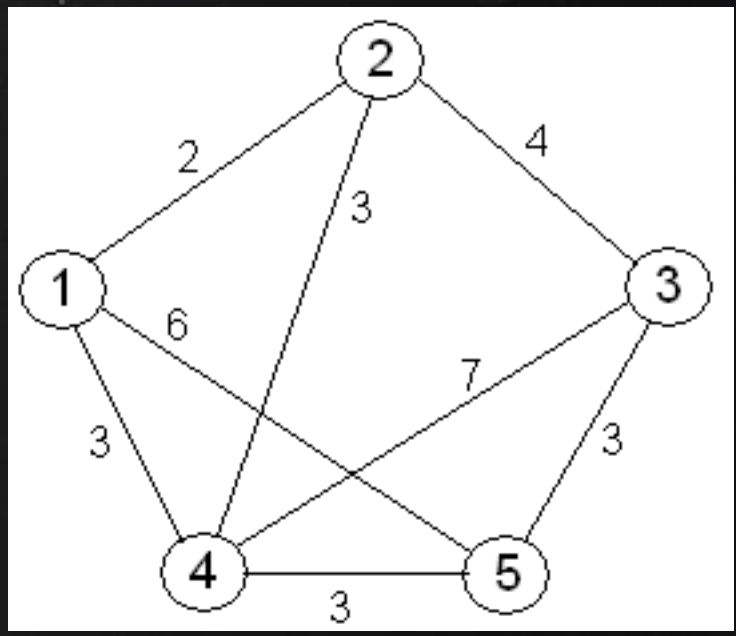
Algoritmos inspirados no comportamento das formigas na busca por alimentos foram introduzidos por Marco Dorigo em sua tese de doutorado. Esse tipo de algoritmo é conhecido na literatura como Ant Colony Optimization (ACO).

Ant System foi o primeiro exemplo de ACO proposto por Dorigo.

Explora a característica do feromônio depositando diferentes quantidades de substância no caminho (dependendo da qualidade do caminho encontrado).

# Ant System

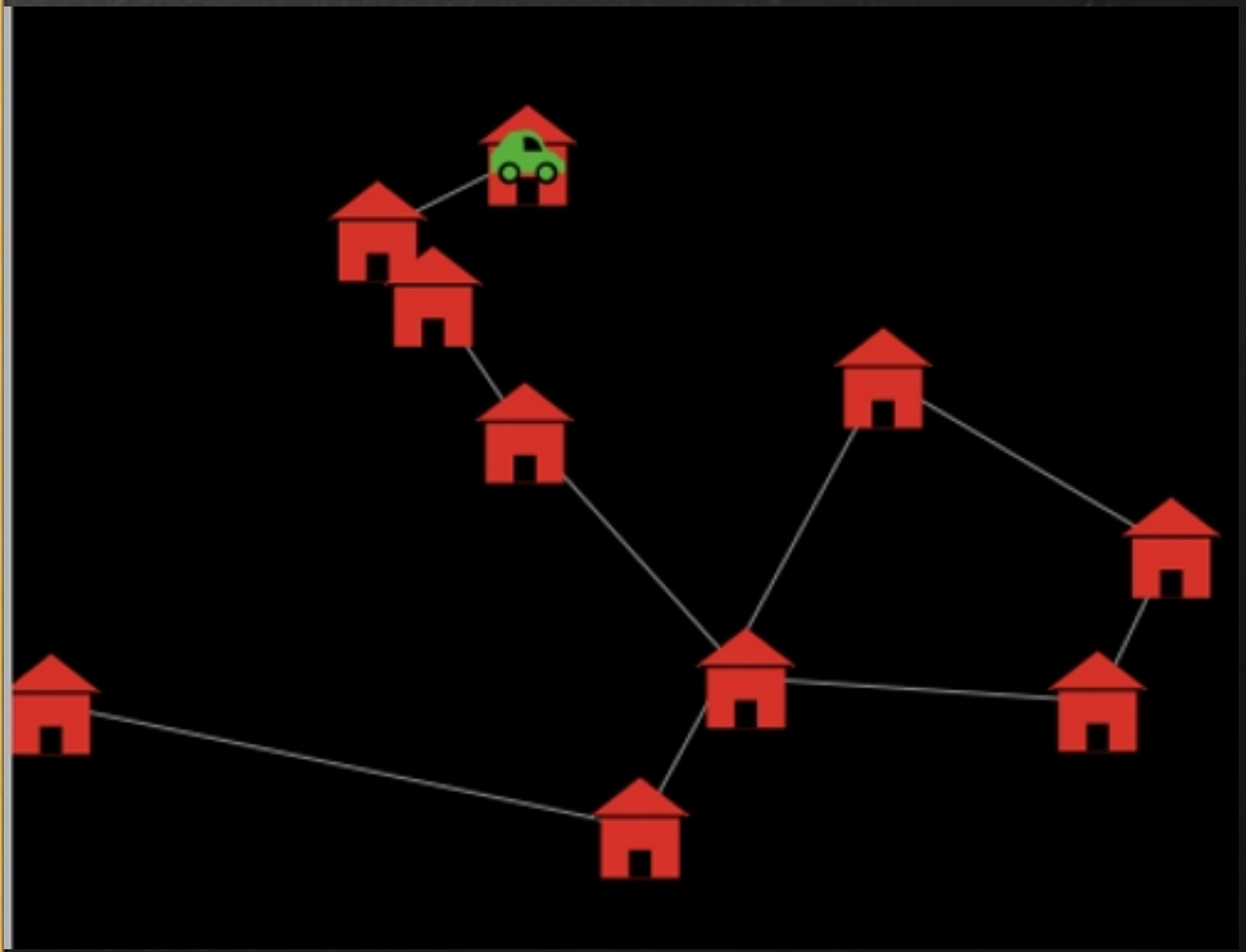
- Gera uma solução (caminho)
- Avalia essa solução
- Deposita sobre cada aresta desse caminho, uma quantidade x de feromônio.
- Sendo x uma função onde se o caminho é menor, x possui maior valor.




Caminho 12345 = 16
Caminho 12354 = 12
Caminho 12435 = 15
Caminho 12453 = 11
Caminho 14235 = 13
....

Ex: x = 1 / f(caminho)

# Aplicação



Exemplo de aplicação do Ant System no TSP.

feita em NetLogo.

# Código

```
para cada iteracao {
    para cada agente {
        s <- gerar_solucao(f)
        q <- avaliar_solucao(s)
        x <- 1 / q
        Δf <- Δf + depos(s, x)
    }
    f <- f + Δf
    Δf <- ε
    f <- f * tx
}
```

f:

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0.8 | 0.2 | 0.4 | 0.5 | 0.4 | 0.6 | 0.3 |
| 2 | 0.8 | 0 | 0.4 | 0.4 | 0.6 | 0.3 | 0.5 | 0.8 |
| 3 | 0.2 | 0.4 | 0 | 0.2 | 0.6 | 0.3 | 0.7 | 0.7 |
| 4 | 0.4 | 0.4 | 0.2 | 0 | 0.2 | 0.5 | 0.5 | 0.2 |
| 5 | 0.5 | 0.6 | 0.6 | 0.2 | 0 | 0.5 | 0.4 | 0.4 |
| 6 | 0.4 | 0.3 | 0.3 | 0.5 | 0.5 | 0 | 0.2 | 0.1 |
| 7 | 0.6 | 0.5 | 0.7 | 0.5 | 0.4 | 0.2 | 0 | 0.9 |
| 8 | 0.3 | 0.8 | 0.7 | 0.2 | 0.4 | 0.1 | 0.9 | 0 |

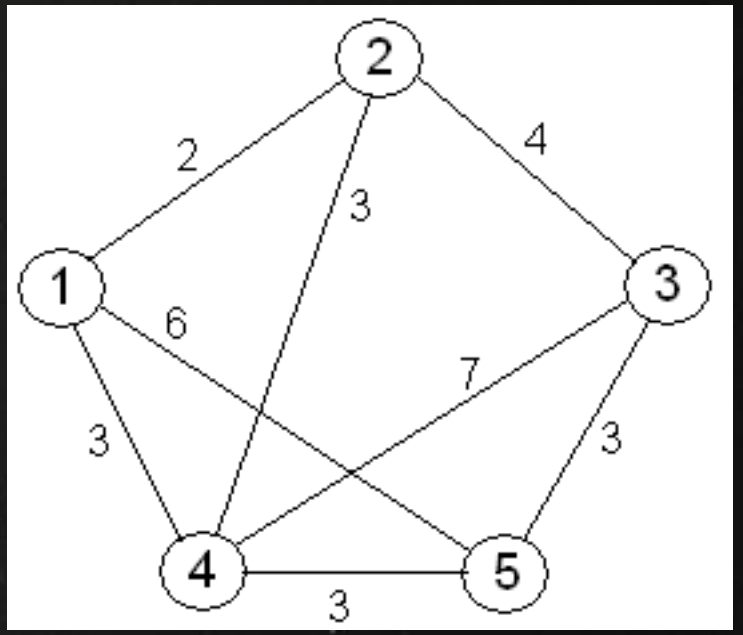
tx: (0, 1)

# Heurística

```
s <- gerar_solucao(f)

gerar_solucao(f) {
    escolher_proximo(f)
}

gerar_solucao(f, d) {
    v <- 1 / d
    escolher_proximo(f + v)
}
```




| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | 0 | 0.8 | 0 | 0.4 | 0.5 |
| 2 | 0.8 | 0 | 0.4 | 0.4 | 0 |
| 3 | 0 | 0.4 | 0 | 0.2 | 0.6 |
| 4 | 0.4 | 0.4 | 0.2 | 0 | 0.2 |
| 5 | 0.5 | 0 | 0.6 | 0.2 | 0 |

# Ant Colony Optimization

Cassio Conti
cassio@inf.ufsc.br