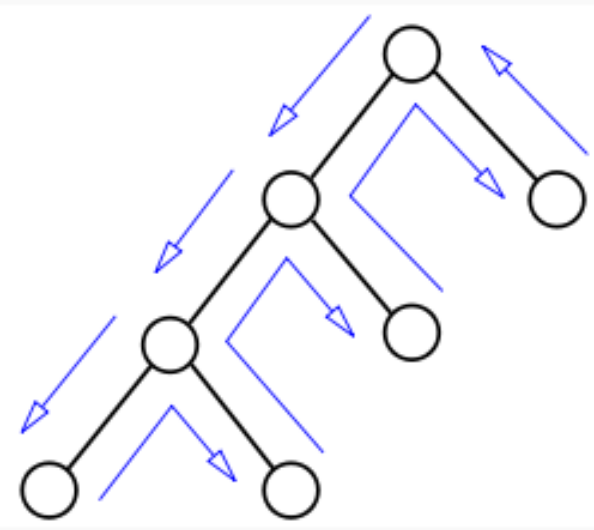
Backtracking

Marcos Castro

Backtracking

- De acordo com o Wikipédia:
  - Backtracking é um tipo de algoritmo que representa um refinamento da busca por força bruta, em que múltiplas soluções podem ser eliminadas sem serem explicitamente examinadas.



Backtracking

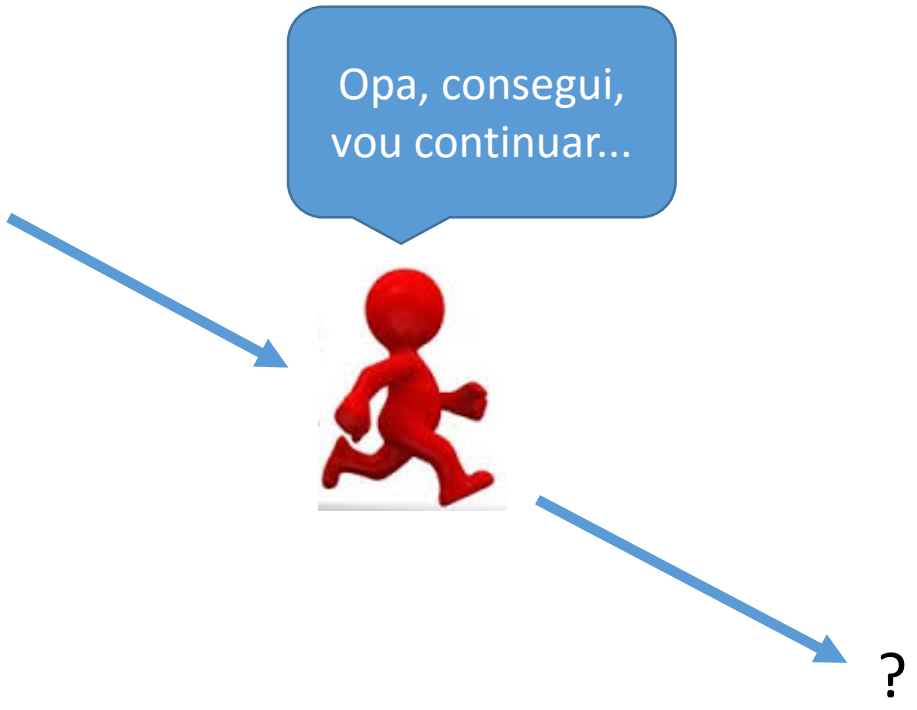
- A recursividade pode ser usada para resolver problemas cuja solução é do tipo tentar todas as alternativas possíveis.
- O backtracking executa podas quando não é possível encontrar uma solução pelo caminho escolhido.

Backtracking

- NULL quer encontrar a saída do labirinto...

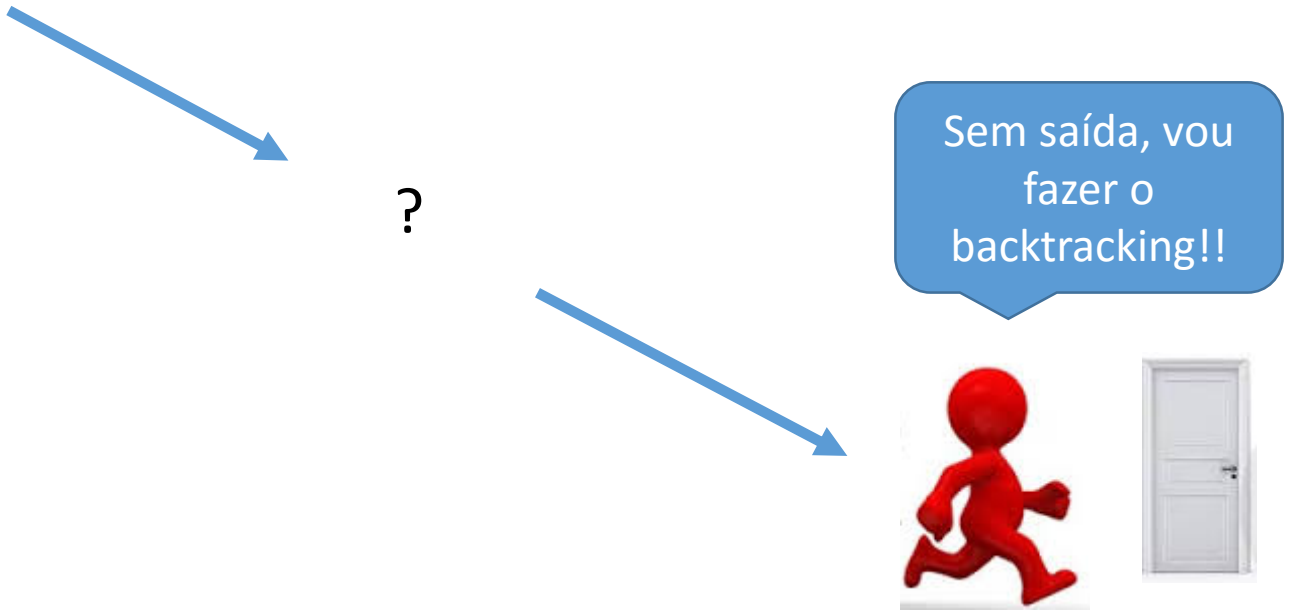
Backtracking

- NULL quer encontrar a saída do labirinto...



Backtracking

- NULL quer encontrar a saída do labirinto...

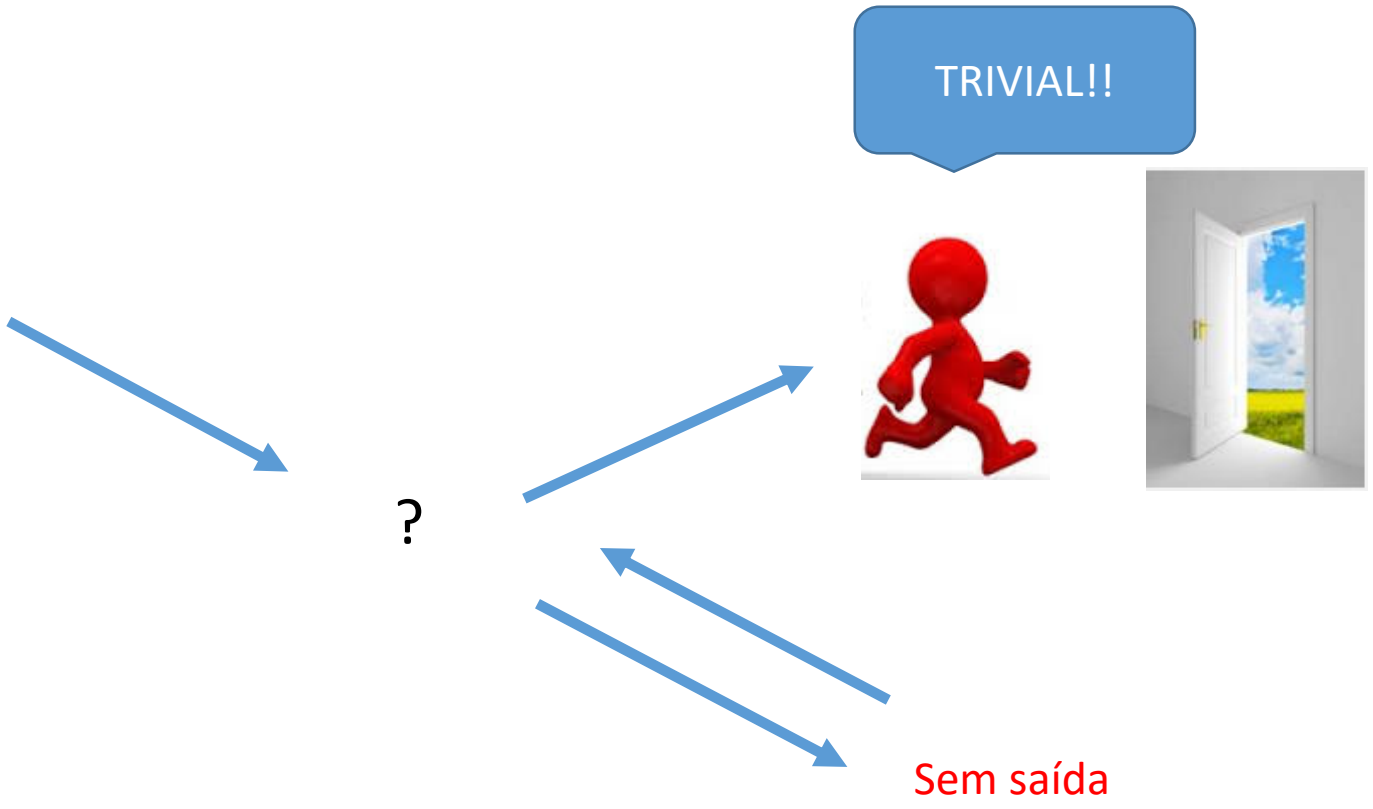


Backtracking

- NULL quer encontrar a saída do labirinto...

Backtracking

- NULL quer encontrar a saída do labirinto...

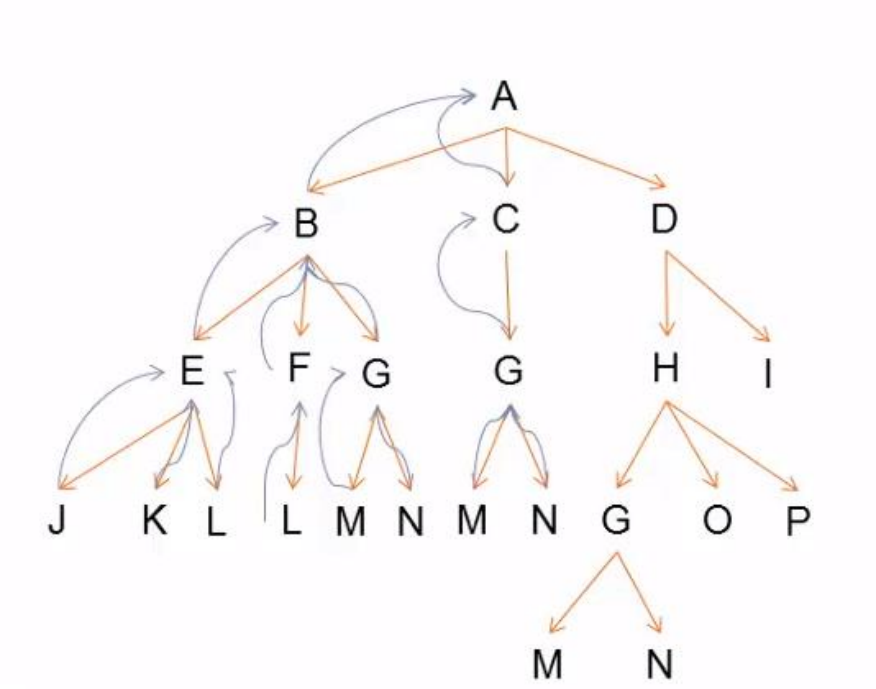


Backtracking

- Imagine um problema com as seguintes características:
  - Você tem que fazer uma série de decisões...
  - Você não tem informações suficientes para saber o que escolher...
  - Cada decisão leva a um novo conjunto de escolhas...
  - Alguma sequência de escolhas pode ser uma solução para o seu problema...
  - Backtracking pode ser uma boa forma de experimentar várias sequências de decisões até encontrar uma que funciona!

Backtracking

- A busca em profundidade (DFS) explora tanto quanto possível um ramo antes de retroceder. É o que acontece no backtracking!



Gerando todos os subconjuntos

- Problema: gerando todos os subconjuntos
  - Temos um conjunto S = {1 ... N}
  - Objetivo: imprimir todos os subconjuntos a partir de N elementos.
  - Para S = {1, 2} (N = 2) temos os subconjuntos: {1,2}, {1}, {2}, {}
  - {1, 2} é o mesmo que {2, 1}
  - O número de possíveis subconjuntos é 2^N

Gerando todos os subconjuntos

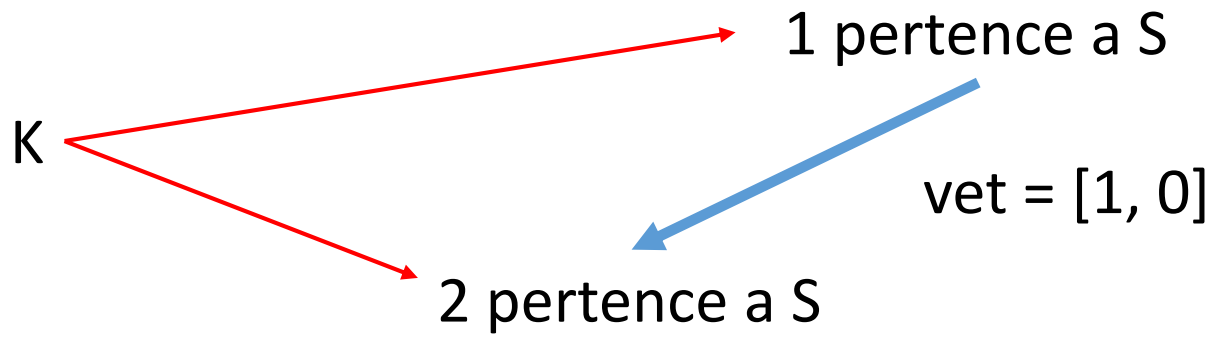
- Ideia: ou o elemento faz parte do subconjunto ou não faz parte.
- Pode-se construir um vetor de bool de tamanho N.
- Para S = {1, 2}, teríamos um vetor inicialmente com {0, 0}.
- Para S = {1, 2, 3}, teríamos um vetor inicialmente com {0, 0, 0}.
- Esse vetor irá nos ajudar a construir todos os subconjuntos.
- Se vet é o vetor de bool, vet[i] indica se o i-ésimo elemento está ou não está no subconjunto.

Gerando todos os subconjuntos

- Uma variável K indicará qual elemento será colocado ou removido do subconjunto.
- S(K) = (true, false)
- Seja F nossa função para gerar todos os subconjuntos, ela pode ser definida como F(K, N).
- Inicialmente F é chamada da seguinte forma:
  - F(1, N) // indexando a partir do 1

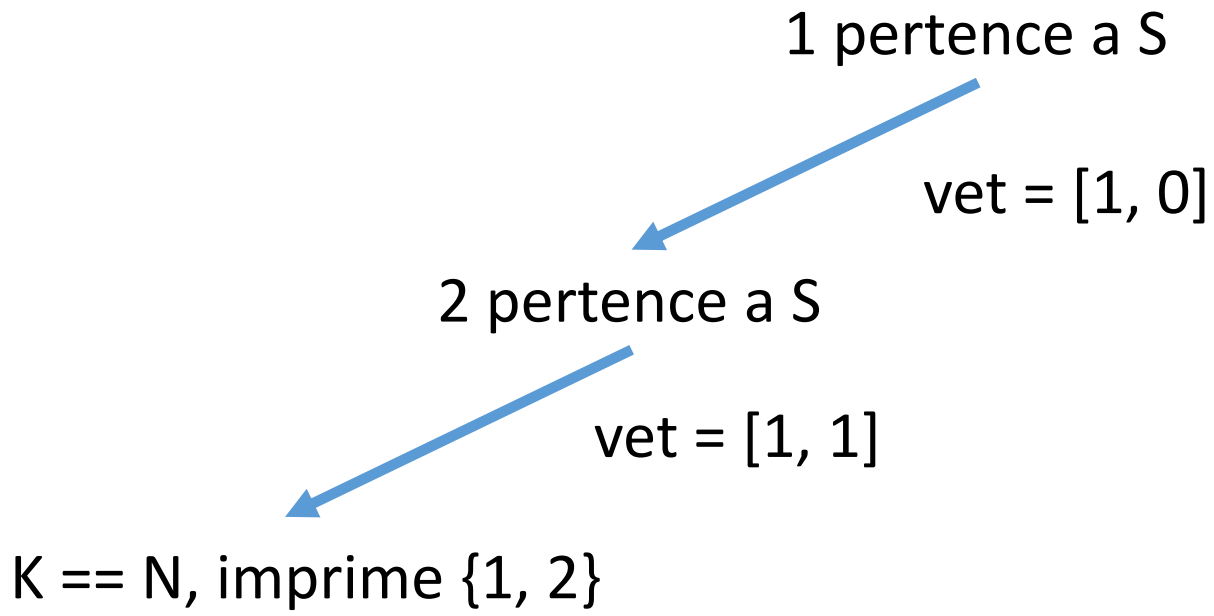
Gerando todos os subconjuntos

- Construindo a árvore para S = {1, 2}, N = 2



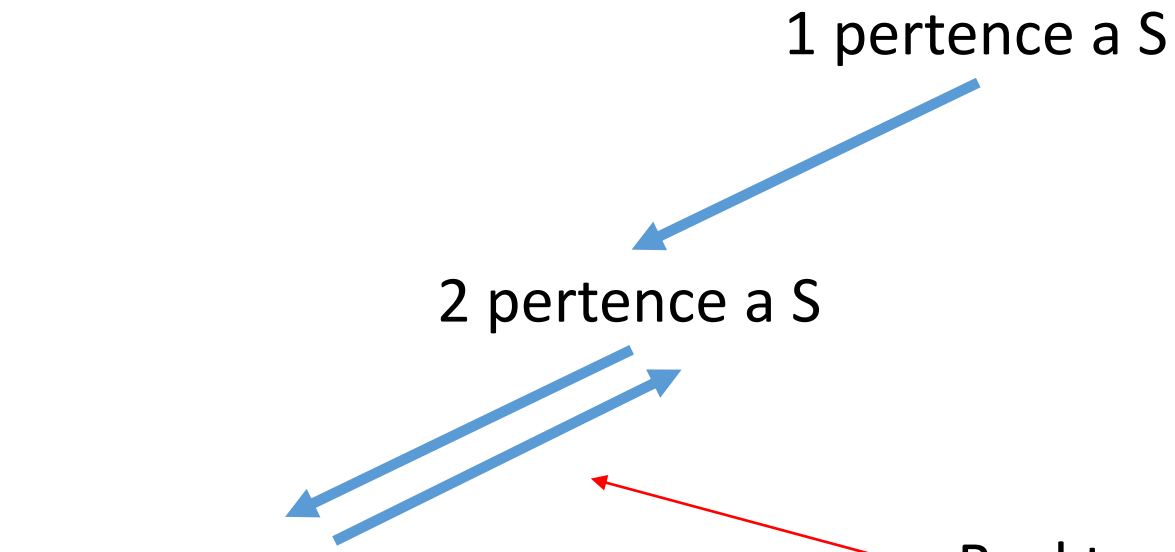
Gerando todos os subconjuntos

- Construindo a árvore para S = {1, 2}, N = 2



Gerando todos os subconjuntos

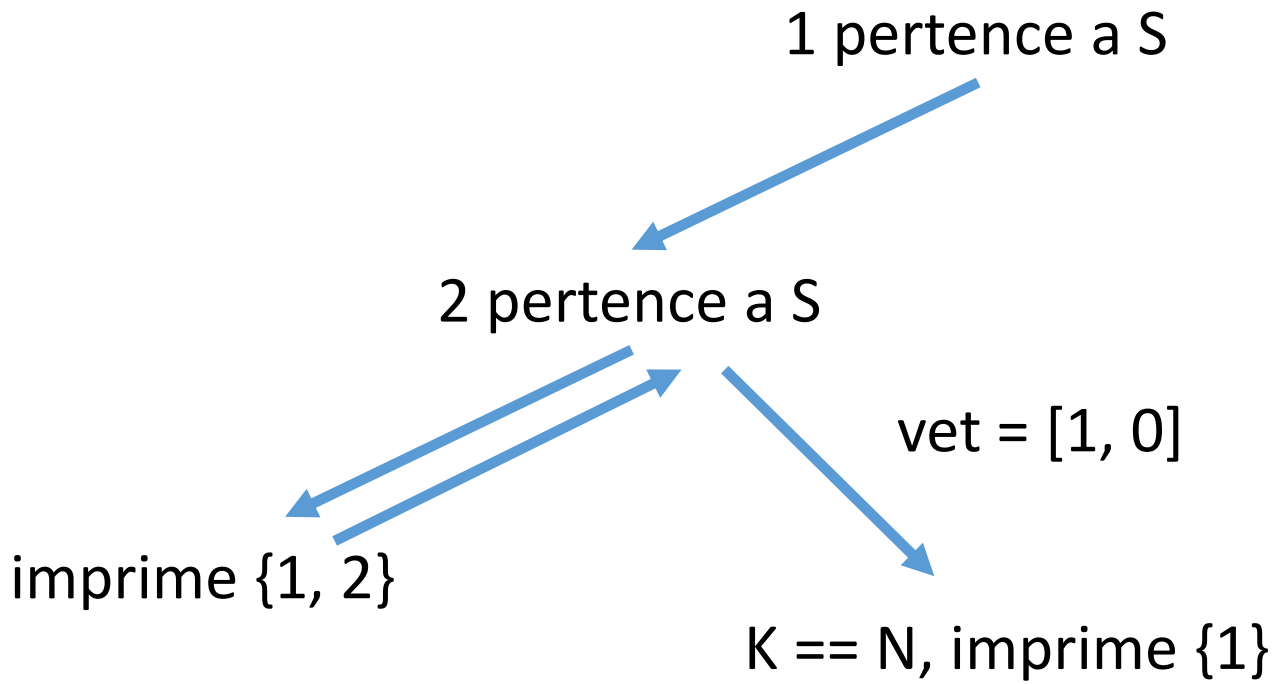
- Construindo a árvore para S = {1, 2}, N = 2



Backtracking, o vet até então era [1, 1],
quando retrocede, fazemos vet[K] = 0,
se K = 2, teremos que vet = [1, 0]

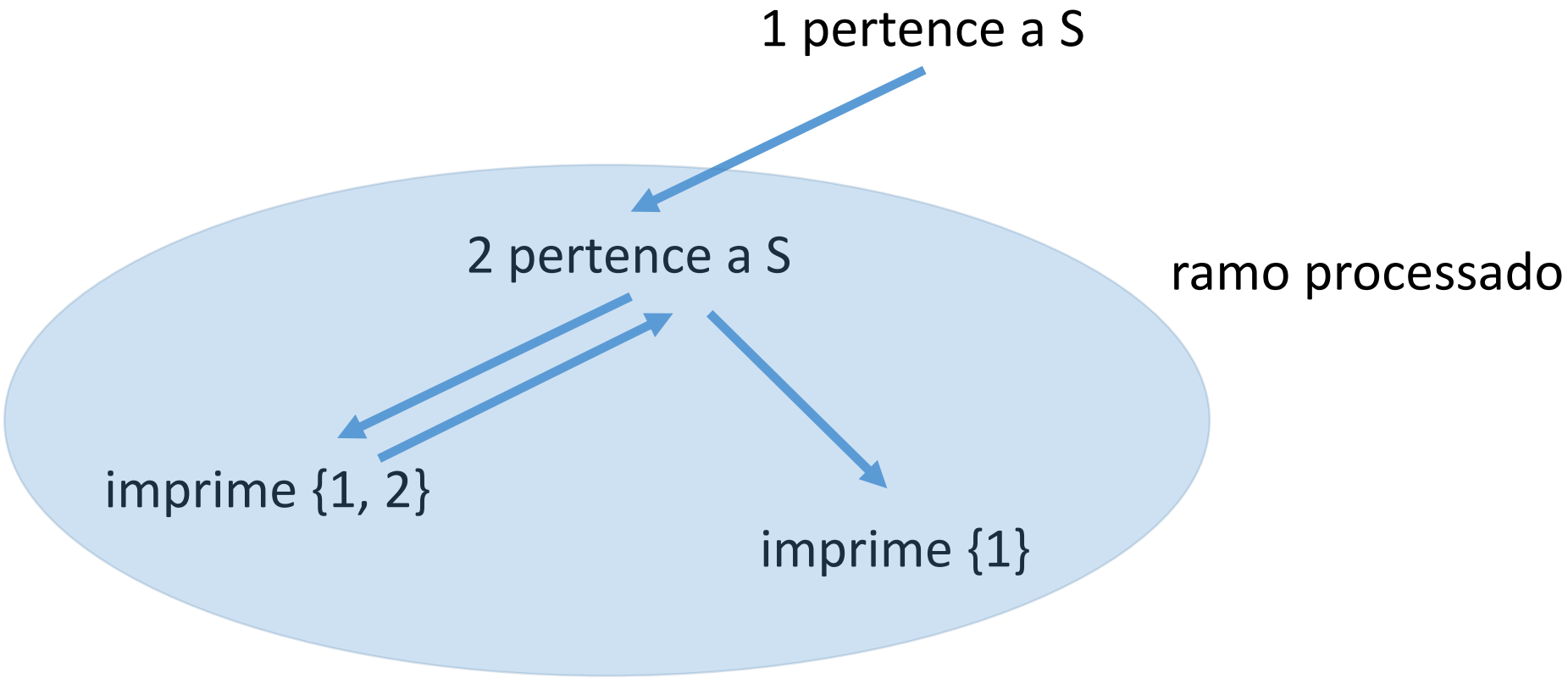
Gerando todos os subconjuntos

- Construindo a árvore para S = {1, 2}, N = 2



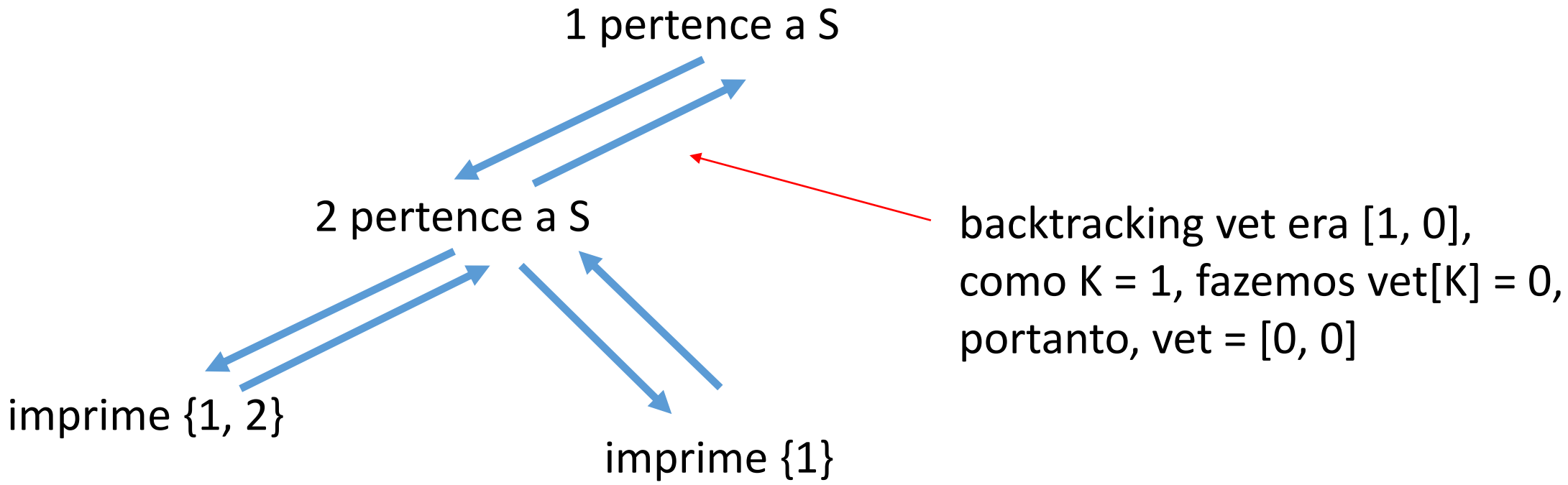
Gerando todos os subconjuntos

- Construindo a árvore para S = {1, 2}, N = 2



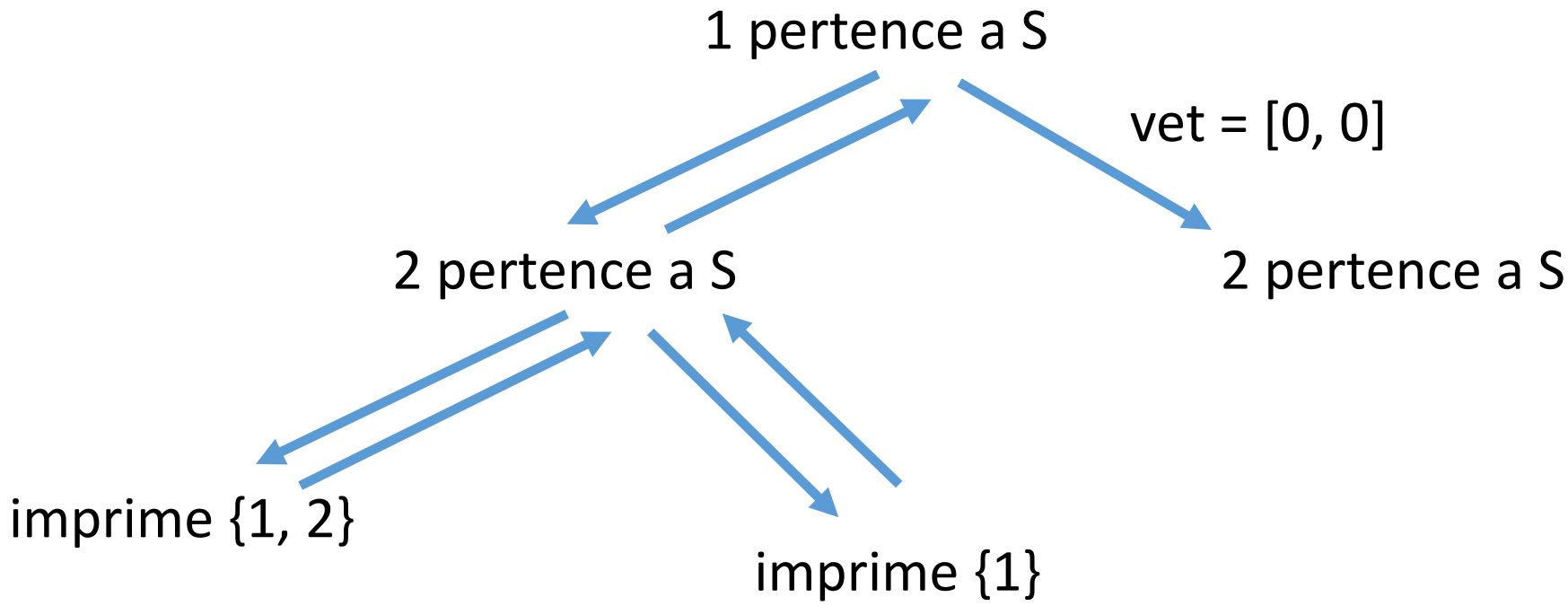
Gerando todos os subconjuntos

- Construindo a árvore para S = {1, 2}, N = 2



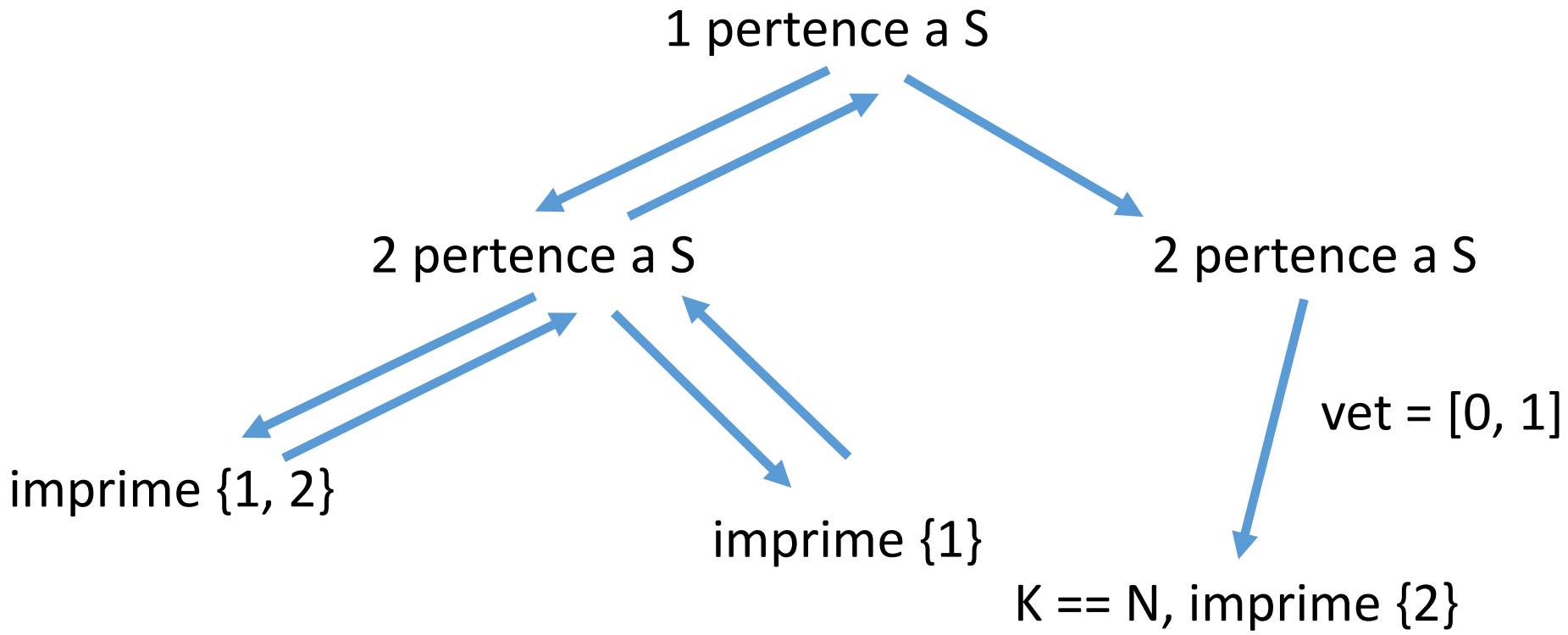
Gerando todos os subconjuntos

- Construindo a árvore para S = {1, 2}, N = 2



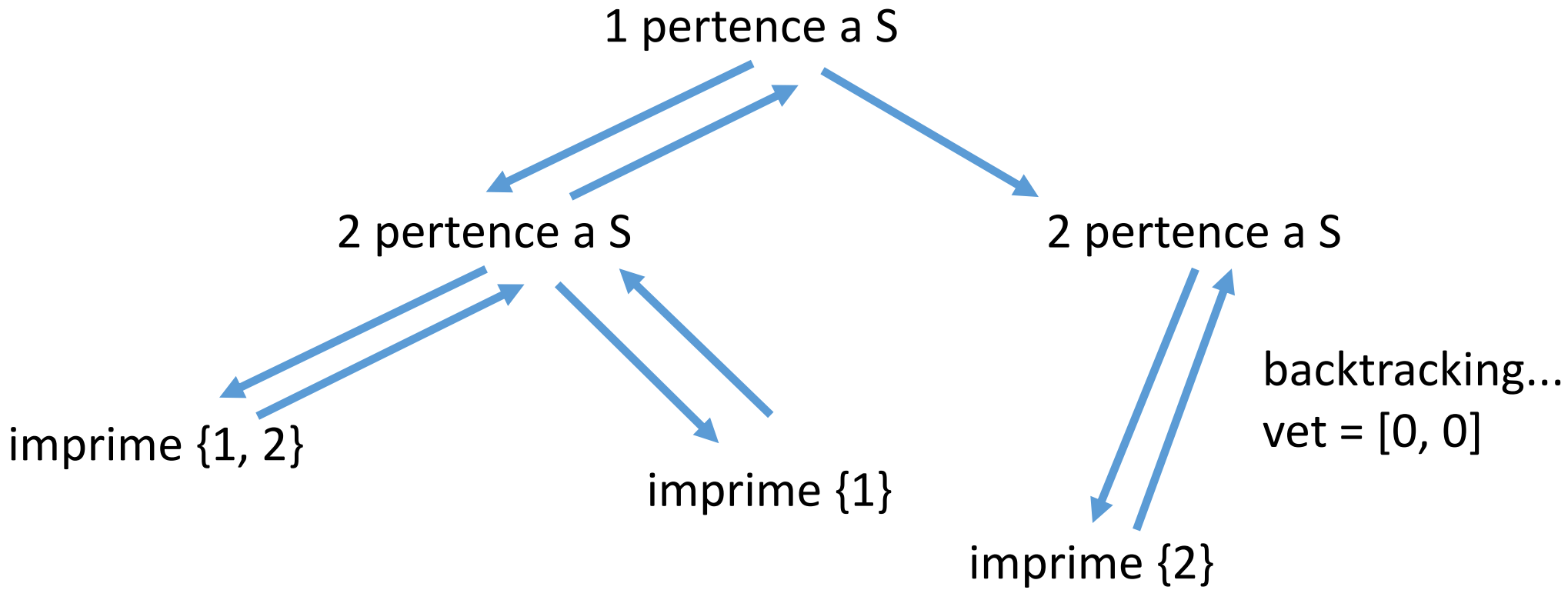
Gerando todos os subconjuntos

- Construindo a árvore para S = {1, 2}, N = 2



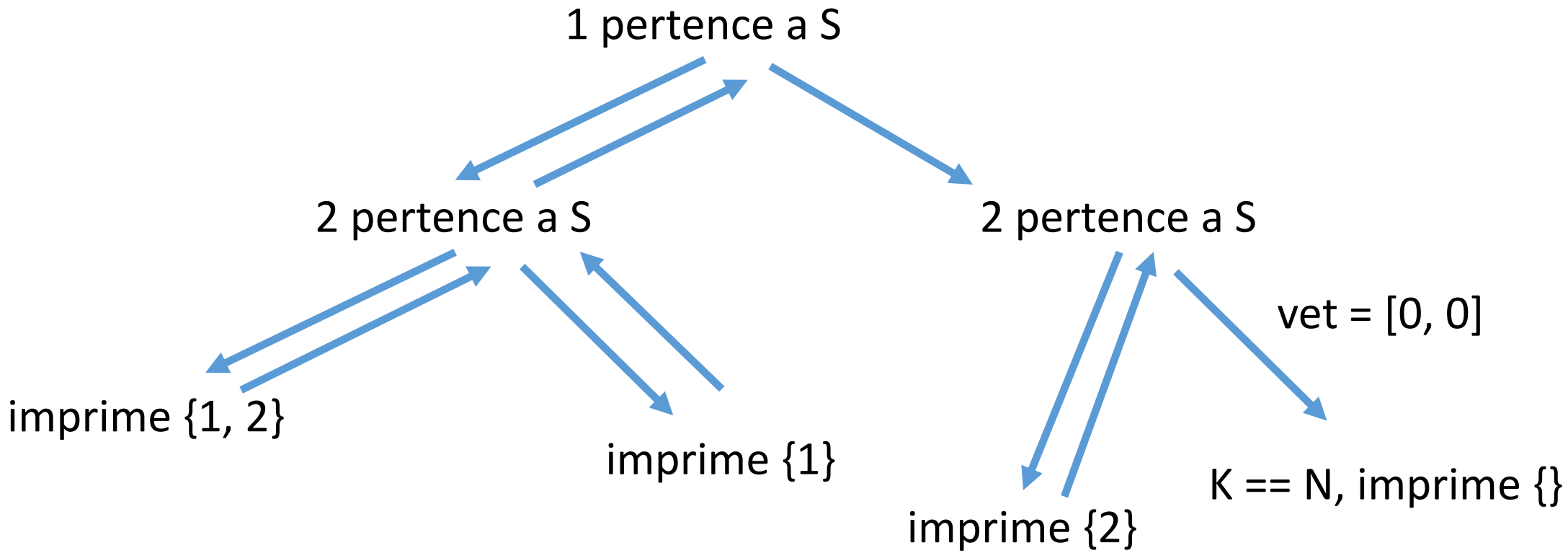
Gerando todos os subconjuntos

- Construindo a árvore para S = {1, 2}, N = 2



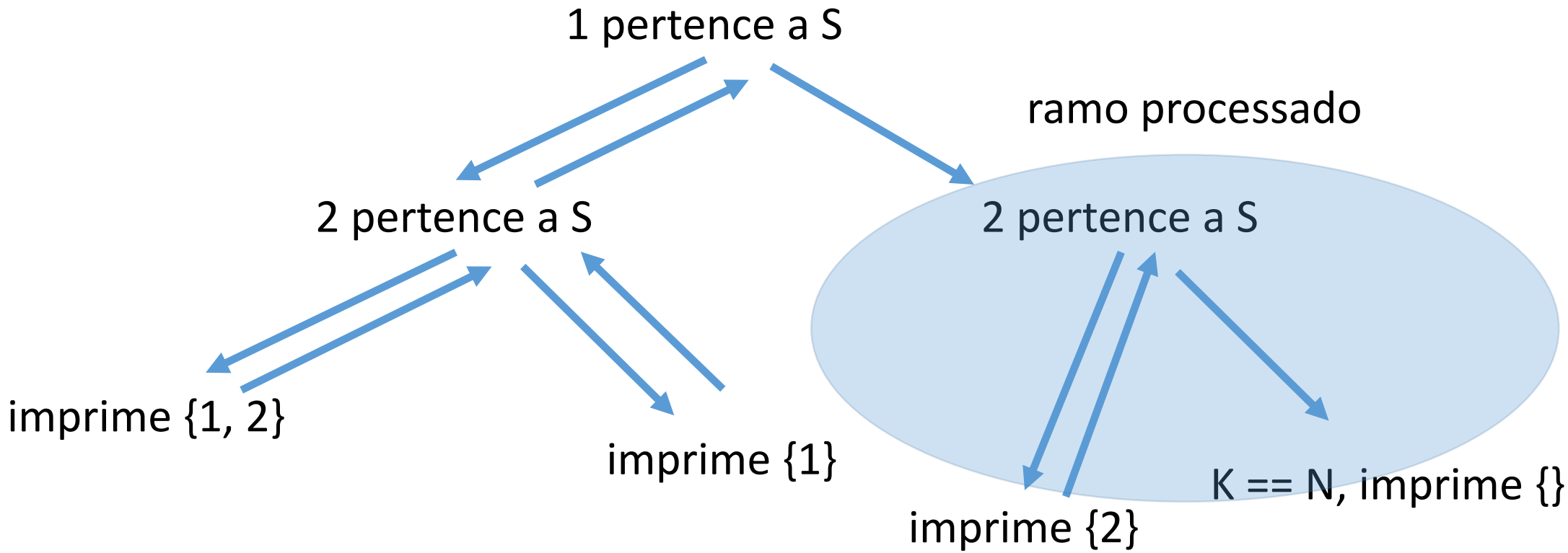
Gerando todos os subconjuntos

- Construindo a árvore para S = {1, 2}, N = 2



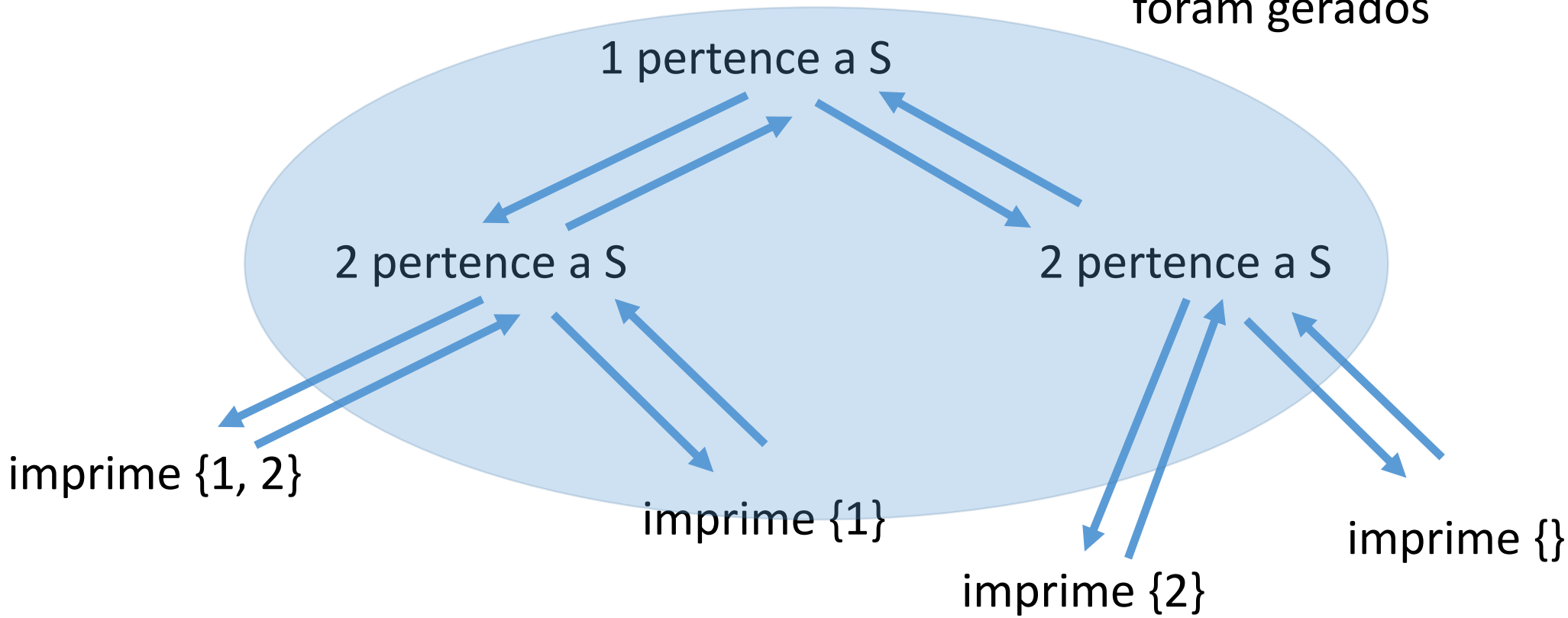
Gerando todos os subconjuntos

- Construindo a árvore para S = {1, 2}, N = 2



Gerando todos os subconjuntos

- Construindo a árvore para S = {1, 2}, N = 2



Gerando todos os subconjuntos

- Construindo a função F (indexando a partir do 0)...

```
1  se (K == N)
2  {
3      // imprime o subconjunto
4      para i = 0 até N - 1
5      {
6          se (vet[i] == true)
7              imprime (i + 1)
8      }
9  }
```

Gerando todos os subconjuntos

- Construindo a função F ...

```
10   senao
11   {
12       vet[K] = 1
13       F(K + 1, N)
14       vet[K] = 0
15       F(K + 1, N)
16   }
```

Dúvidas?

mcastrosouza@live.com

twitter.com/mcastrosouza