SCC 5789 – Base de Dados

Profa. Dra. Cristina Dutra de Aguiar Ciferri

Árvore R

Luiz Olmes Carvalho

Apresentação

- Conceitos introdutórios.
- Estrutura da Árvore R.
- Consulta.
- Inserção.
- Split.
- Variações da Árvore R.
- Demonstração: applet.

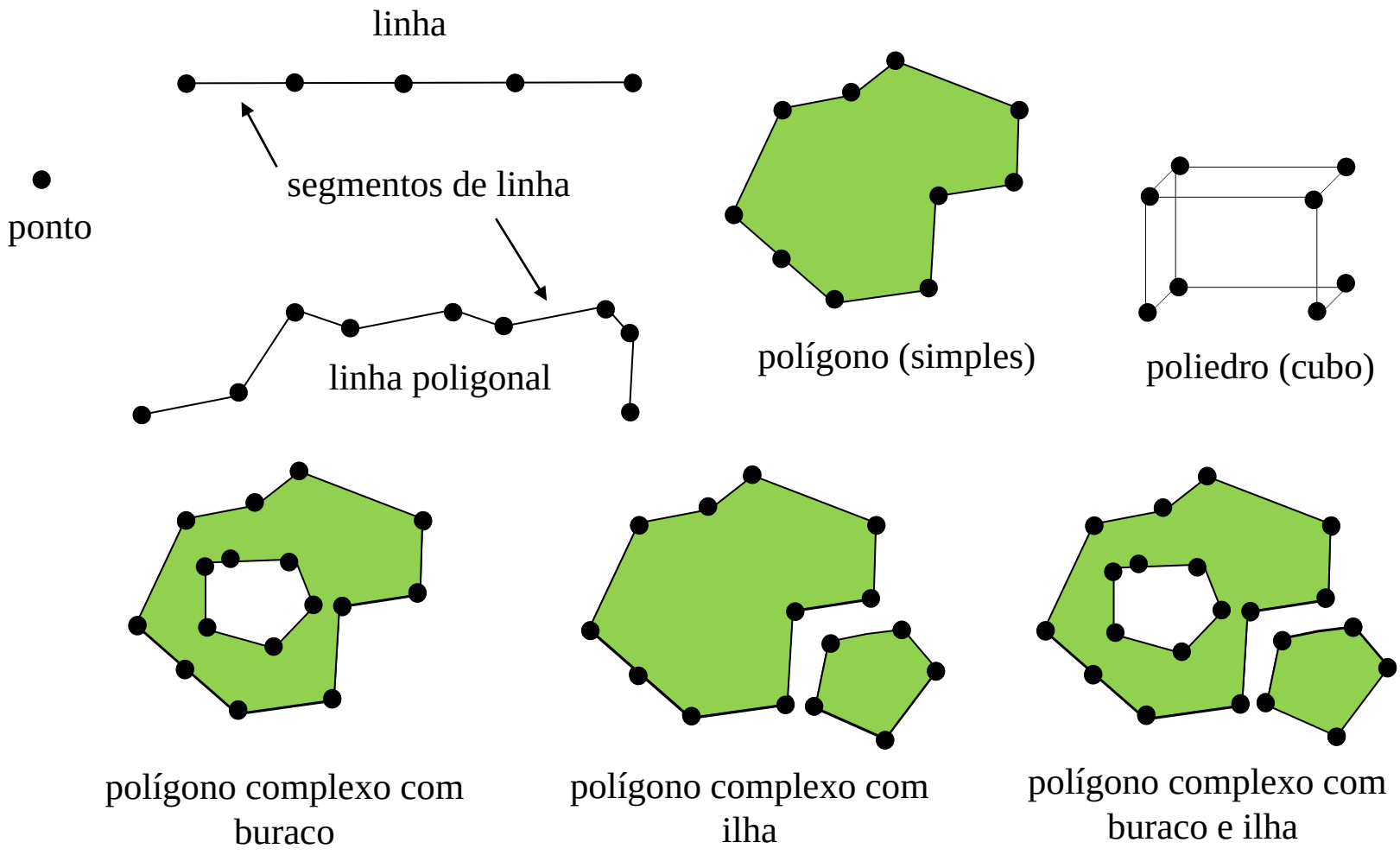
Árvore R

Conceitos Introdutórios

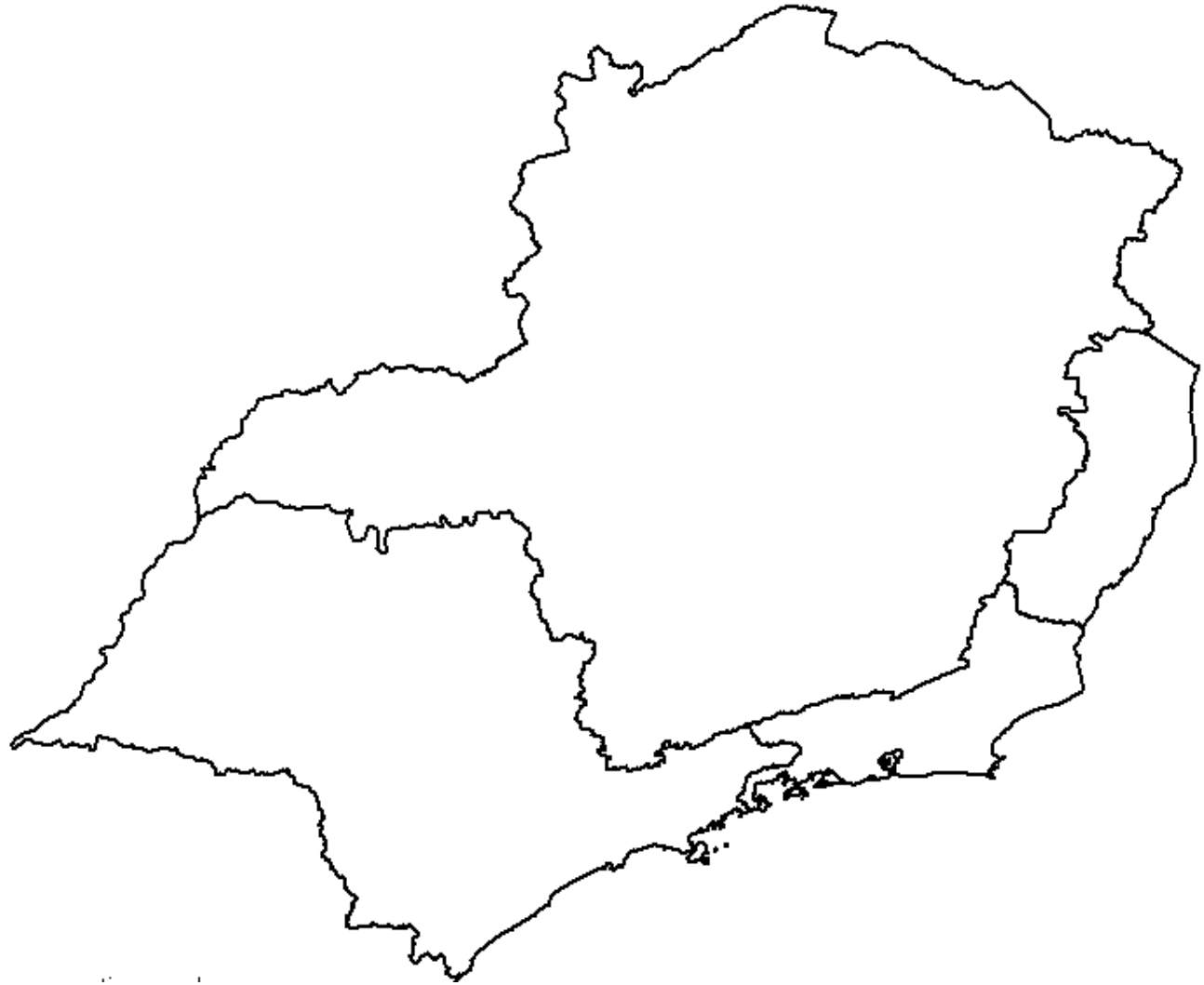
Tipos de dados espaciais

- Ponto: unidade mínima representativa de um objeto espacial.
- Linha: sequência de pontos retilíneos.
- Linha poligonal: sequência de pontos não retilíneos.
- Polígono: sequência fechada de linhas ou linhas poligonais.
- Polígono complexo: polígono com buracos e/ou partes disjuntas.
- Poliedro: sólido composto por um número finito de faces.

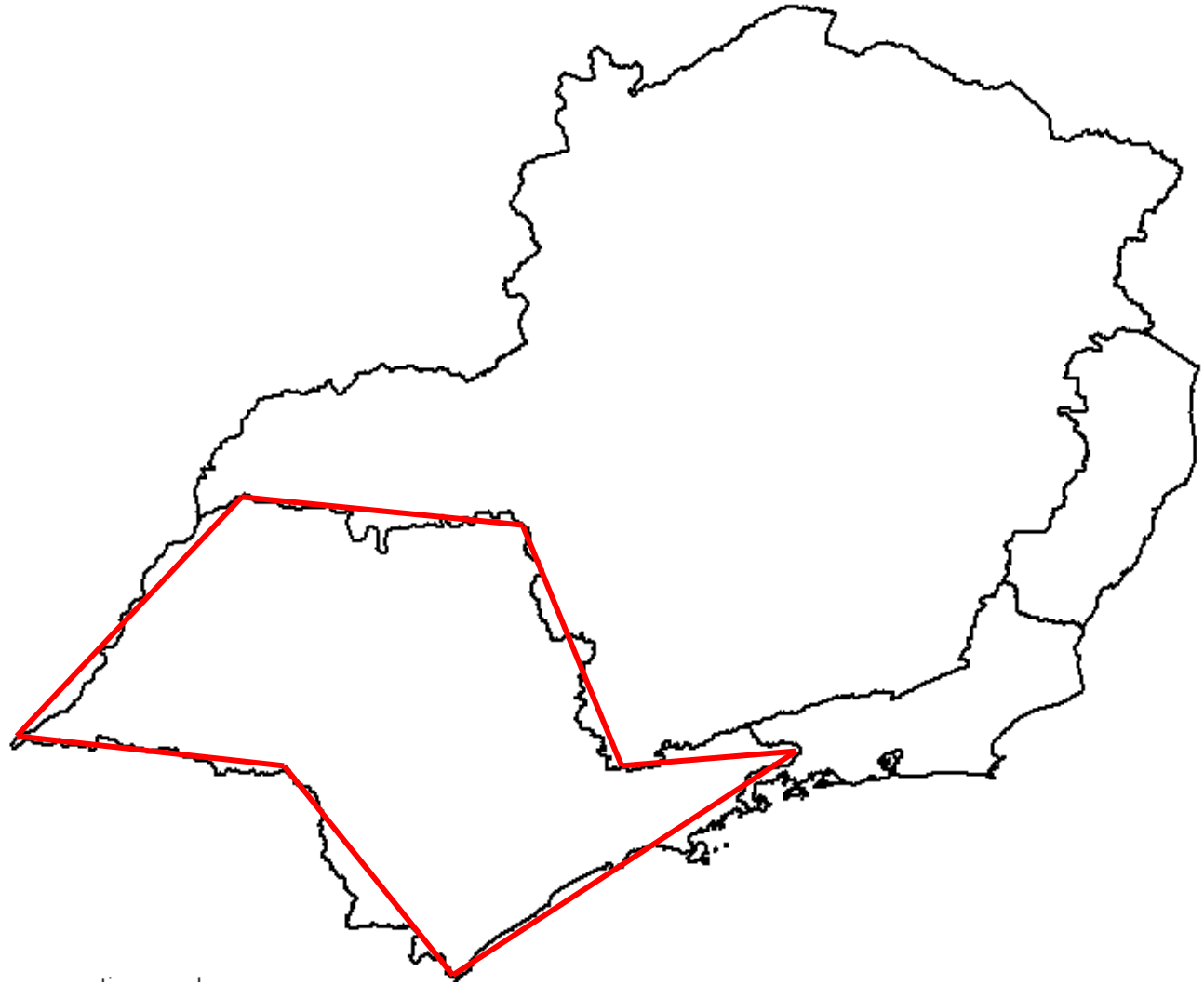
Tipos de dados espaciais



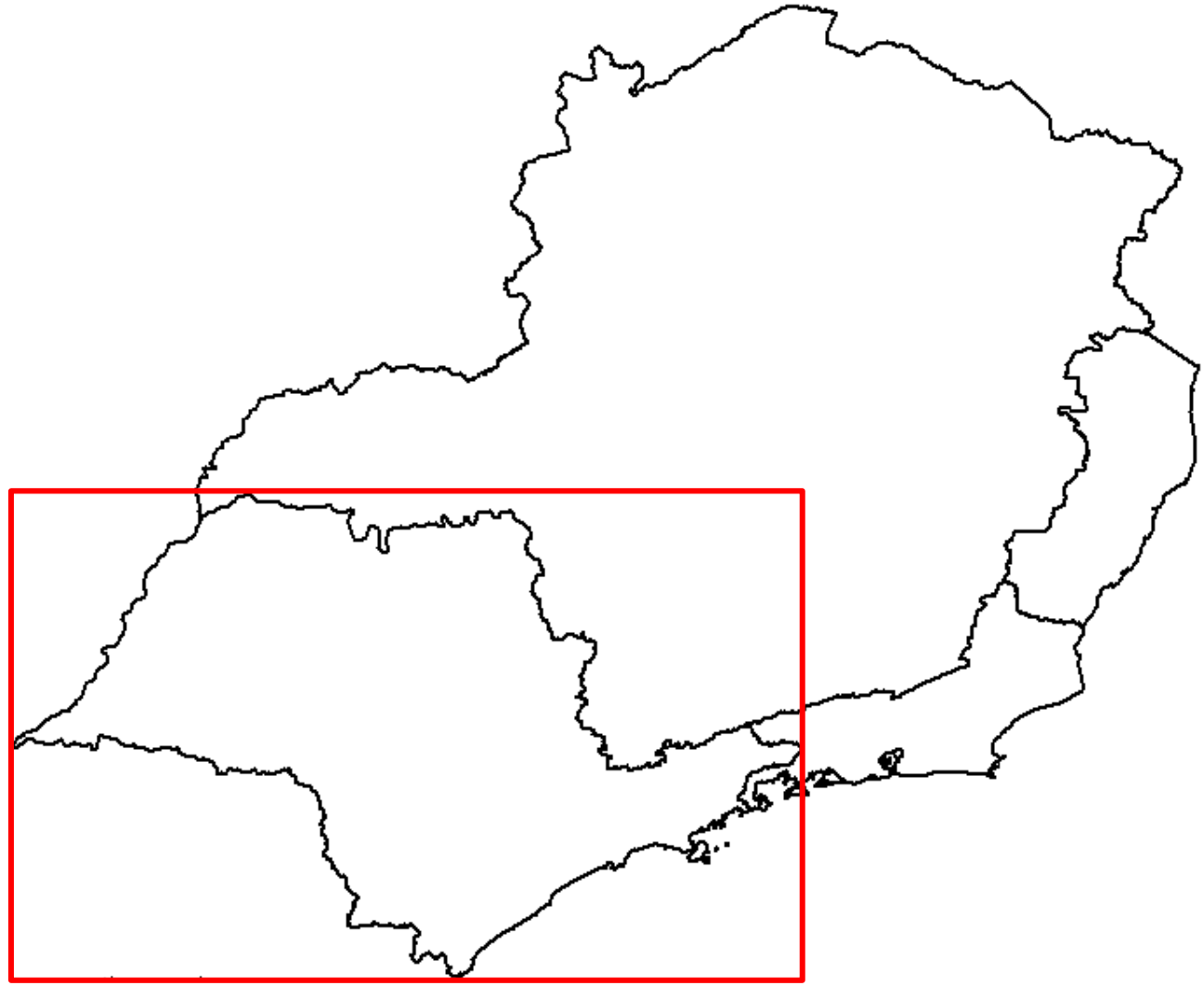
Representação dos dados



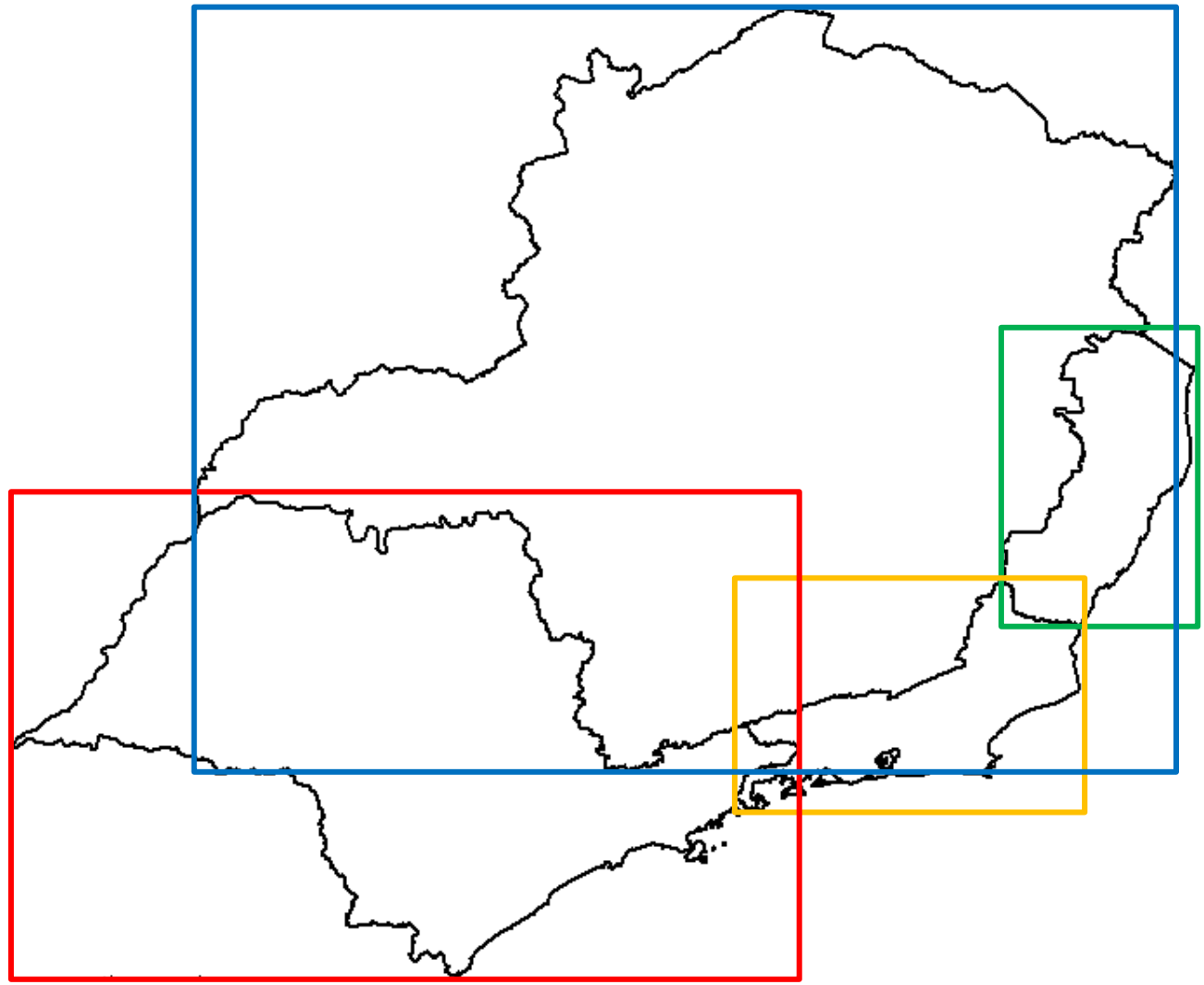
Representação dos dados



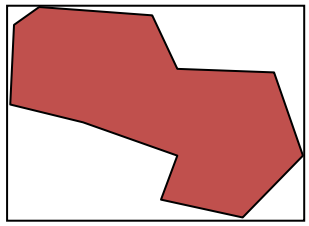
Mininum Bounding Rectangle – MBR



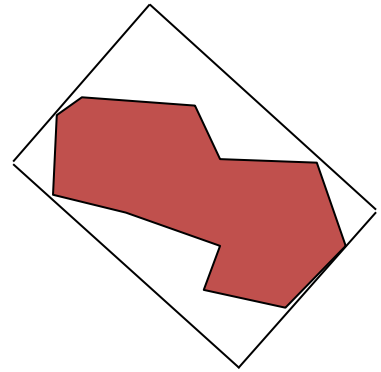
Mininum Bounding Rectangle – MBR



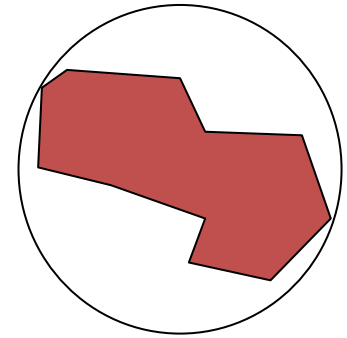
Outras representações conservativas



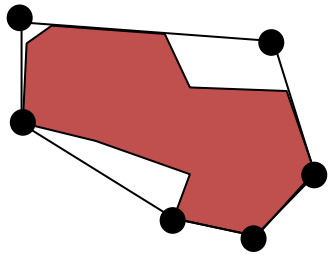
retângulo envolvente mínimo



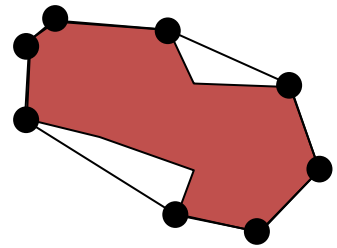
retângulo envolvente mínimo rotacionado



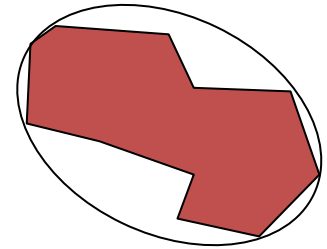
círculo envolvente mínimo



polígono envolvente mínimo com *6* vértices



casco convexo



elipse envolvente mínima

Estrutura da Árvore R

Árvore R

- Antonin Guttman.

- 1984.

Árvore R – Aplicações

- Sistemas de Informações Geográficas (GIS).
- Sistemas CAD.
- Arquiteturas VLSI.
- Sistemas P2P, Bioinformática, Data Streams.

Árvore R

- Dinâmica
  - Permite novas inserções e remoções.
- Hierárquica
  - Nós folhas e nós índices.
- Armazenamento Secundário
  - Nós são páginas de disco de tamanho fixo.
- Construção Bottom-Up
  - Todos os objetos são inseridos nas folhas.
- Balanceada
  - Folhas no mesmo nível.

Estrutura

- Árvore R de ordem (m, M).
- Número máximo de entradas por nó: M
- Número mínimo de entradas por nó: $m \leq \left\lfloor \frac{M}{2} \right\rfloor$
- Altura máxima da árvore: $h_{max} = \lceil \log_m N \rceil - 1$
  - N: número de objetos inseridos.

Estrutura

- O número mínimo de entradas permitido na raiz é 2, a menos que a raiz seja uma folha. Nesse caso, ela pode conter apenas uma ou nenhuma entrada.

- Todas as folhas estão no mesmo nível.

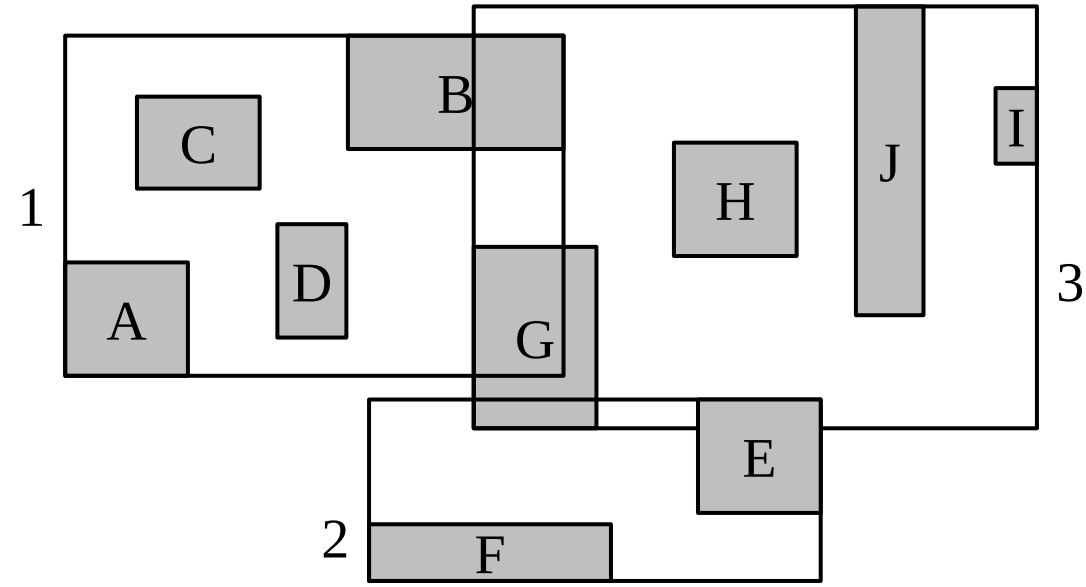
Estrutura dos nós

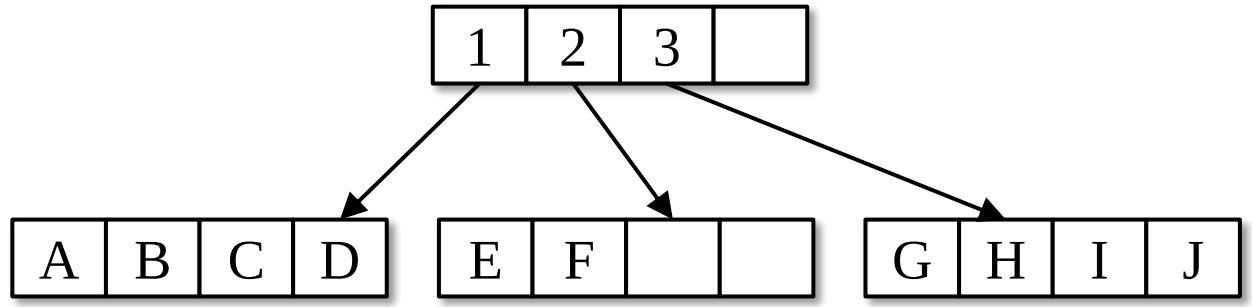
- Folha: <*mbr*, *oid*>
  - *mbr*: retângulo *n*-dimensional que delimita o objeto indexado.
  - *oid*: valor de *identificador de objeto*.

- Índice: <*mbr*, *ptr*>
  - *ptr*: referência ao nó do nível imediatamente inferior.

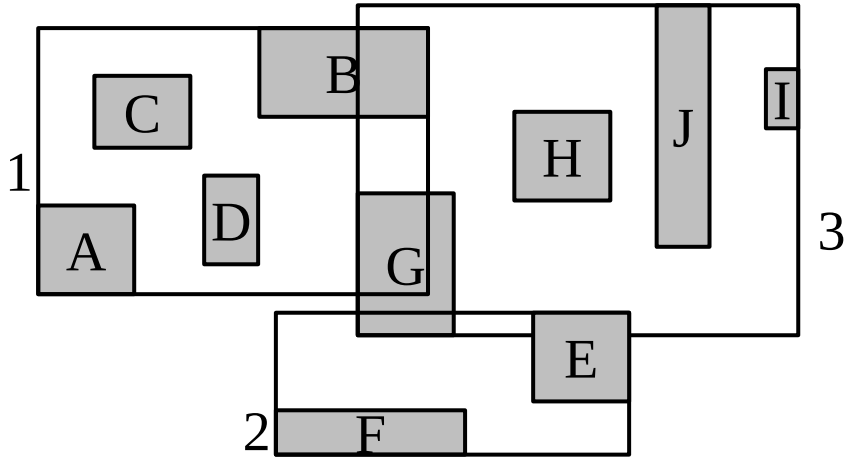
- *mbr*: $<d_0, d_1, d_2, ..., d_{n-1}>$ $d_i = [a, b]$

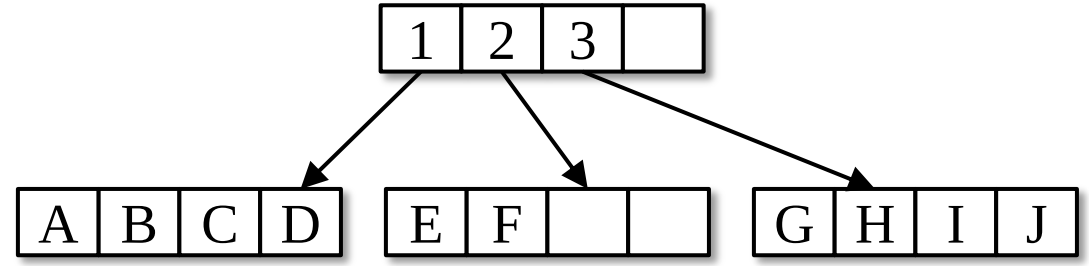
Exemplo: Árvore R(2, 4)





Representação dos nós

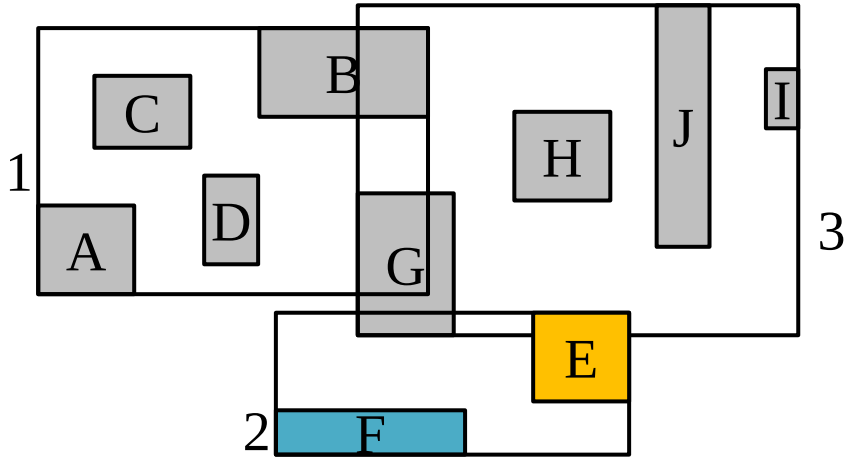


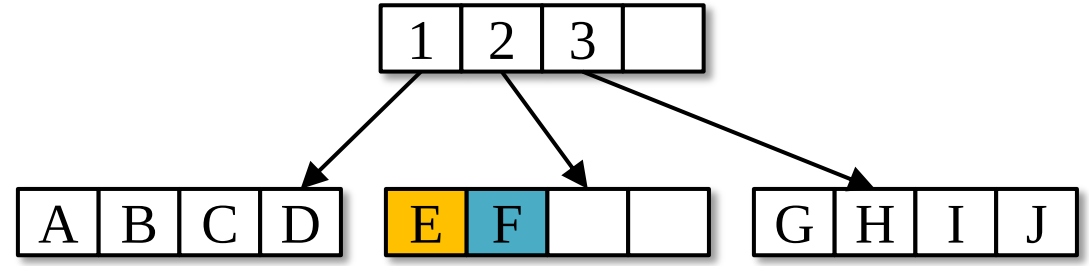


Leaf	α	β	*espaço livre*	28	0	35	3	30	4	36	7

Index	b1	b2	b4	*esp. livre*	32	9	39	49	28	0	36	7	0	15	34	45

Representação dos nós





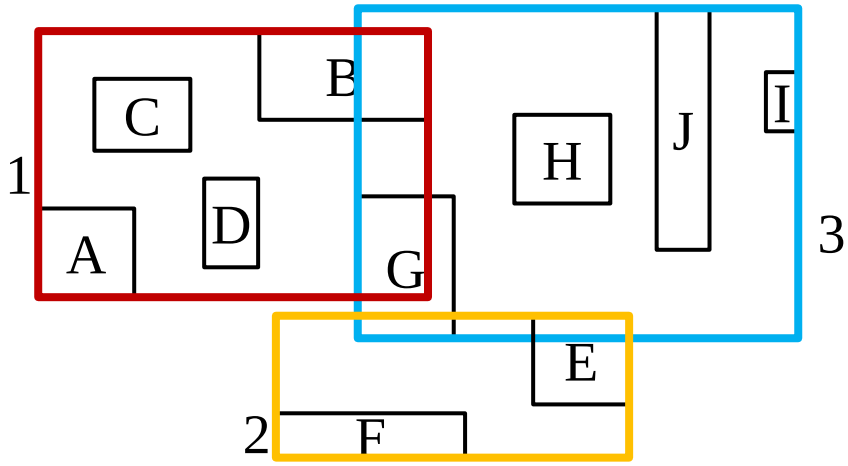
MBR

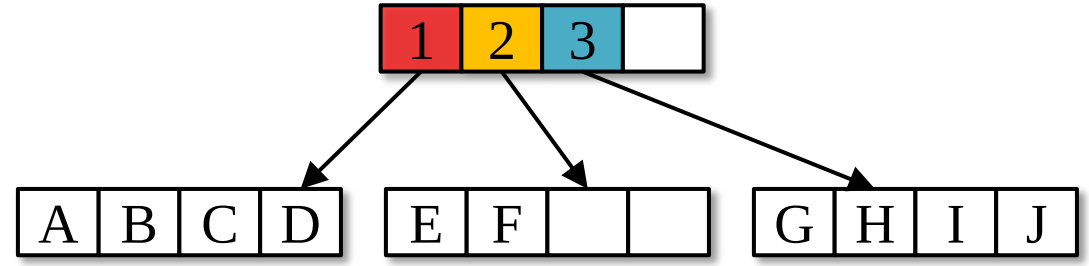
Leaf	α	β	*espaço livre*	28	0	35	3	30	4	36	7

OID

Index	b1	b2	b4	*esp. livre*	32	9	39	49	28	0	36	7	0	15	34	45

Representação dos nós





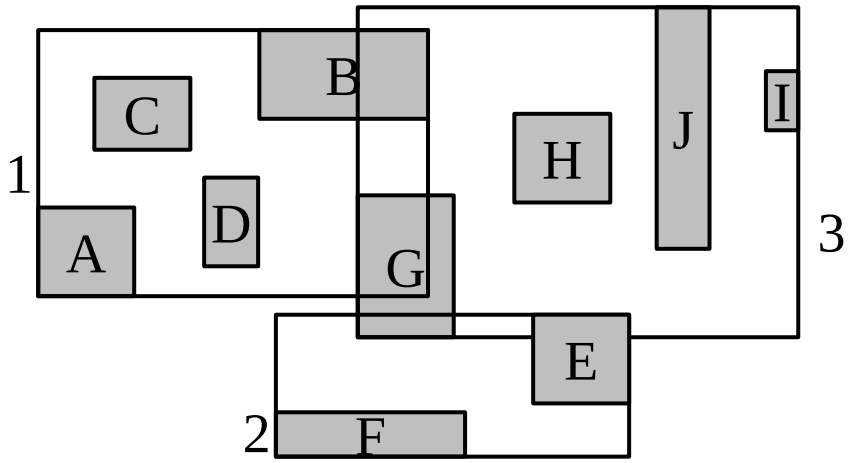
Leaf	α	β	*espaço livre*	28	0	35	3	30	4	36	7

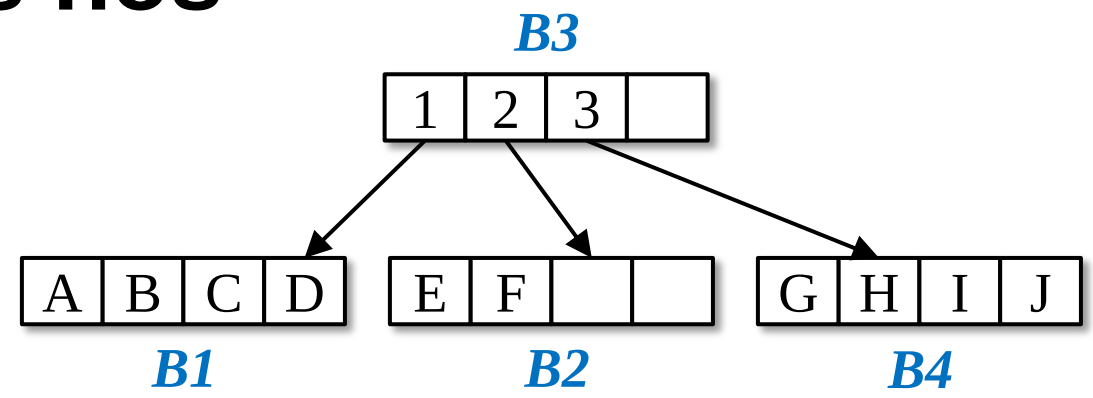
PTR

MBR

Index	b1	b2	b4	*esp. livre*	32	9	39	49	28	0	36	7	0	15	34	45

Representação dos nós





B0	*H*	b3	512	

B1	*L*	α	β	γ	δ	D	C	B	A

B2	*L*	ε	ζ		F	E

B3	*I*	b1	b2	b4		3	2	1

B4	*L*	η	θ	μ	ω	J	I	H	G

Árvore R

Consultas

Consultas espaciais

Point Query

Window Query

Region Query

Adjacency Query

Operadores de Consulta

- Topológicos: encontra todos os objetos que interceptam um dado objeto.

- Direcionais: encontra todos os objetos que, por exemplo, estão ao norte de um dado objeto.

- Distância: encontra todos os objetos que estão a menos que uma distância *d* de um dado objeto (range query) ou os *k* objetos mais próximos de um dado objeto (*k*-nearest-neighbors query).

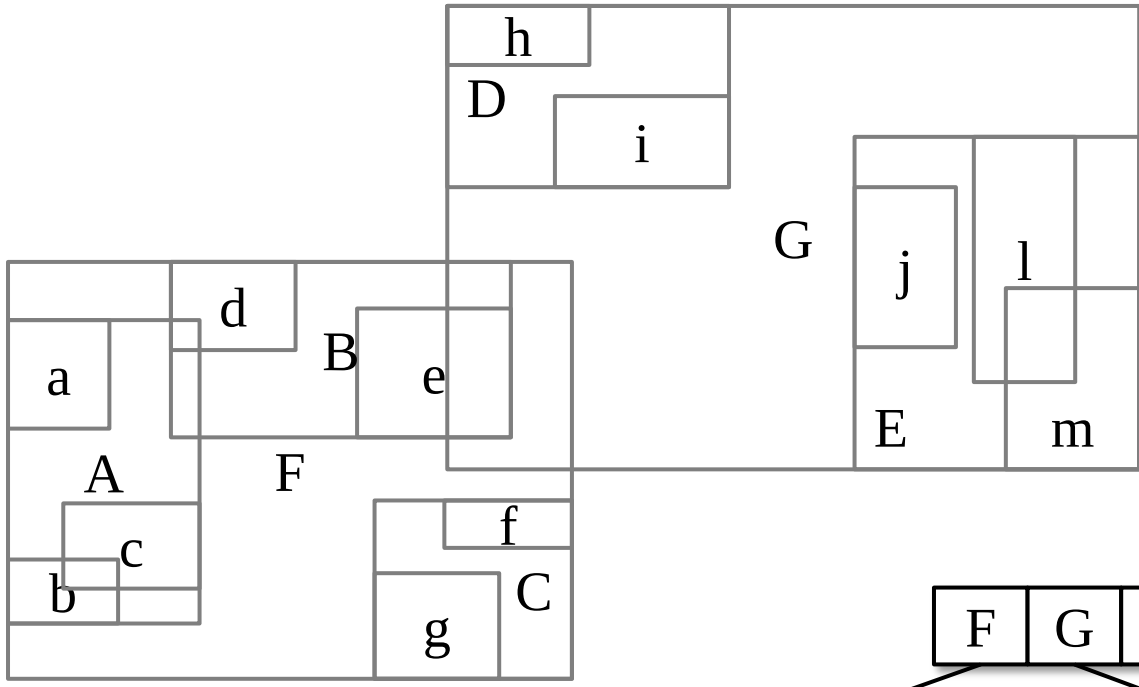
Consulta – Algoritmo (Range Query)

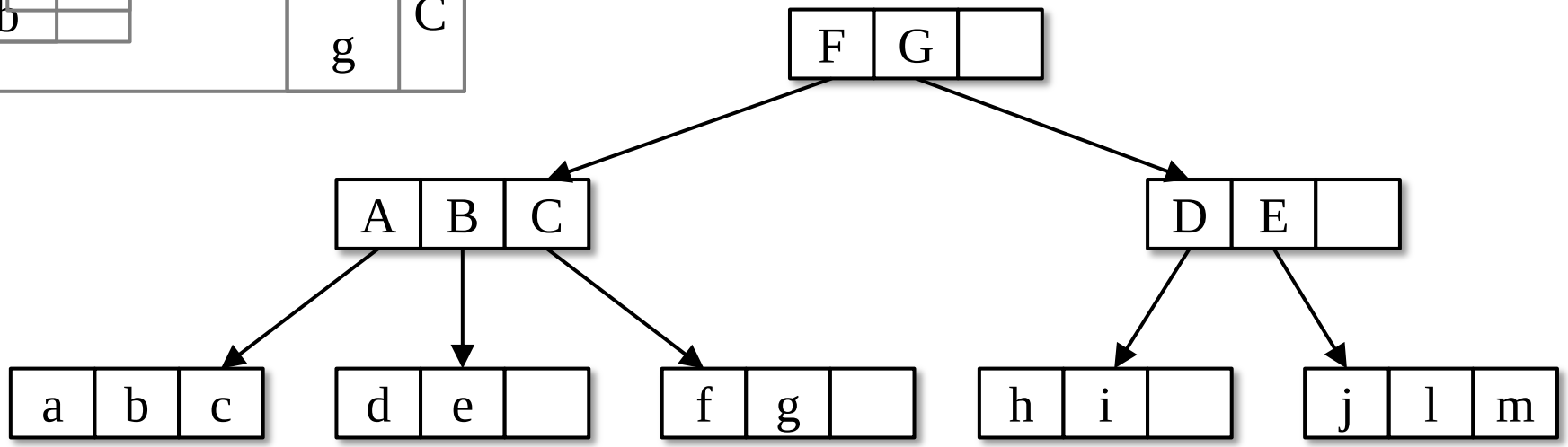
- *Encontra todos os objetos interceptados pelo retângulo de busca Q. Devolve um conjunto S de objetos candidatos.*

- Para todas as entradas de um nó índice, a partir da raiz:
  Verifica se existe sobreposição.
  - Se sim, verifica a respectiva sub-árvore.

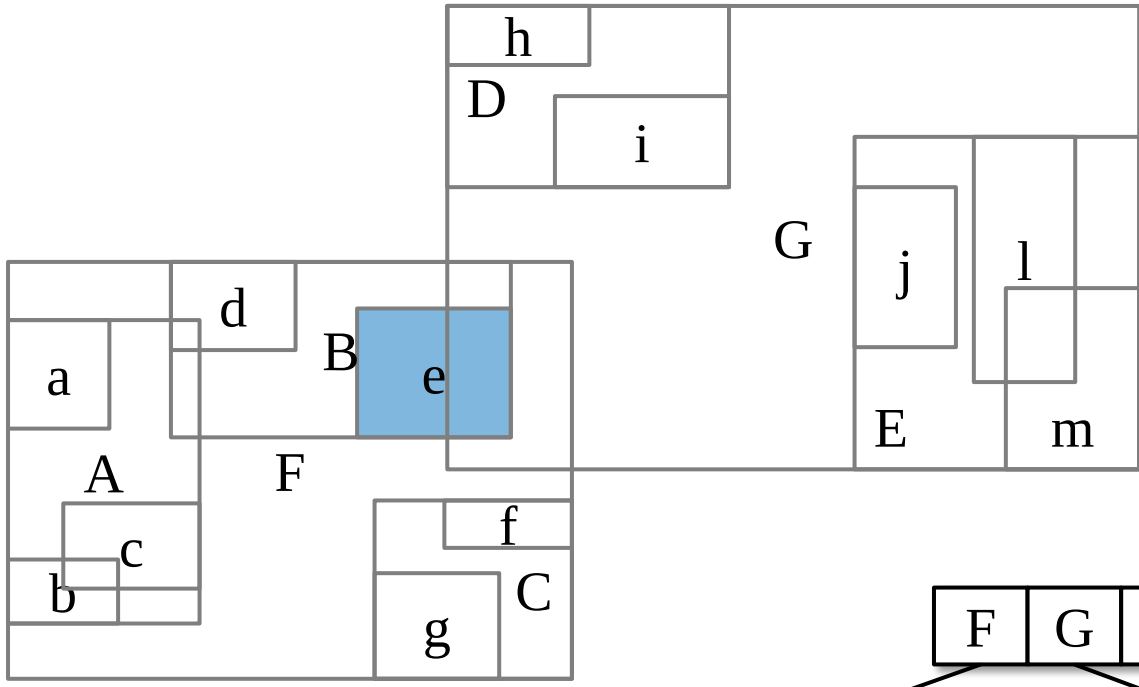
- Se é um nó folha:
  Verifica todas as entradas que interceptam Q.
  - Adiciona no conjunto resposta S.

Consulta – Exemplo

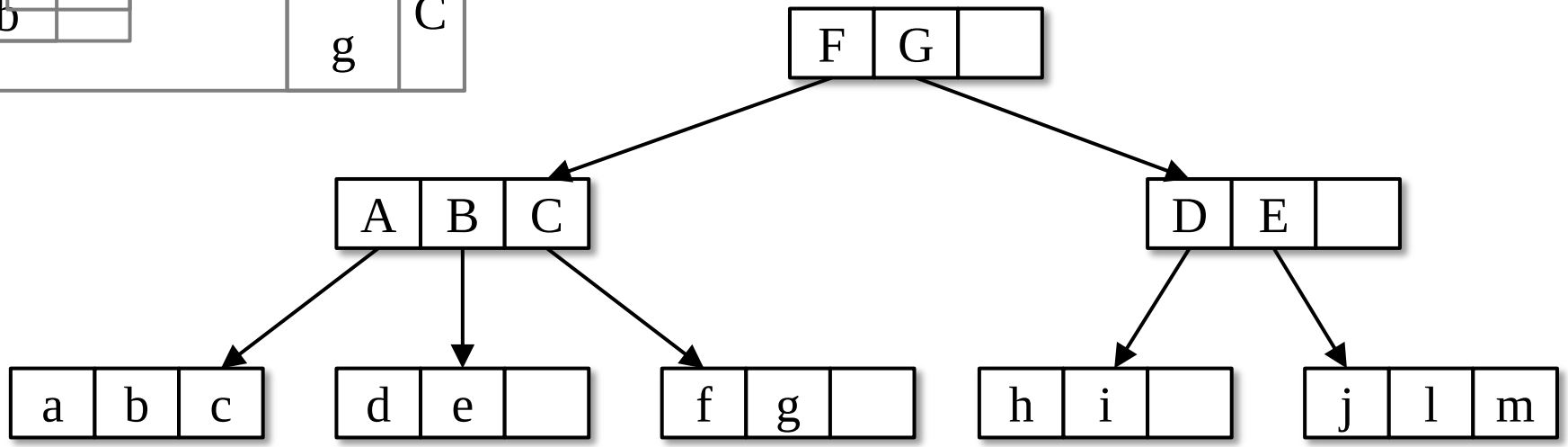




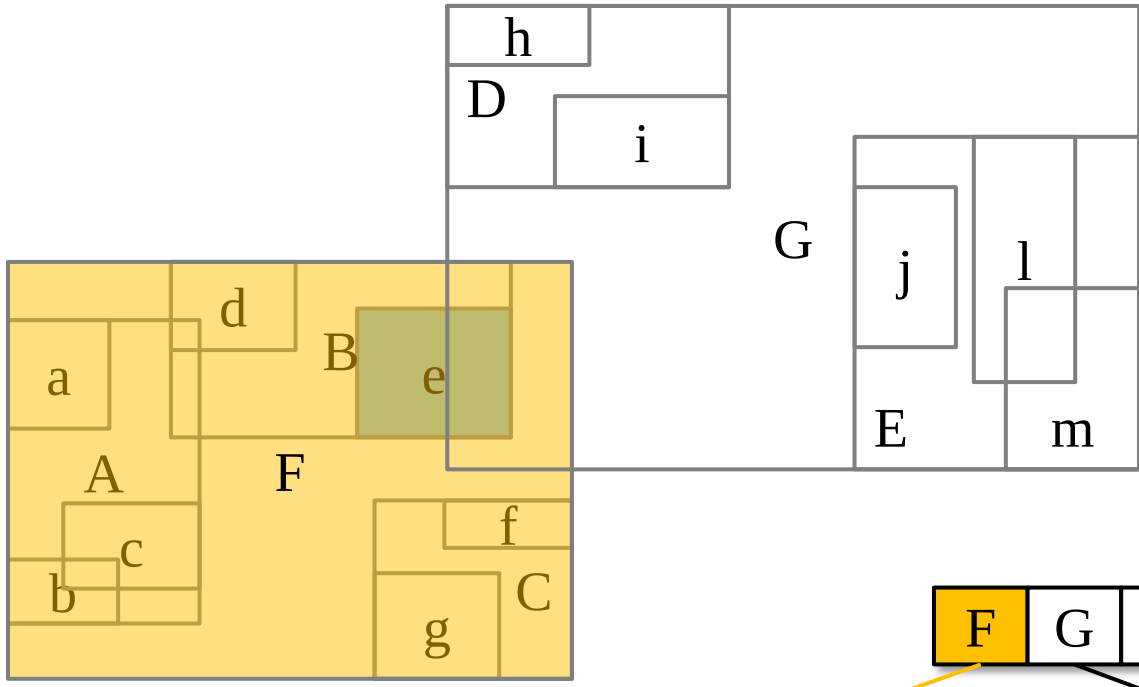
Consulta – Exemplo 1



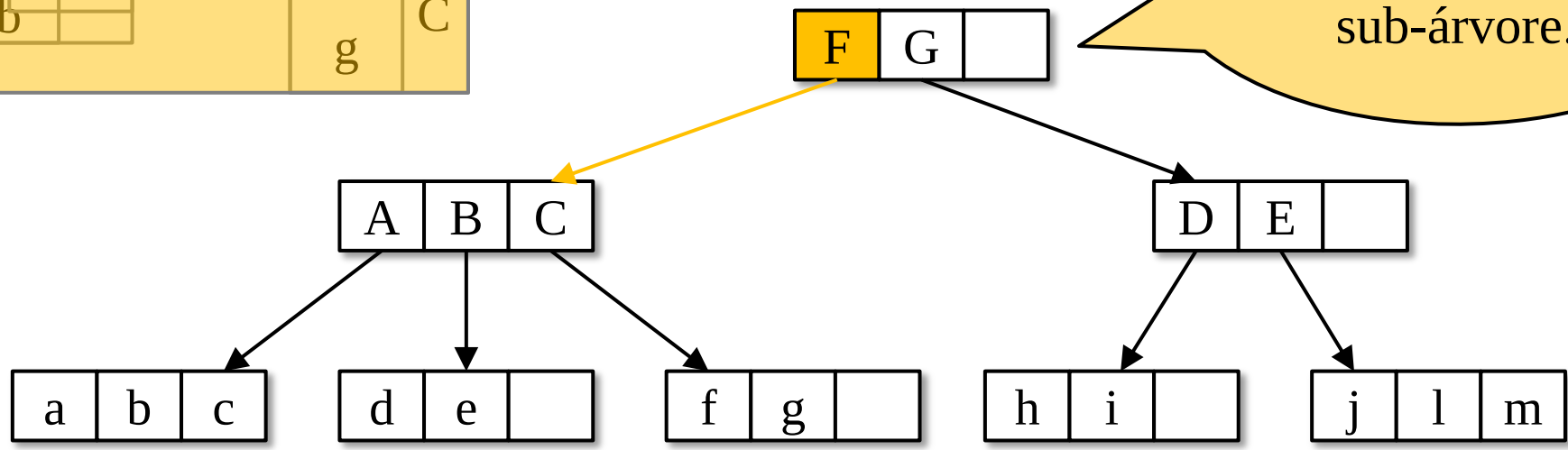
Buscar o objeto "e"



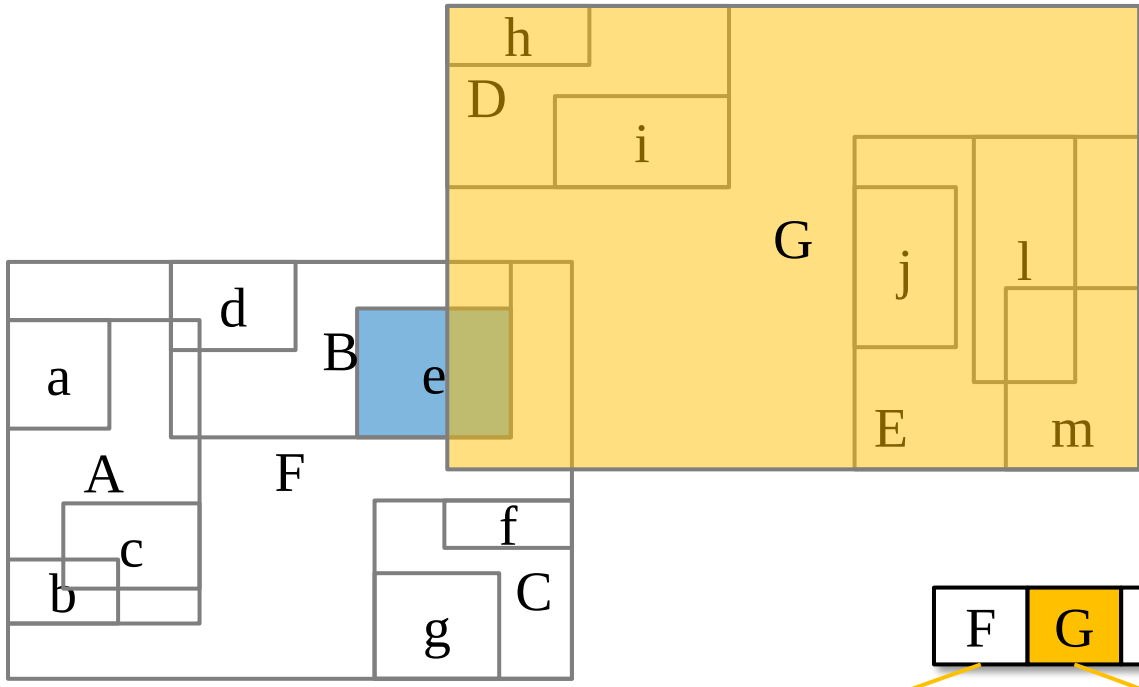
Consulta – Exemplo 1



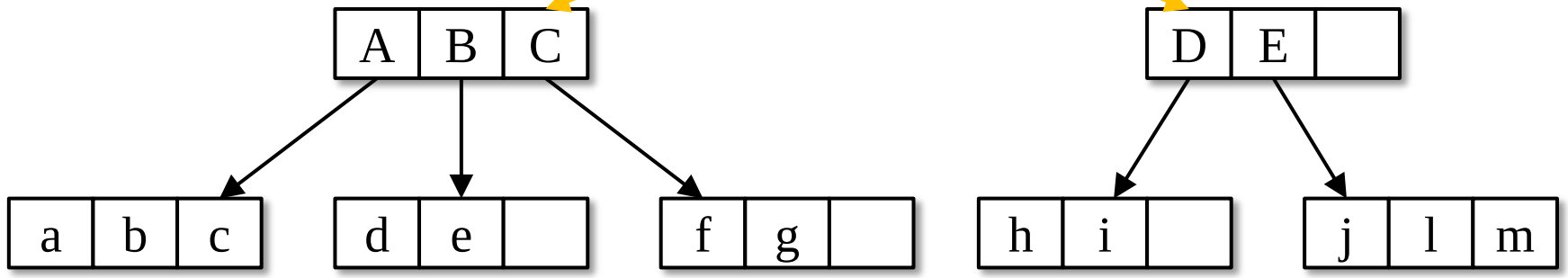
"F" intercepta "e". Então verifica sub-árvore.



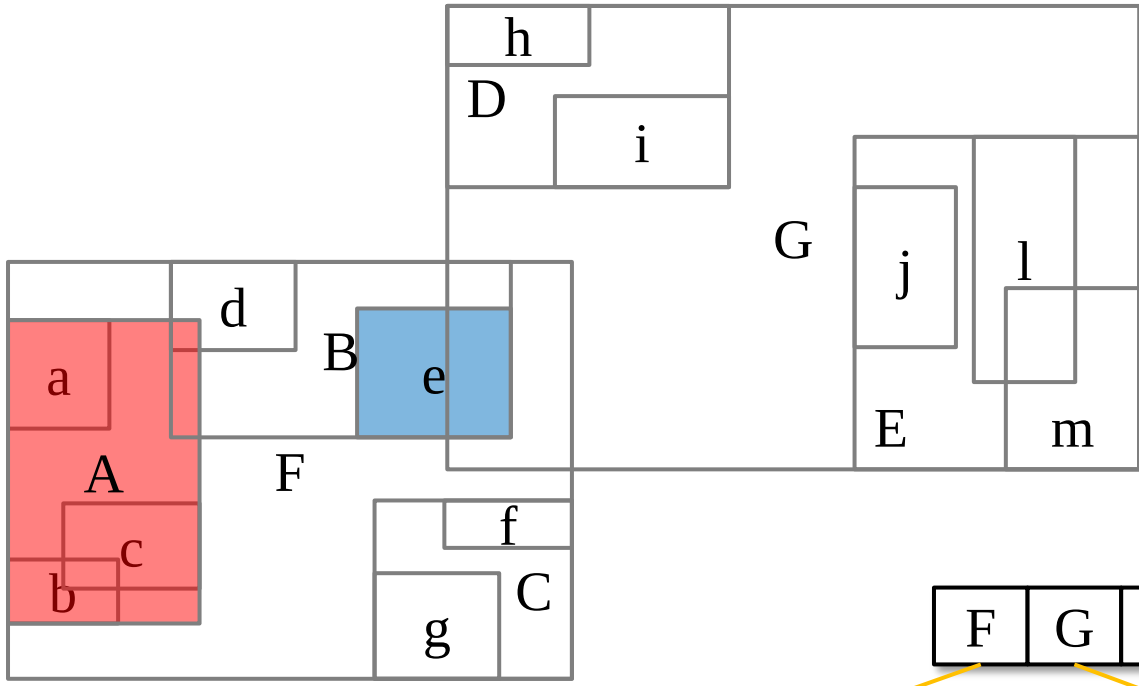
Consulta – Exemplo 1



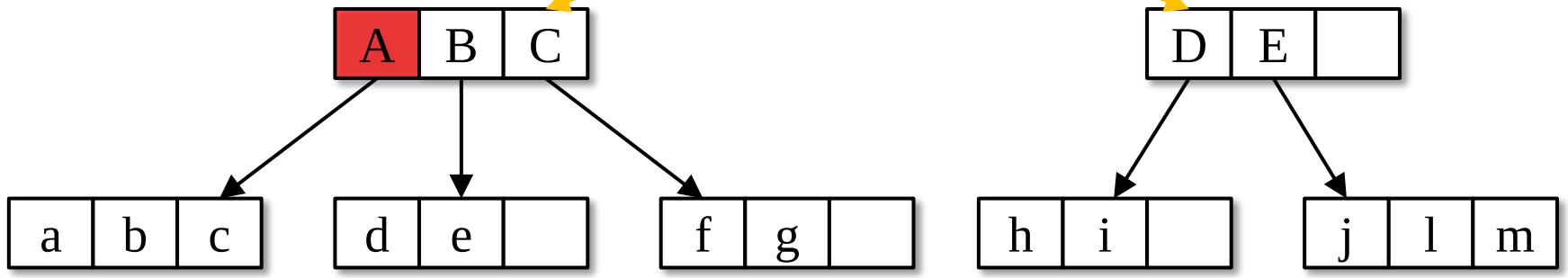
"G" intercepta "e". Então verifica sub-árvore.



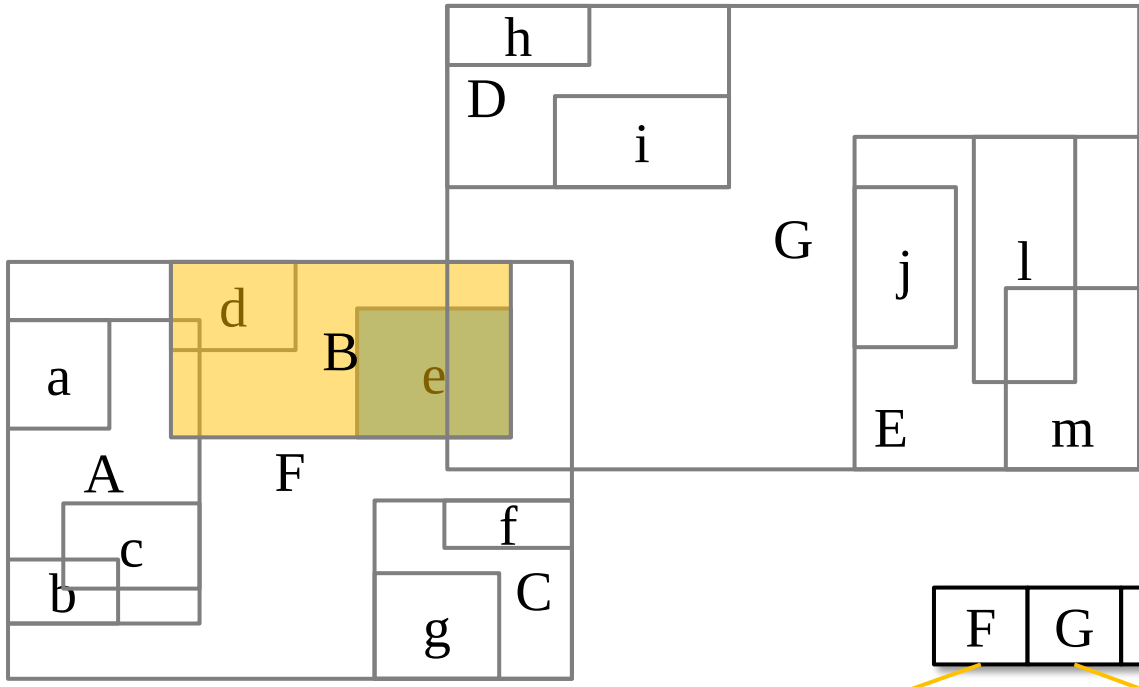
Consulta – Exemplo 1



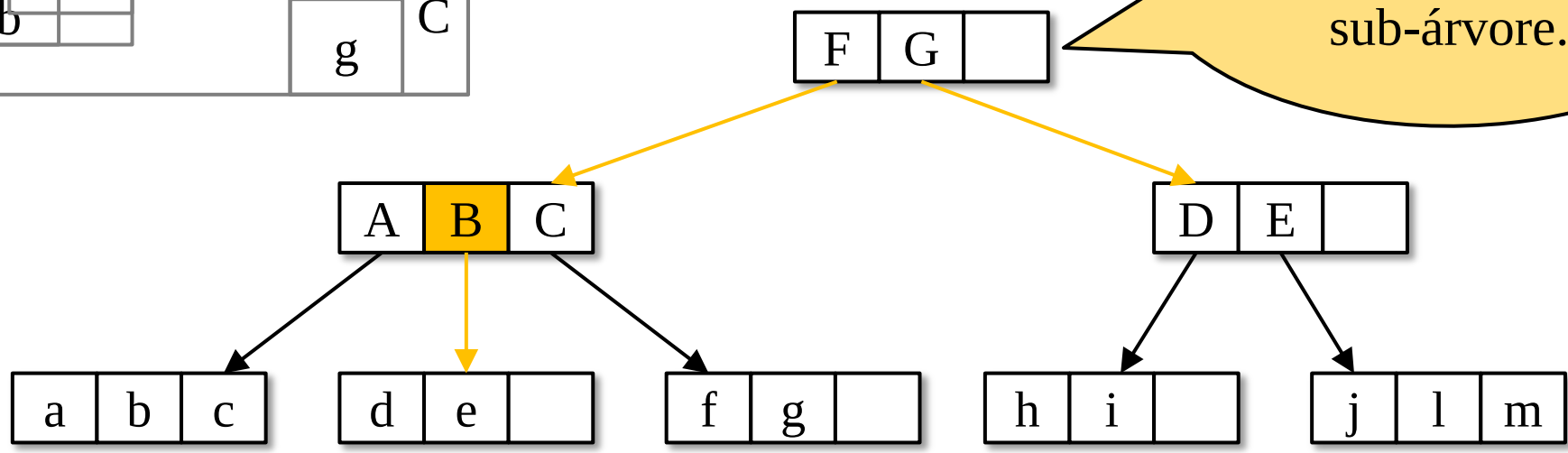
"A" não intercepta "e". Então não verifica sub-árvore.



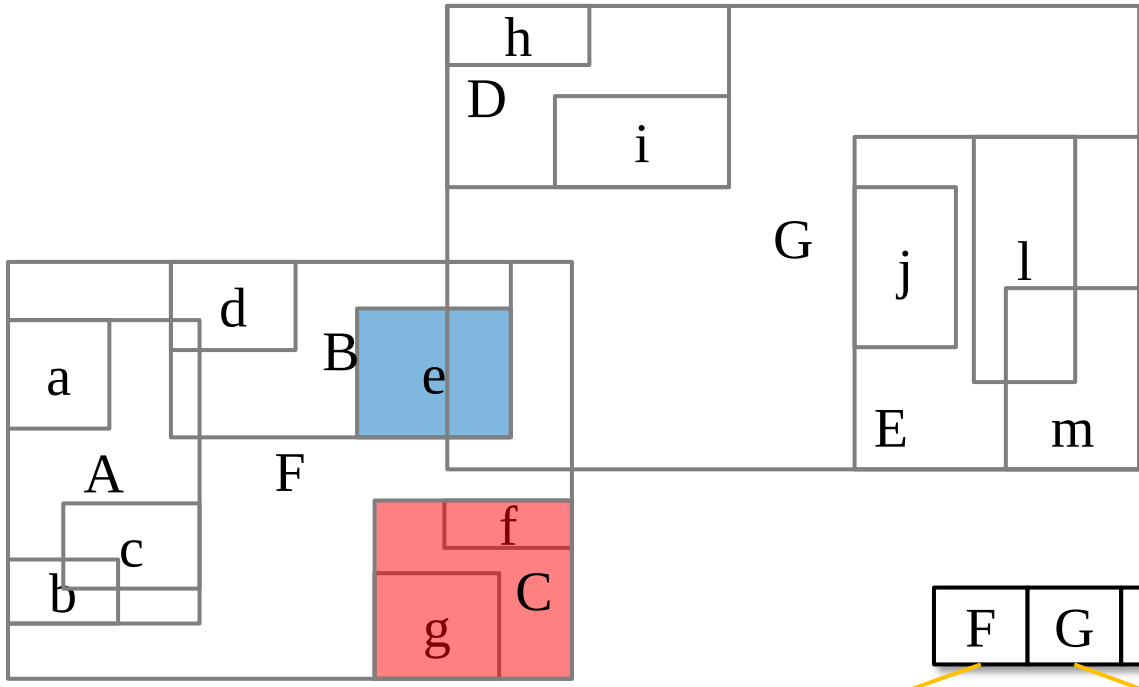
Consulta – Exemplo 1



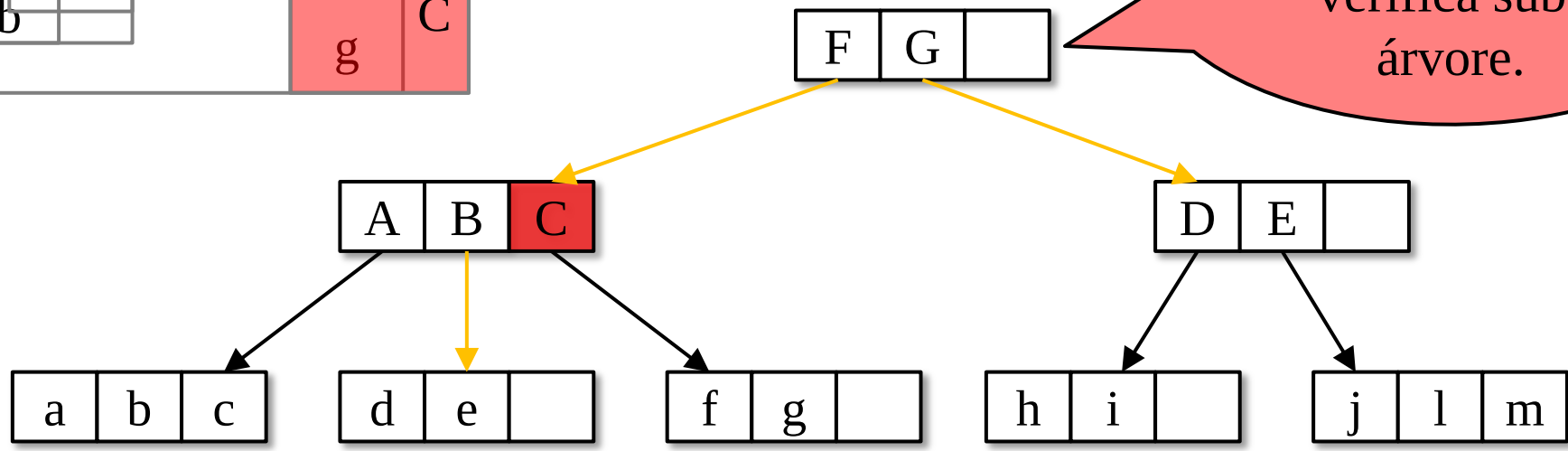
"B" intercepta "e". Então verifica sub-árvore.



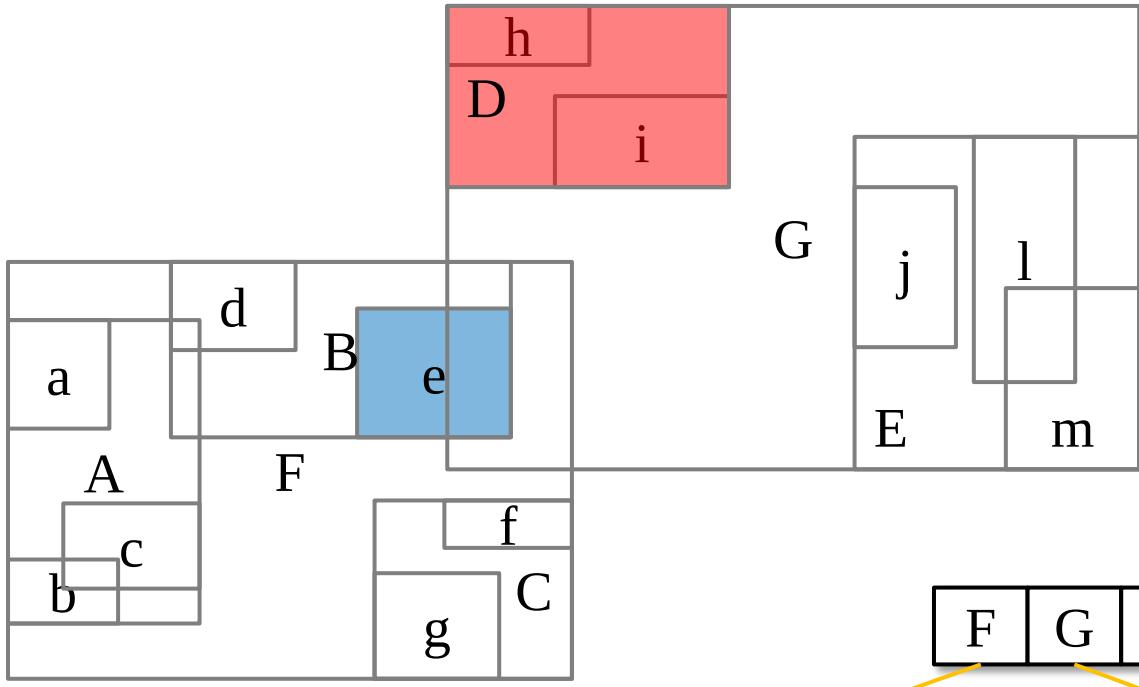
Consulta – Exemplo 1



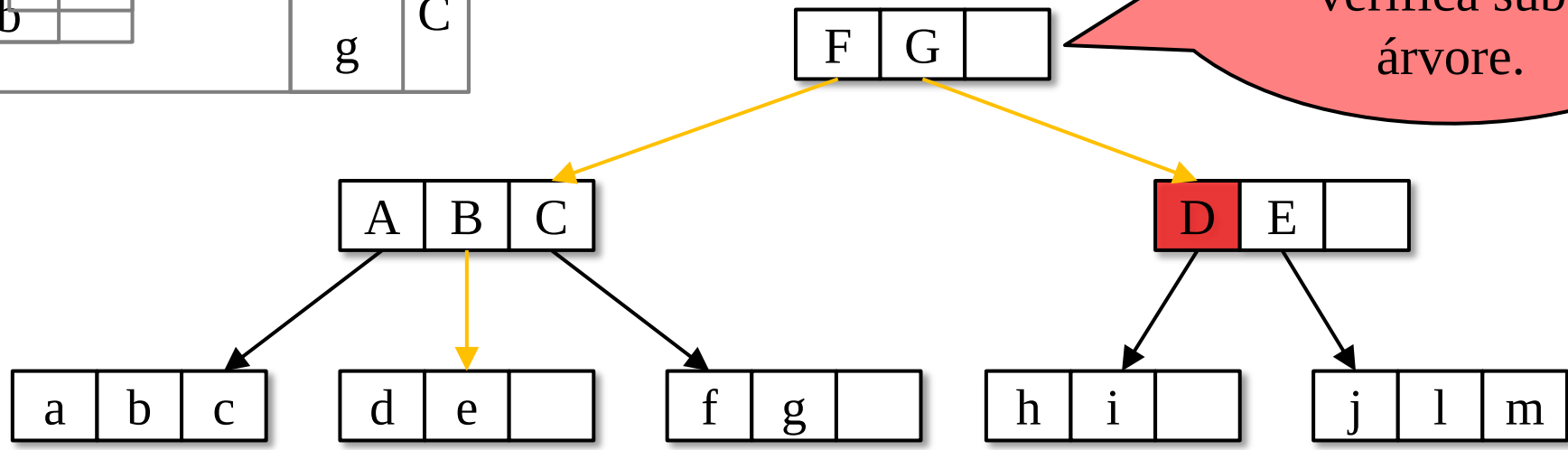
"C" não intercepta "e". Então não verifica sub-árvore.



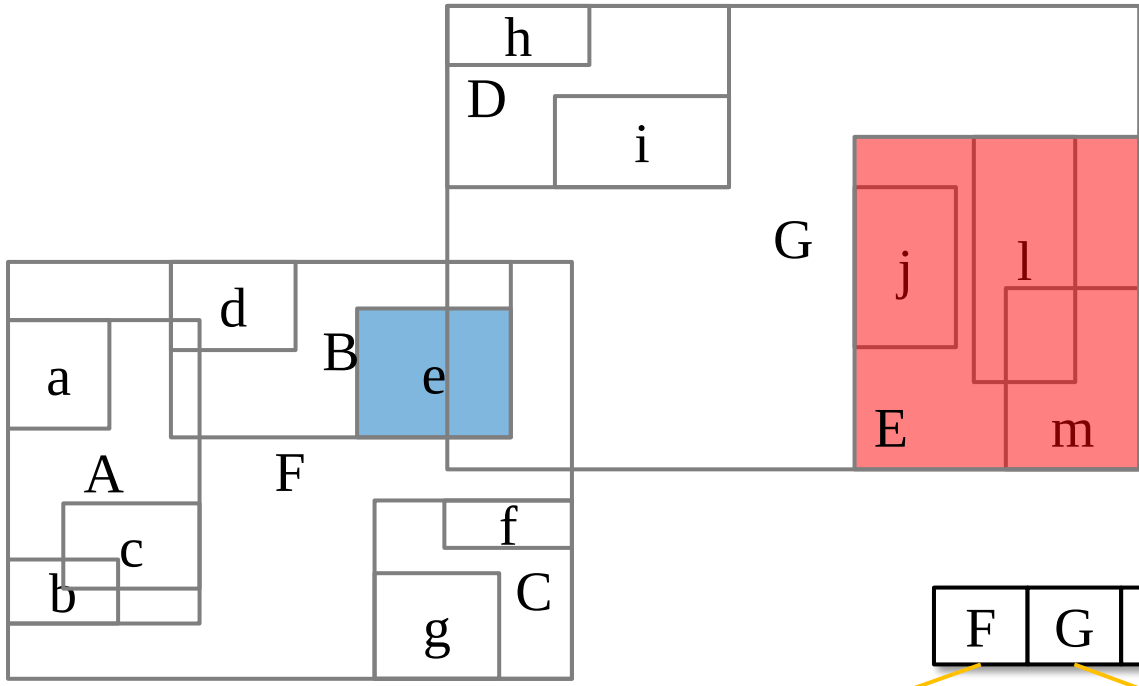
Consulta – Exemplo 1



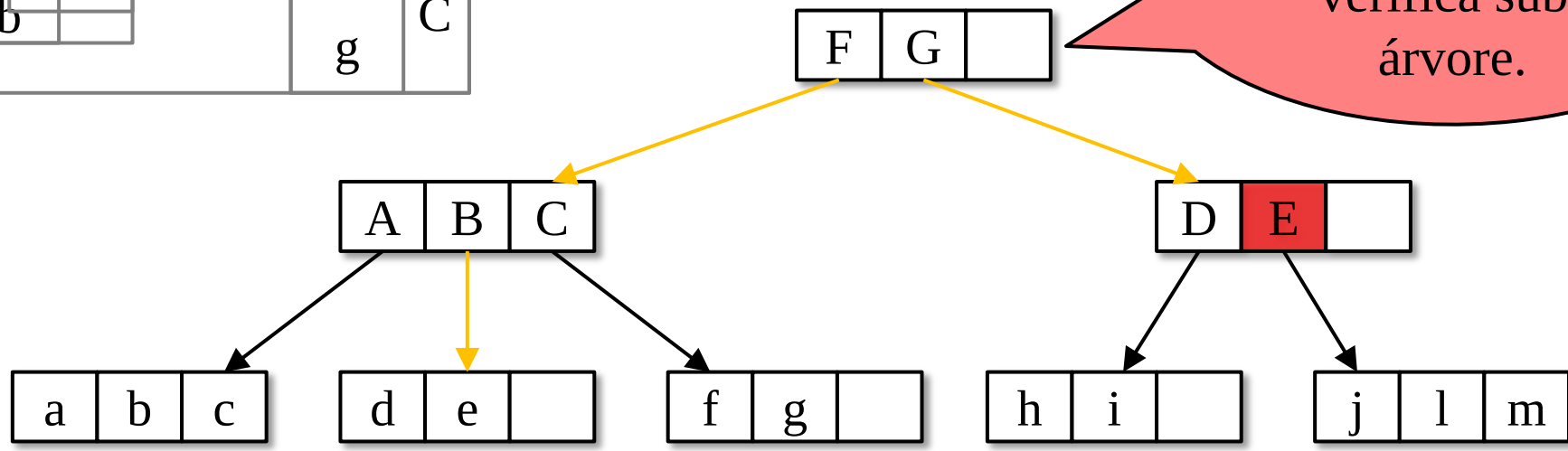
"D" não intercepta "e". Então não verifica sub-árvore.



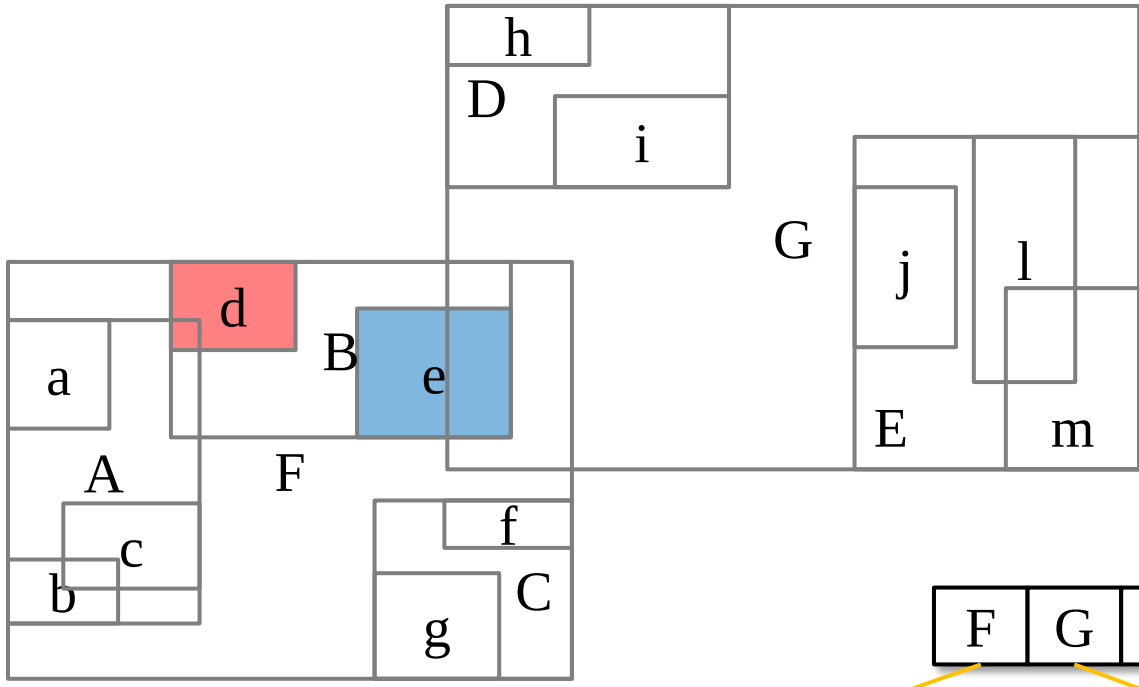
Consulta – Exemplo 1

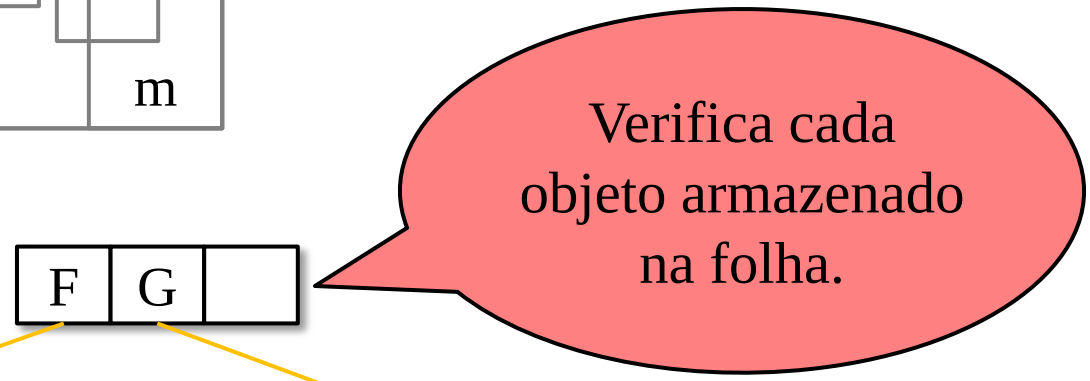


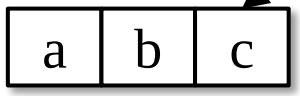
"E" não intercepta "e". Então não verifica sub-árvore.

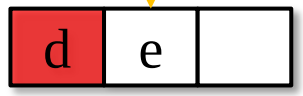


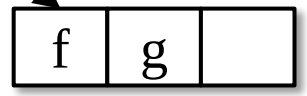
Consulta – Exemplo 1

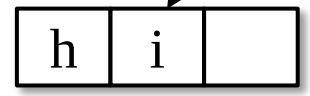


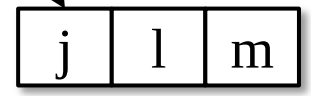




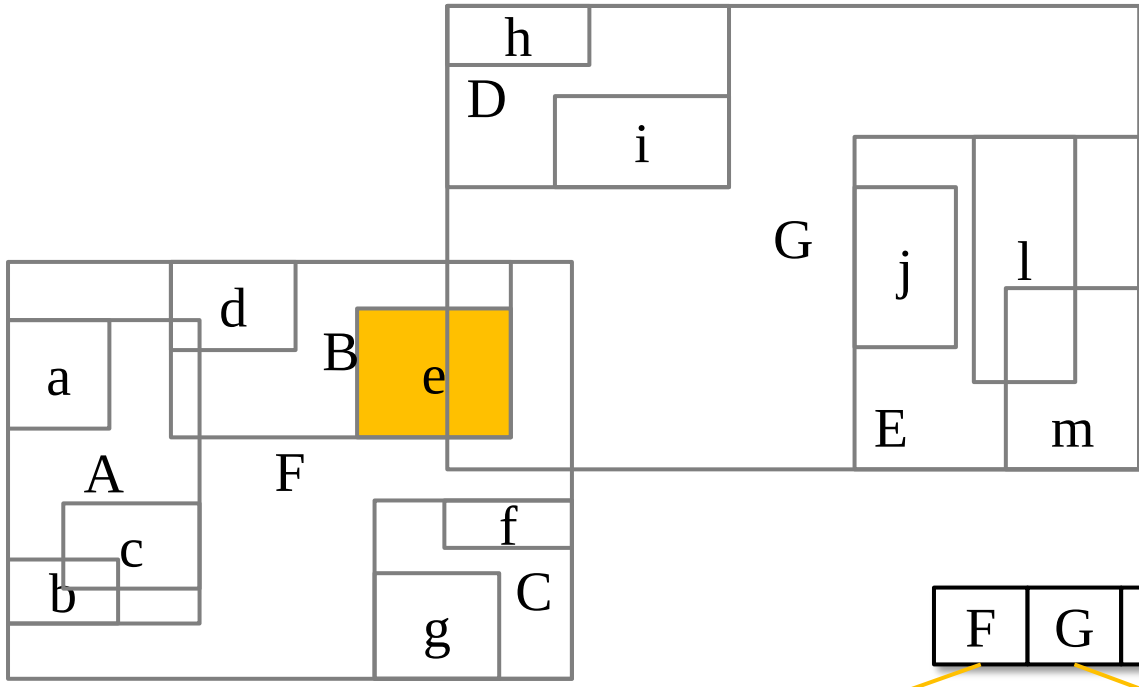




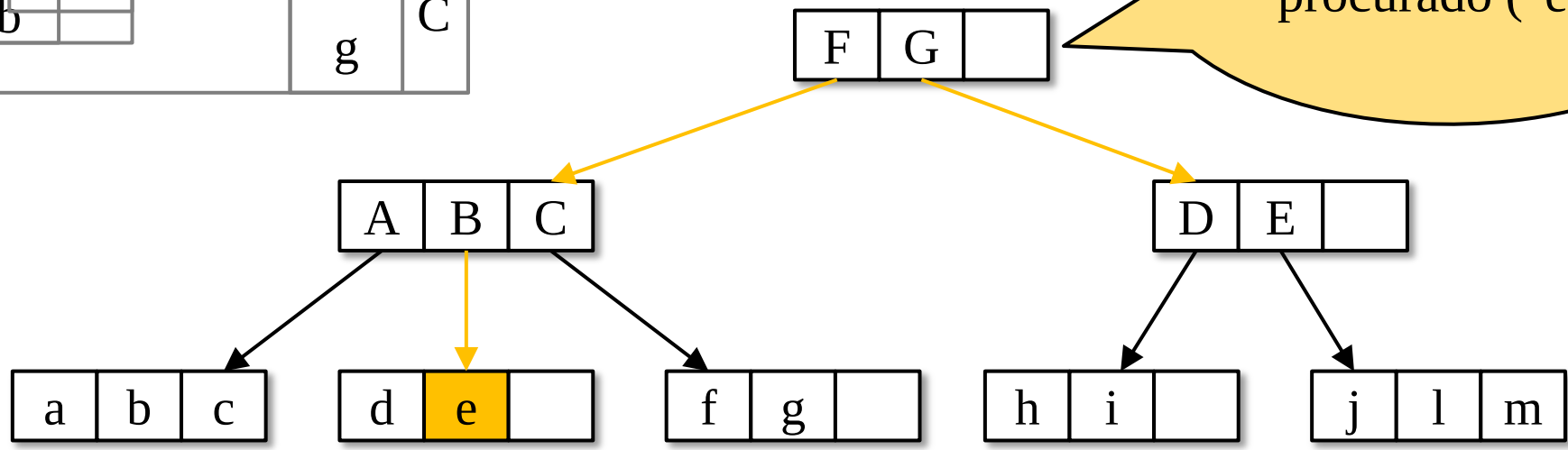




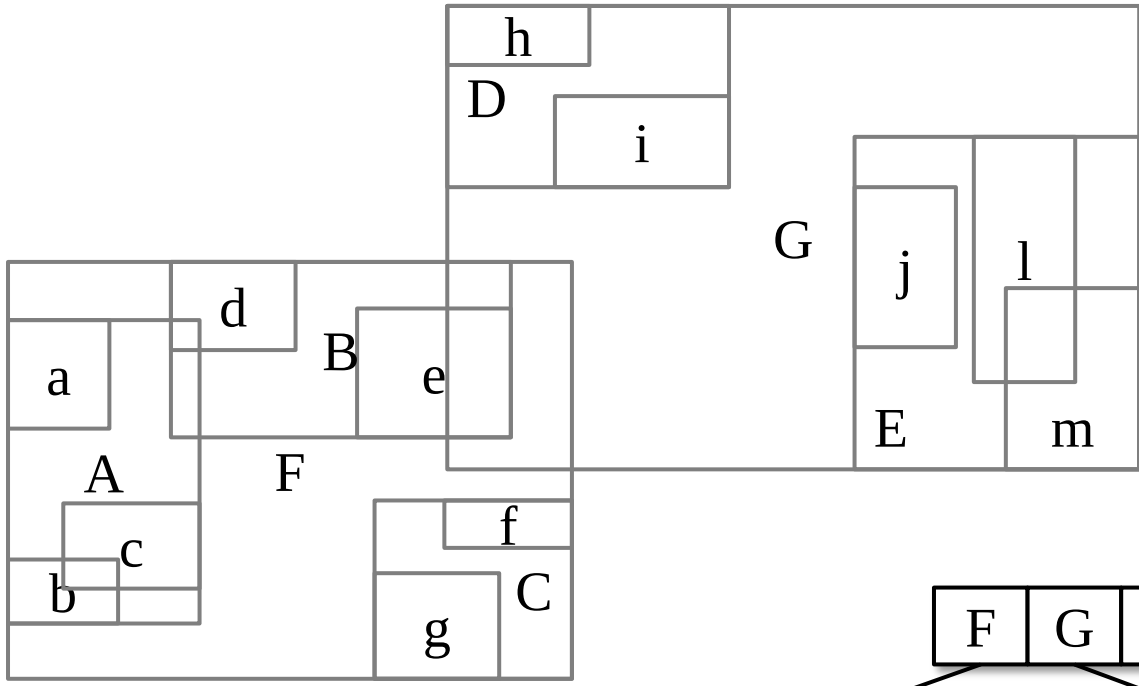
Consulta – Exemplo 1

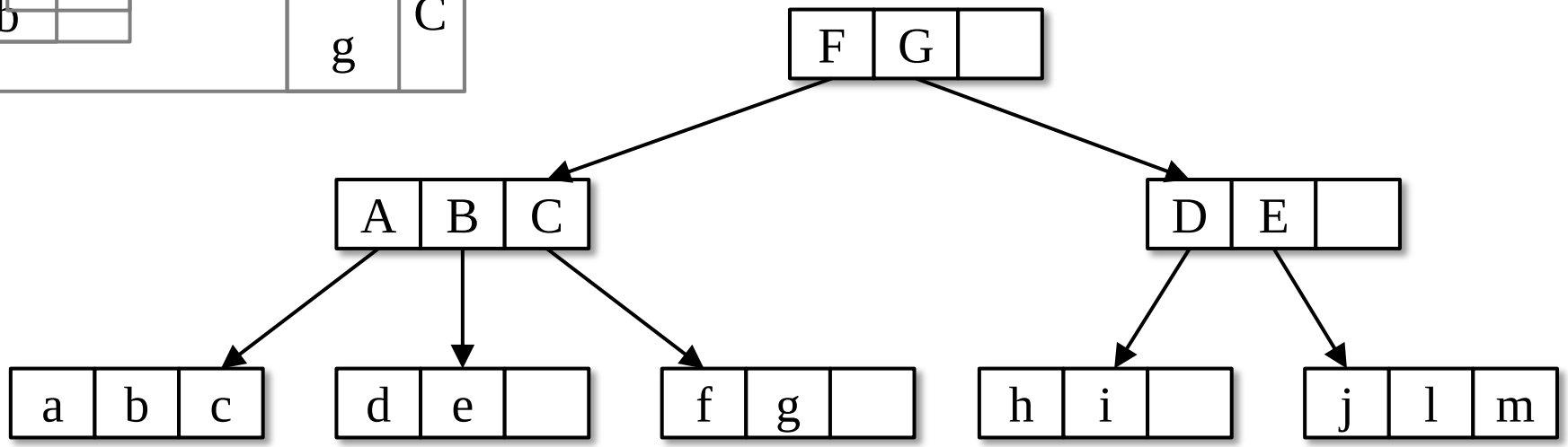


Encontra o objeto procurado ("e").

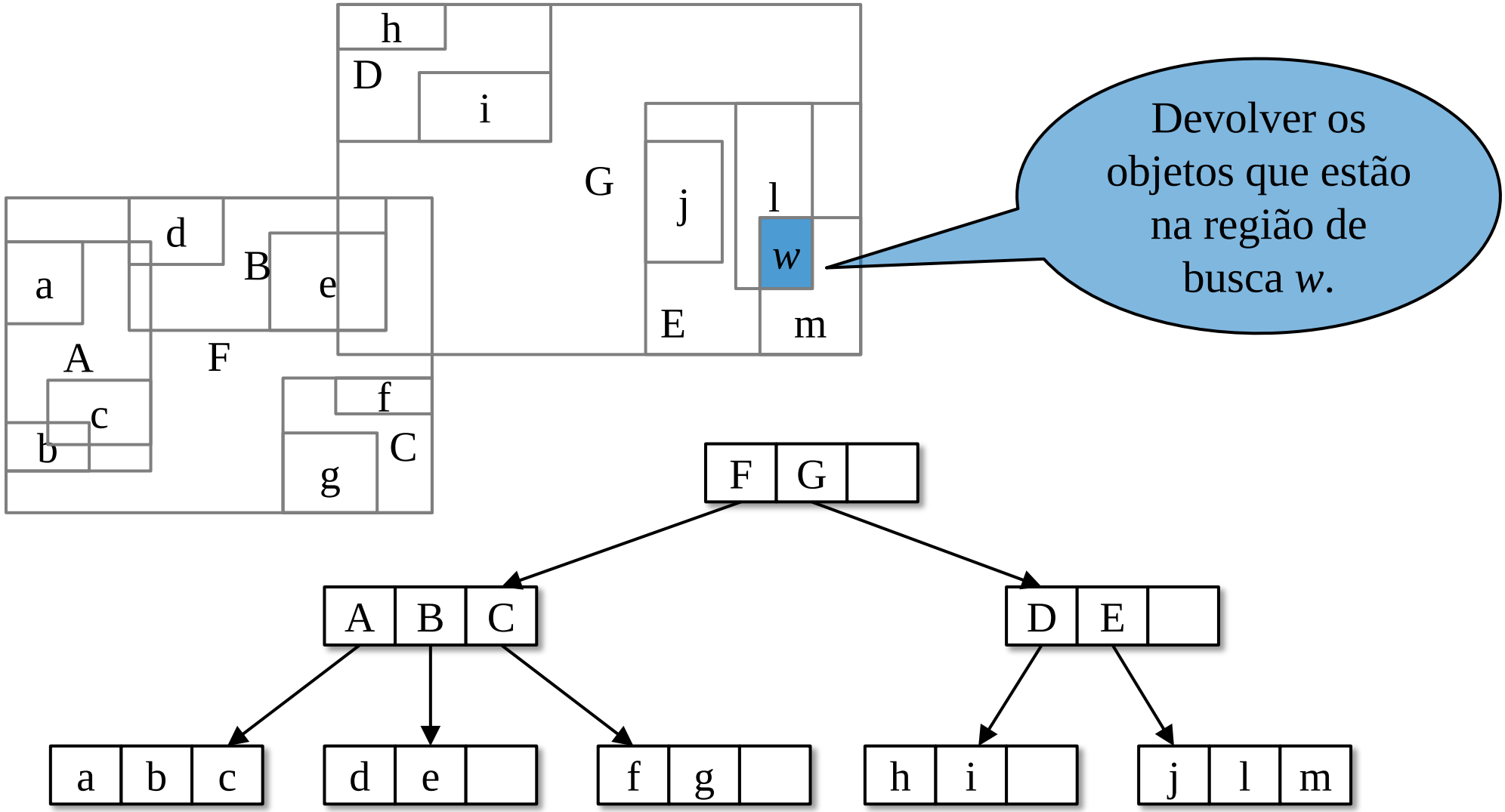


Consulta – Exemplo

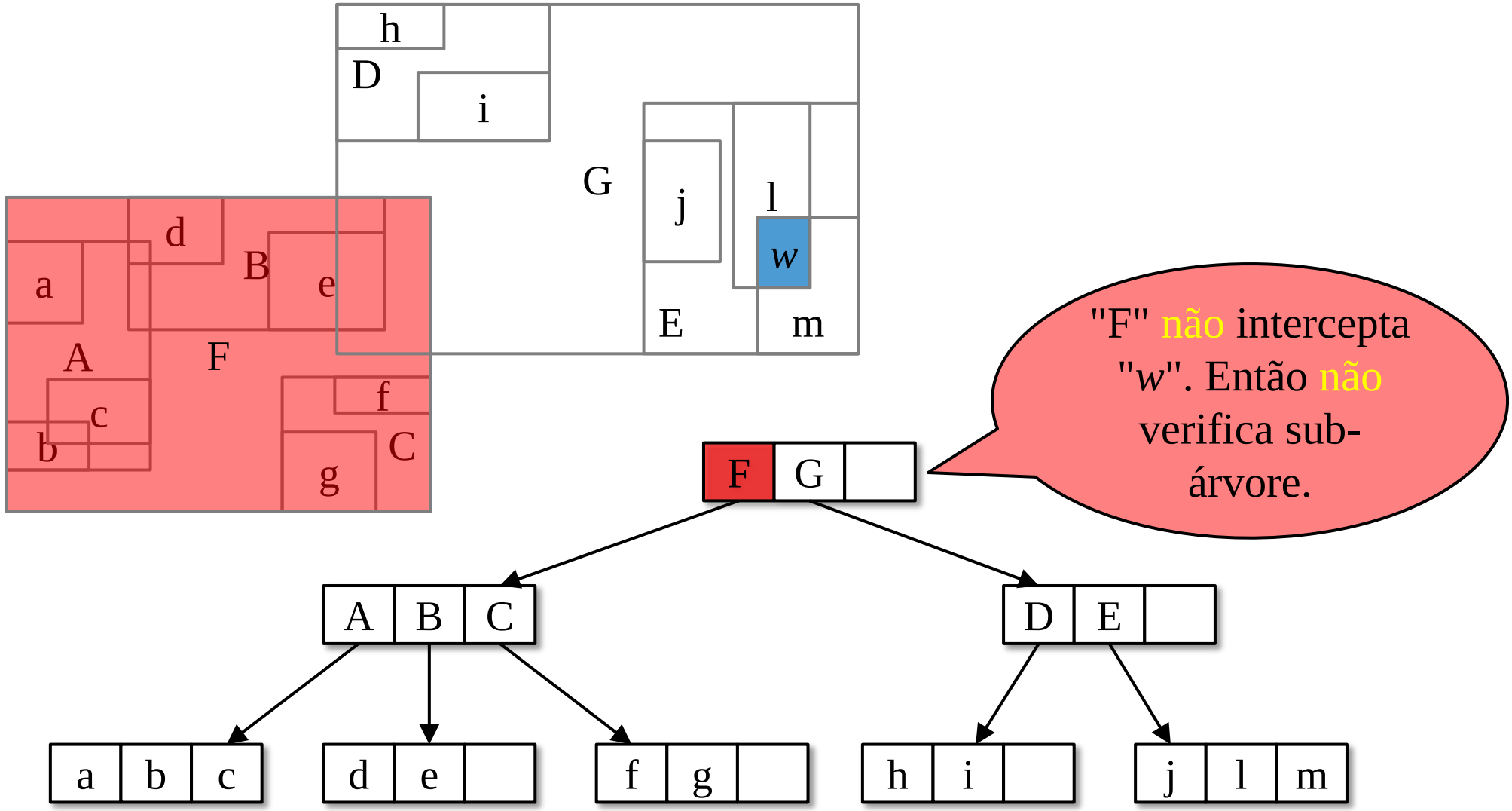




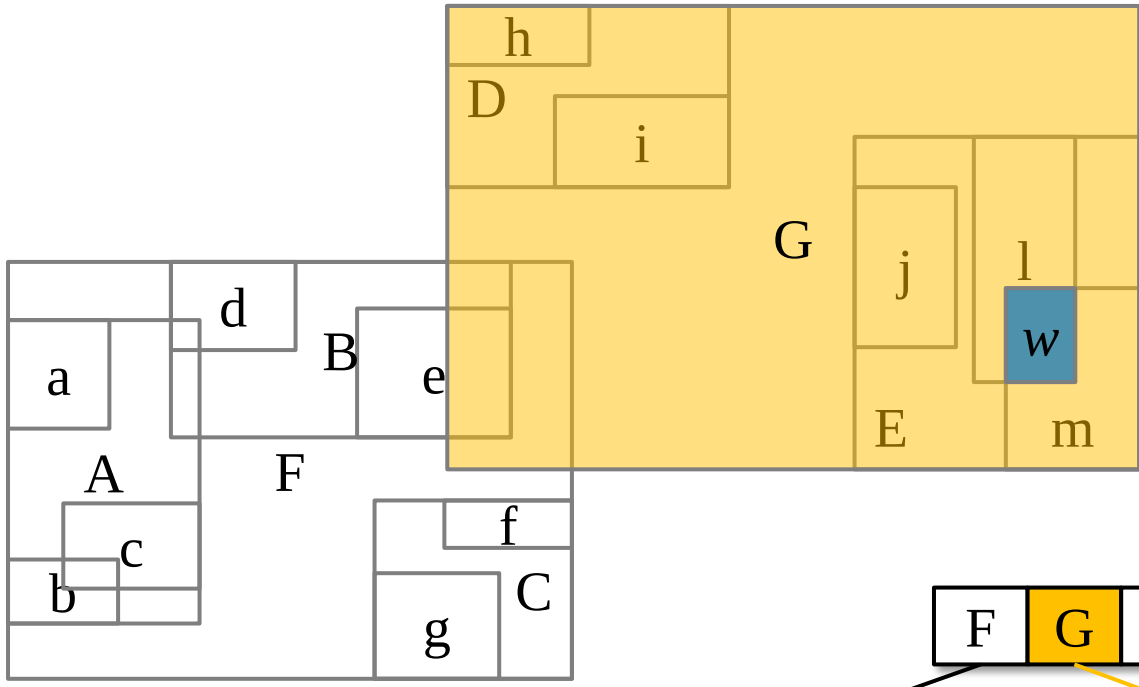
Consulta – Exemplo 2



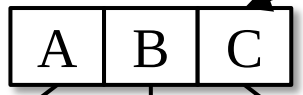
Consulta – Exemplo 2

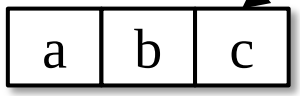


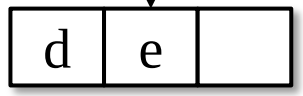
Consulta – Exemplo 2

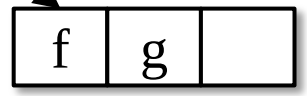


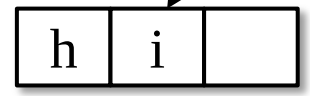
"G" intercepta "*w*". Então verifica sub-árvore.

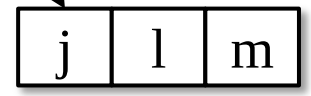




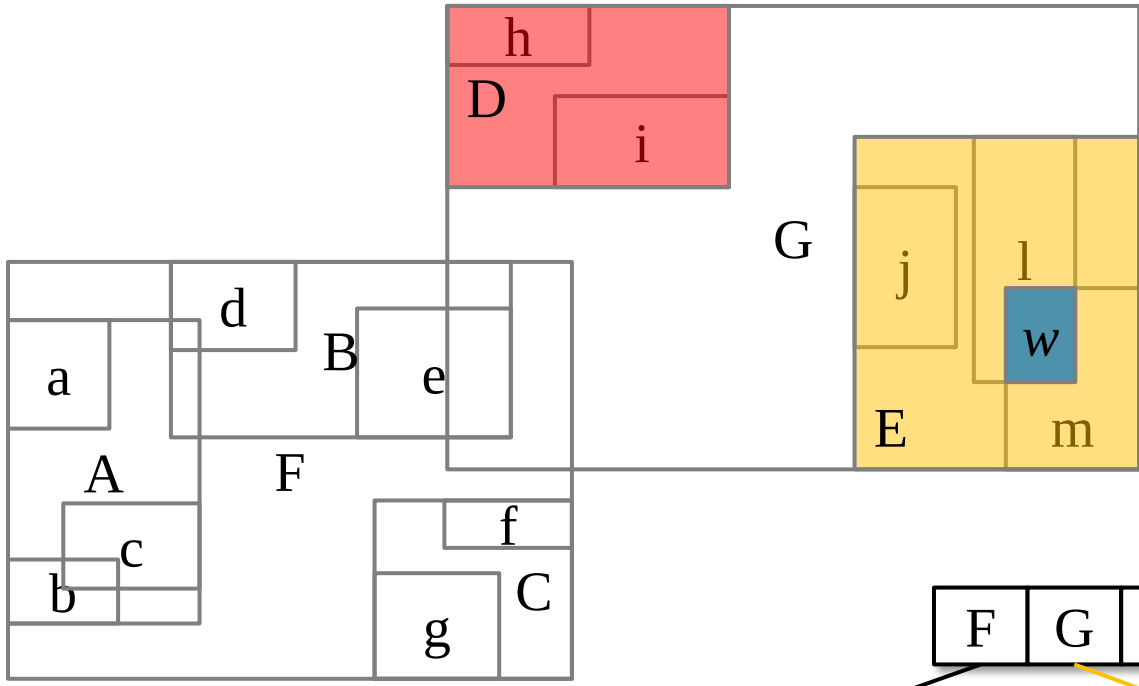




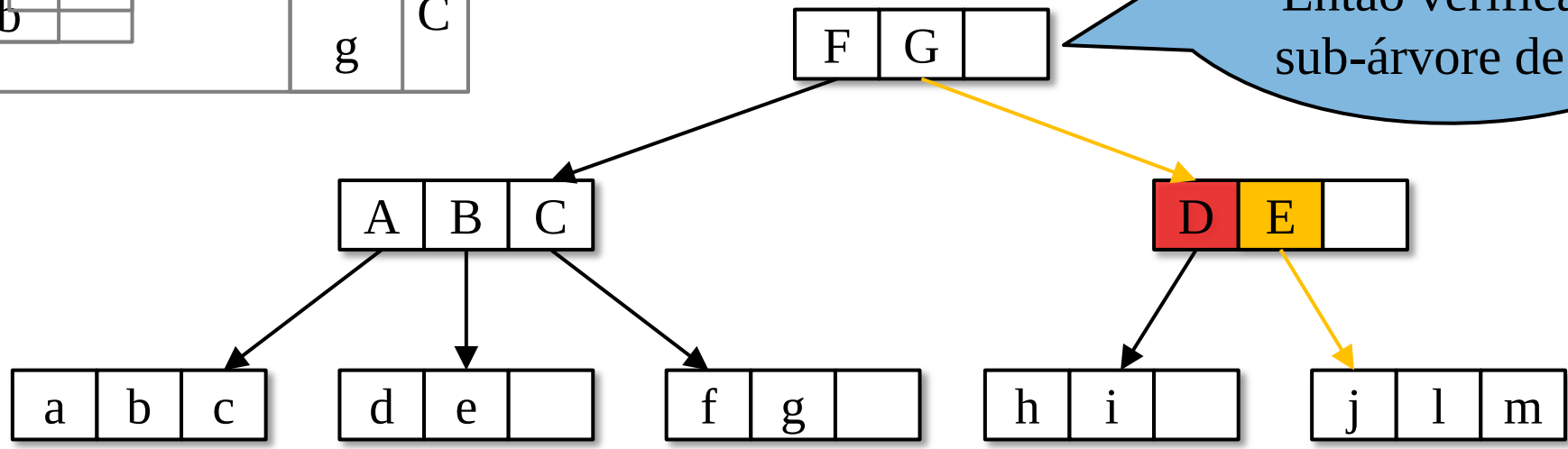




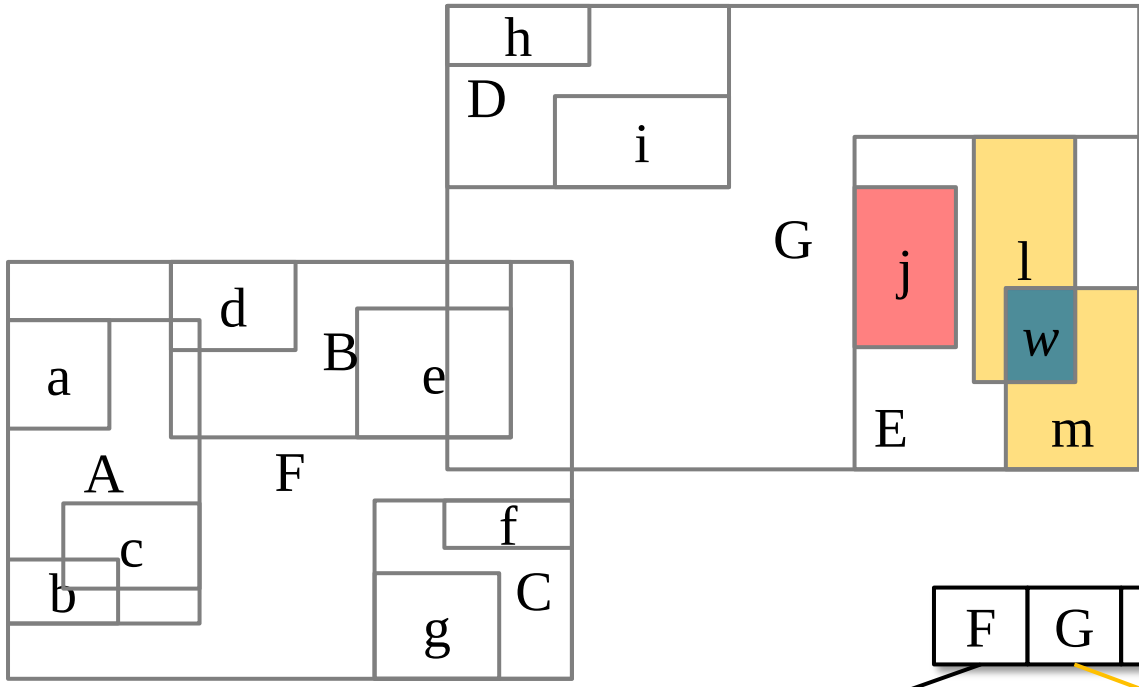
Consulta – Exemplo 2



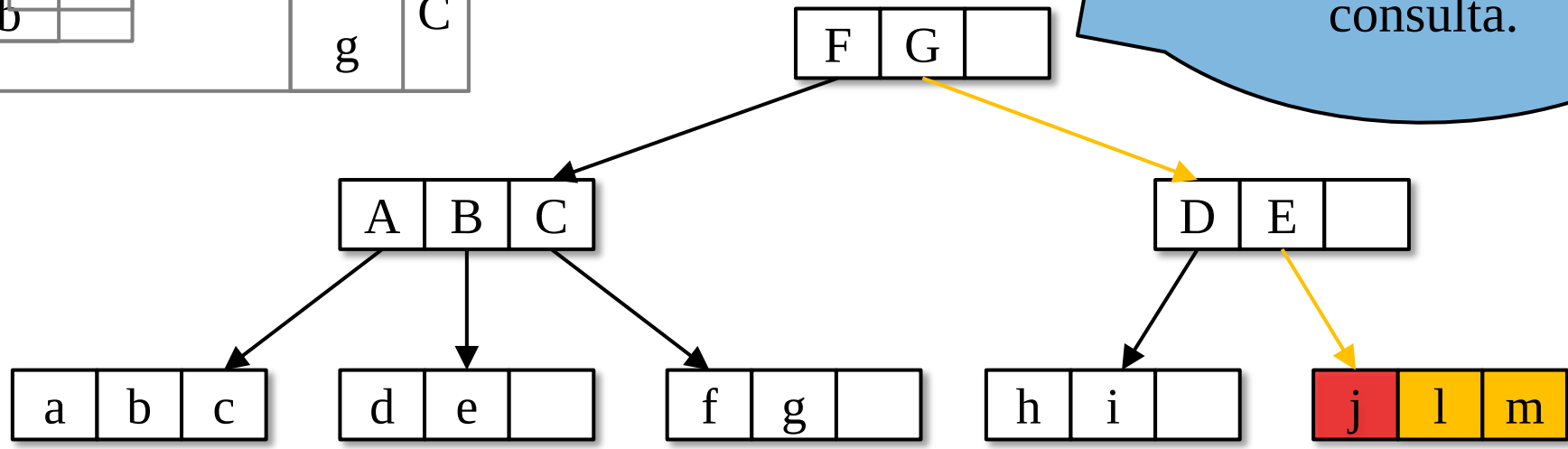
"D" não intercepta "*w*", mas E sim. Então verifica a sub-árvore de E.



Consulta – Exemplo 2

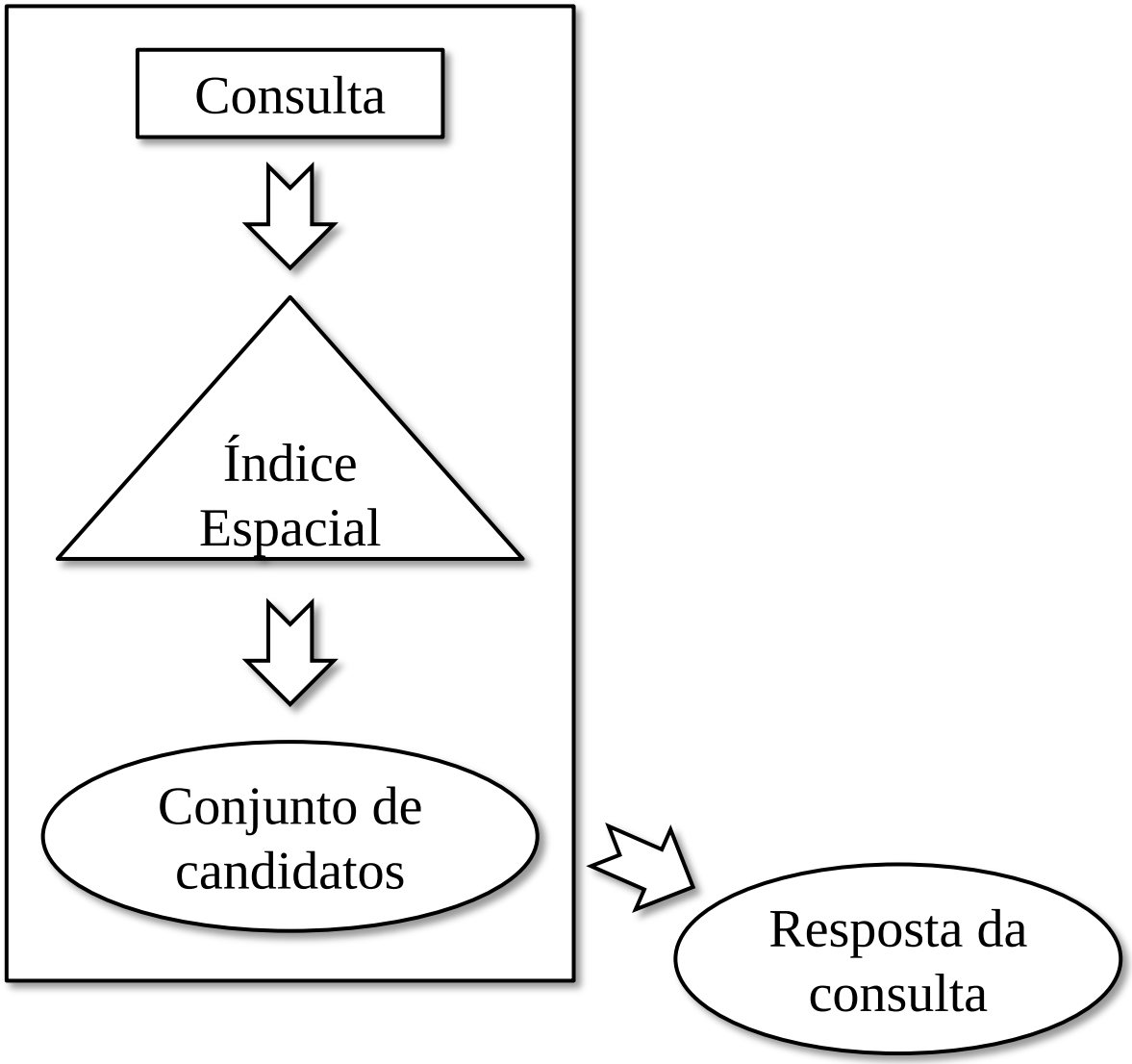


"*j*" não intercepta "*w*", mas "*l*" e "*m*" sim. Então devolve "*l*" e "*m*" como resposta da consulta.

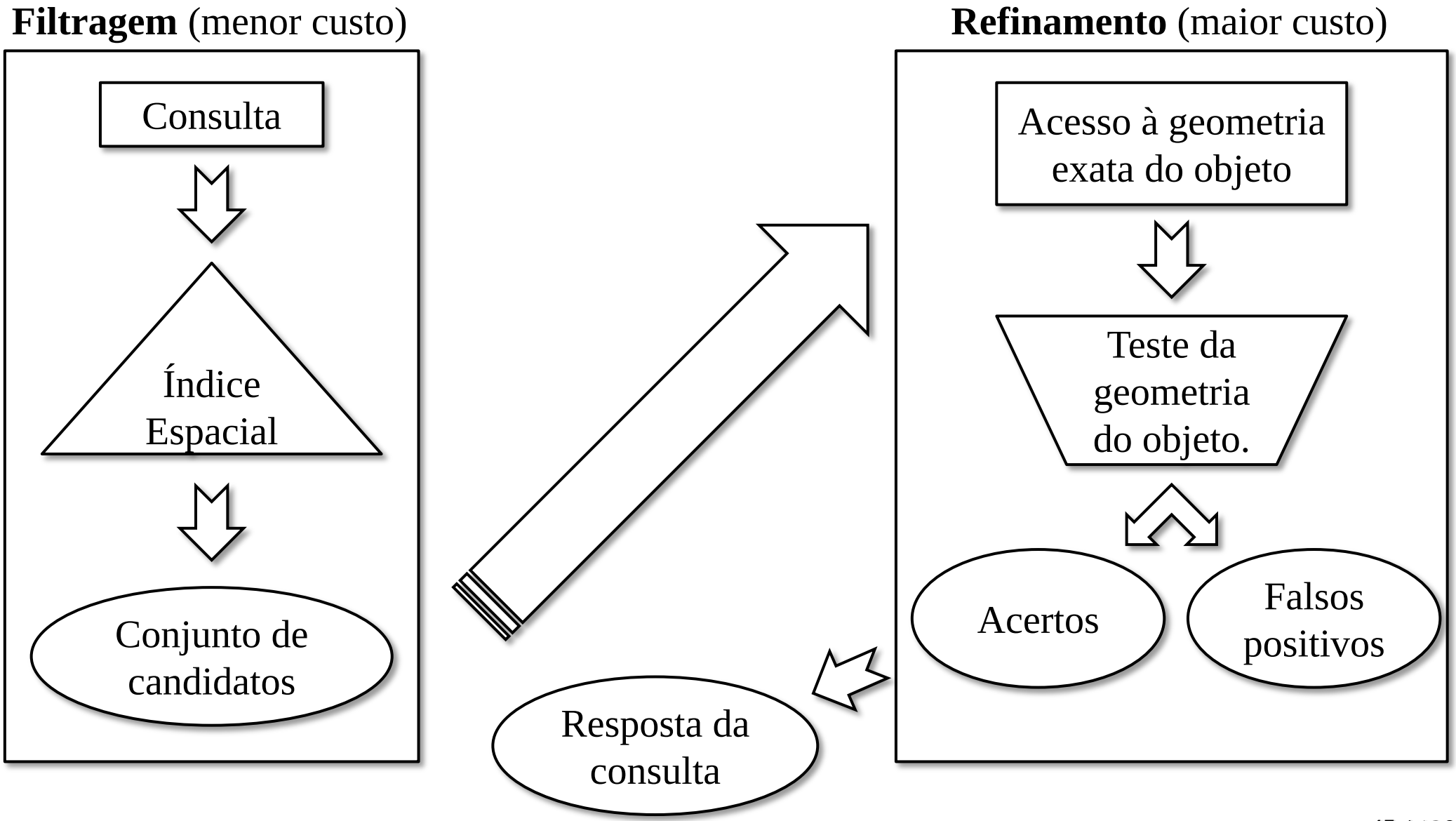


Filtragem e Refinamento

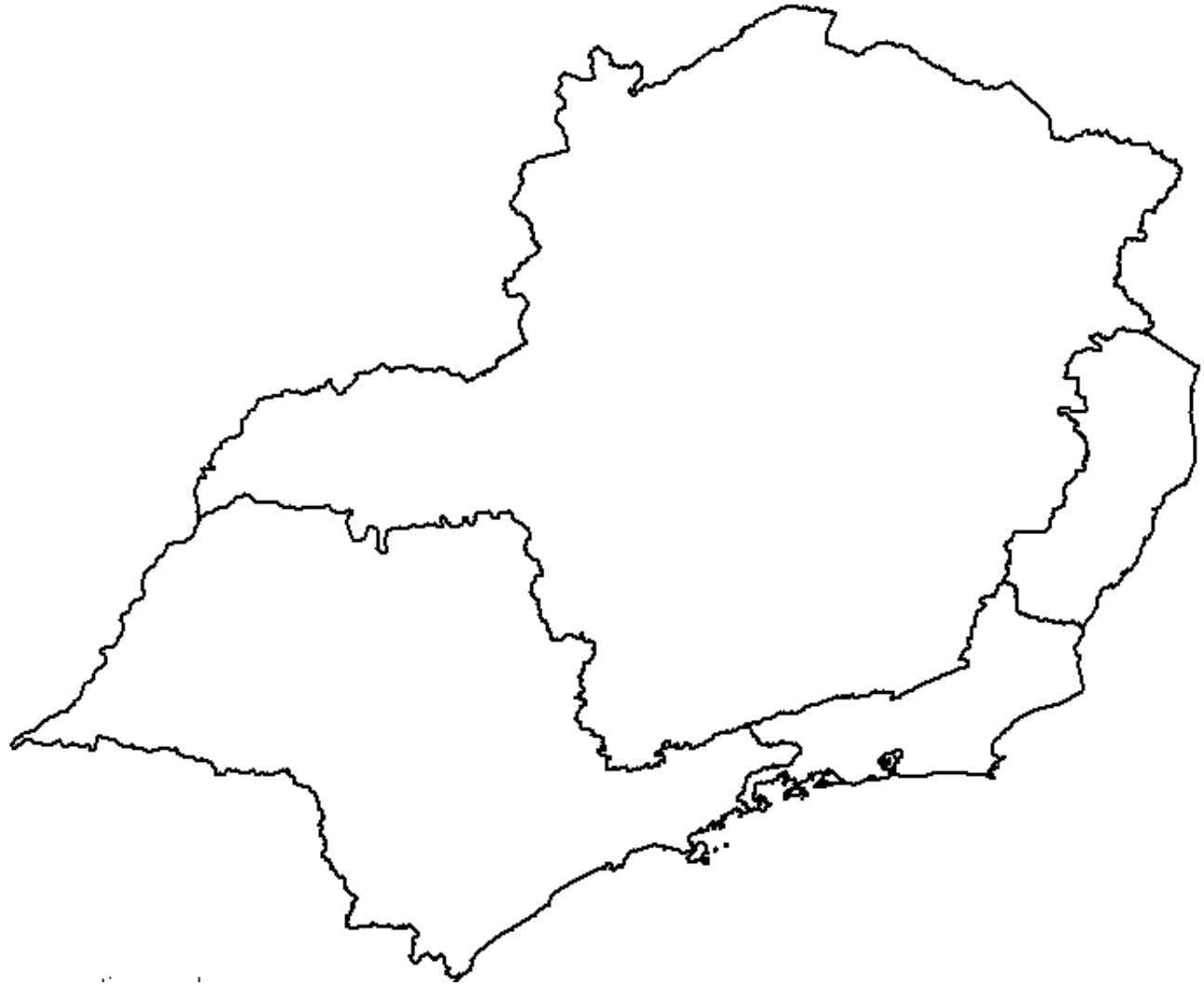
Filtragem (menor custo)



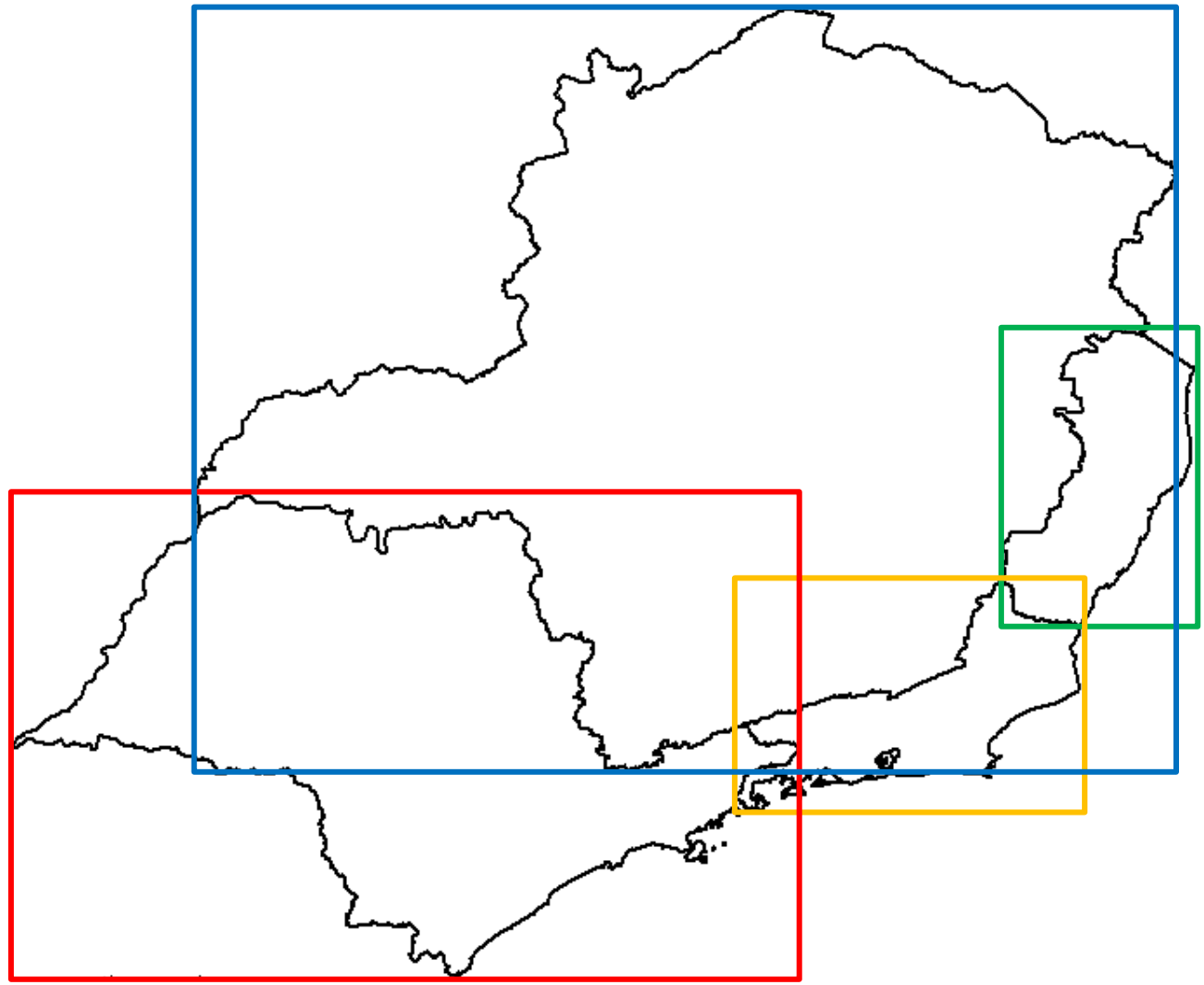
Filtragem e Refinamento



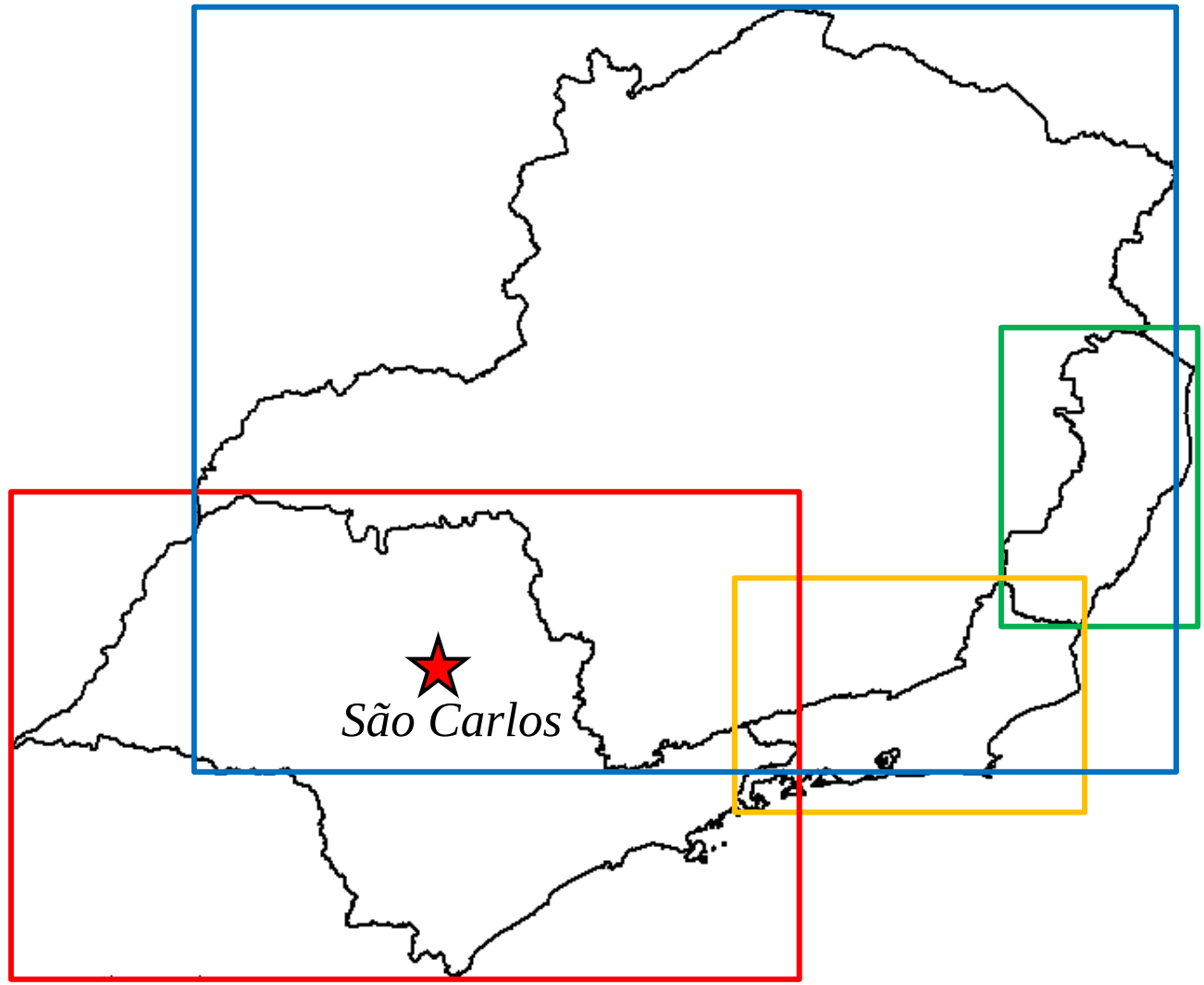
Filtragem e Refinamento



Filtragem e Refinamento



Filtragem e Refinamento



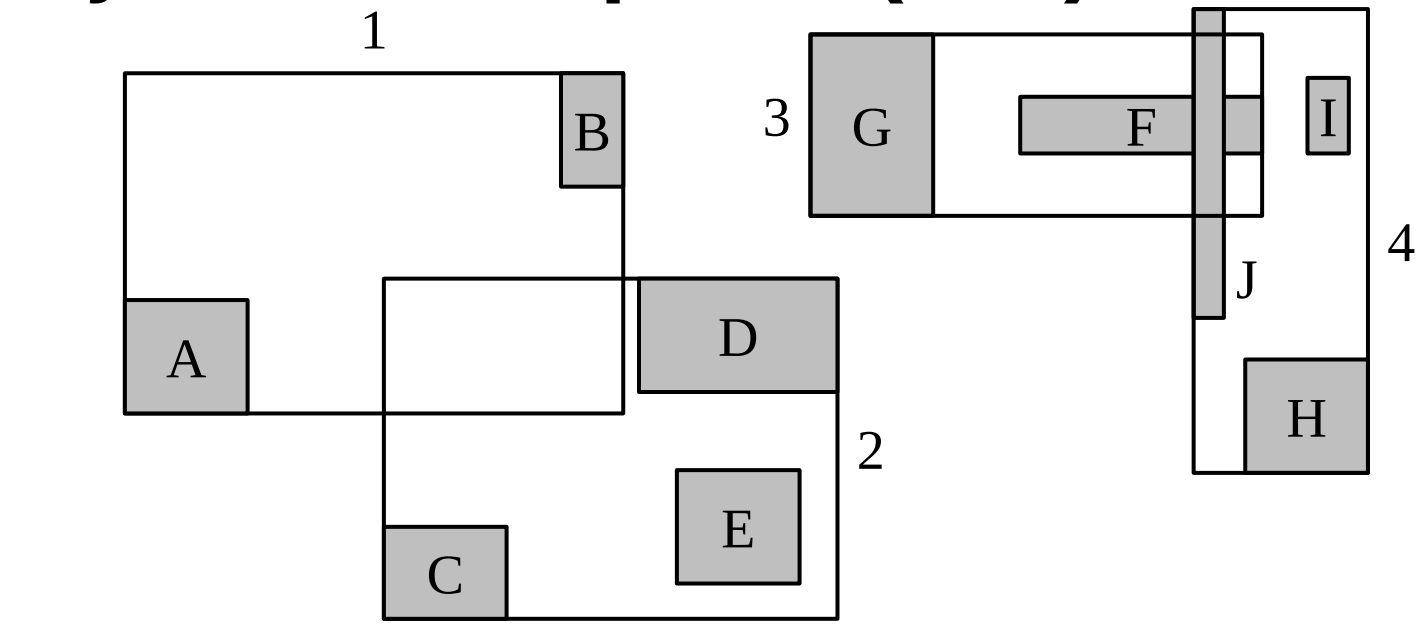
Árvore R

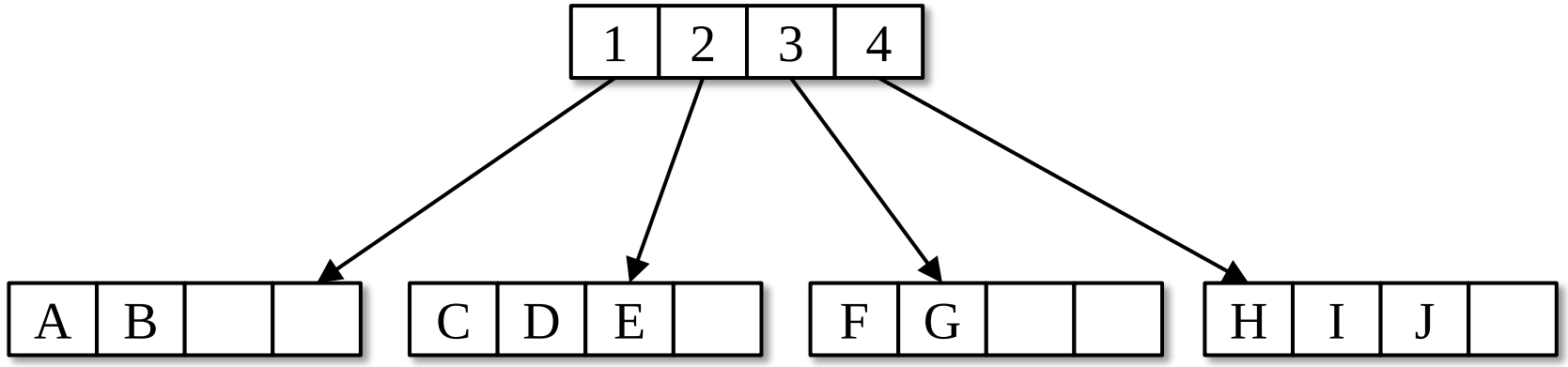
Inserção

Inserção – Algoritmo

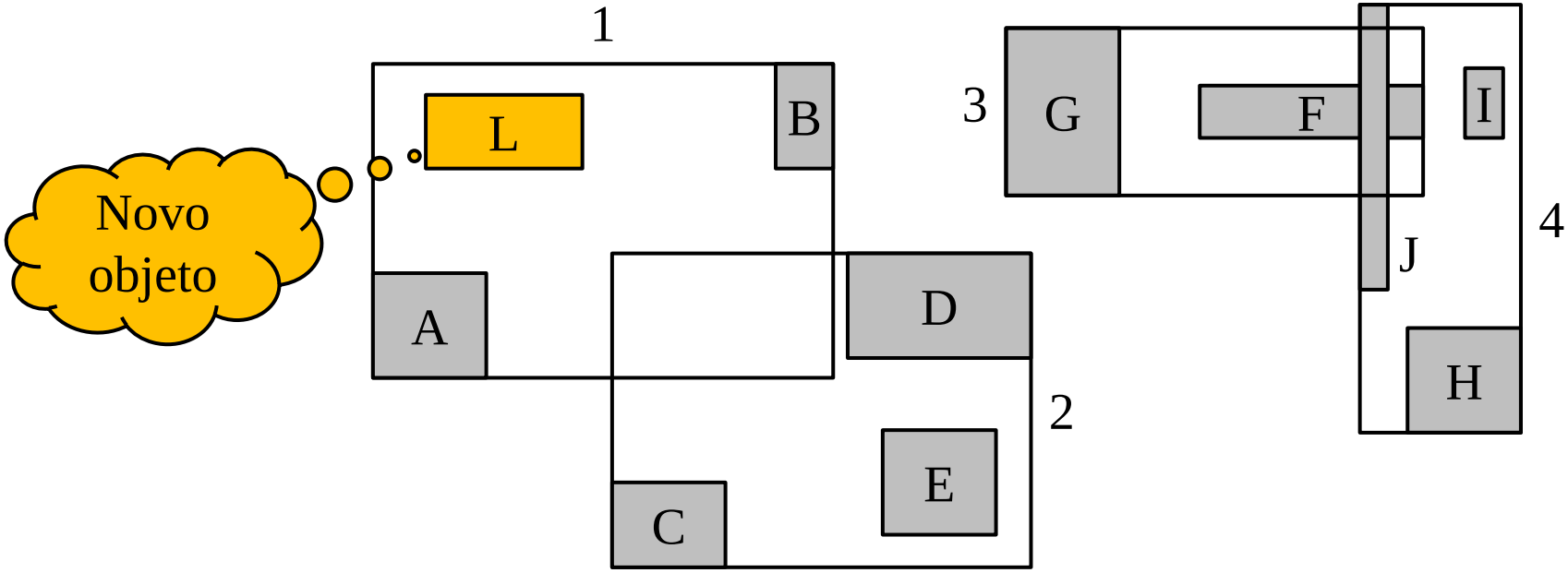
- Percorrer a árvore, a partir do nó raiz, até o nó folha *F* mais apropriado.
  - A cada nível, escolher a entrada cujo MBR necessita do menor aumento de área. Resolver empates selecionando o de menor área.
- Se o nó folha *F* contém espaço suficiente, inserir a nova entrada em *F* e parar o processo de inserção. Caso contrário, dividir a folha *F* em *F1* e *F2*.
  - Ajustar a entrada de *F* no seu nó pai *P* de modo que seu MBR cubra apenas *F1*.
  - Adicionar uma entrada em *P* para *F2*. Este passo pode fazer o nó *P* pode *splitar* recursivamente.
- Propagar as alterações para os níveis superiores.

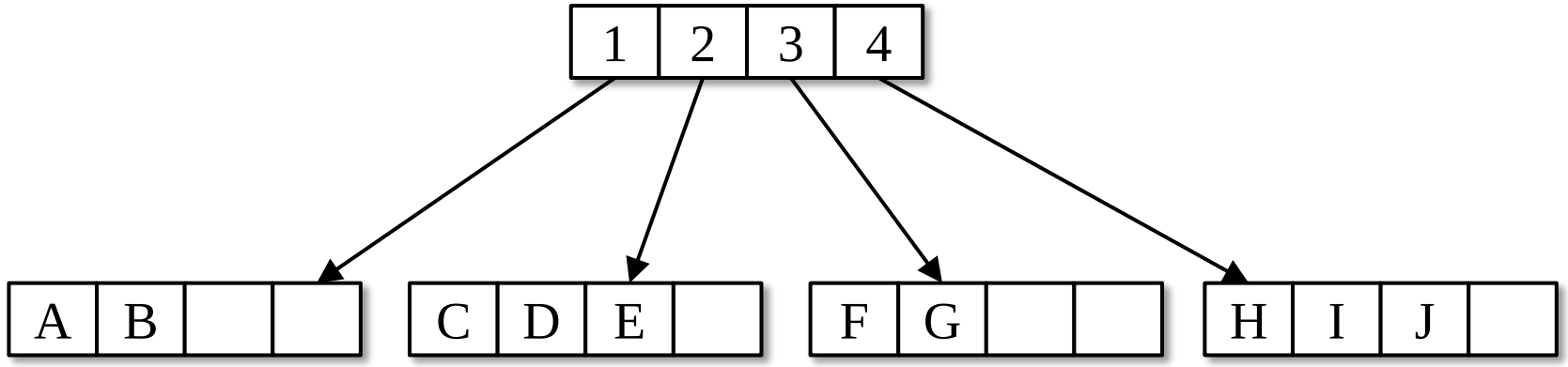
Inserção – Exemplo: R(2, 4)



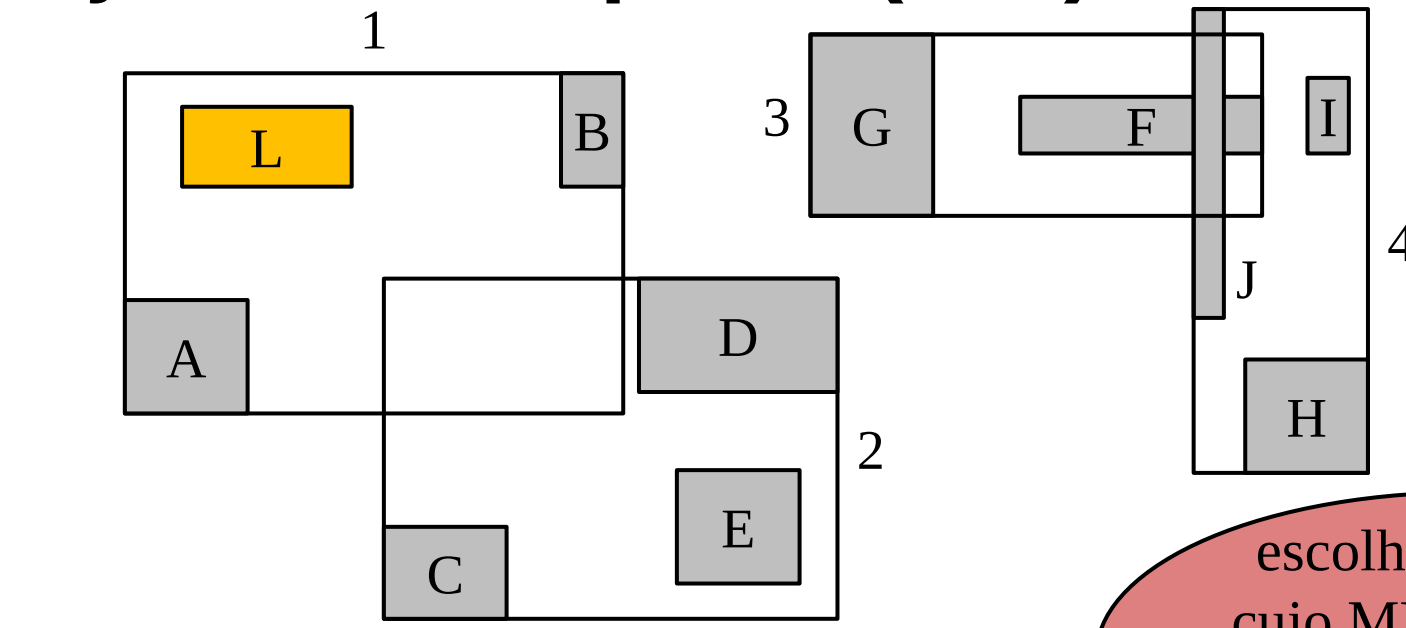


Inserção – Exemplo: R(2, 4)

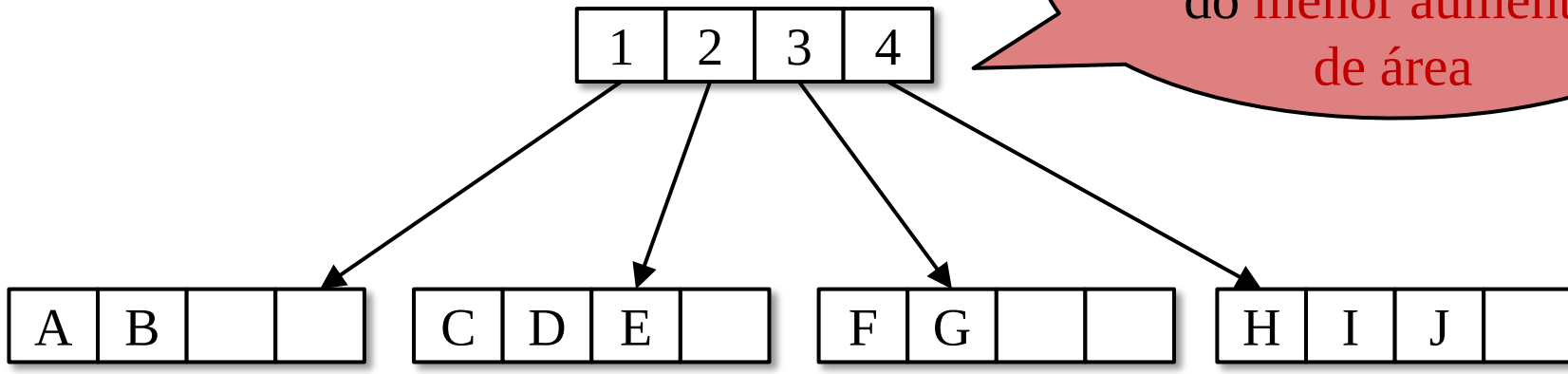




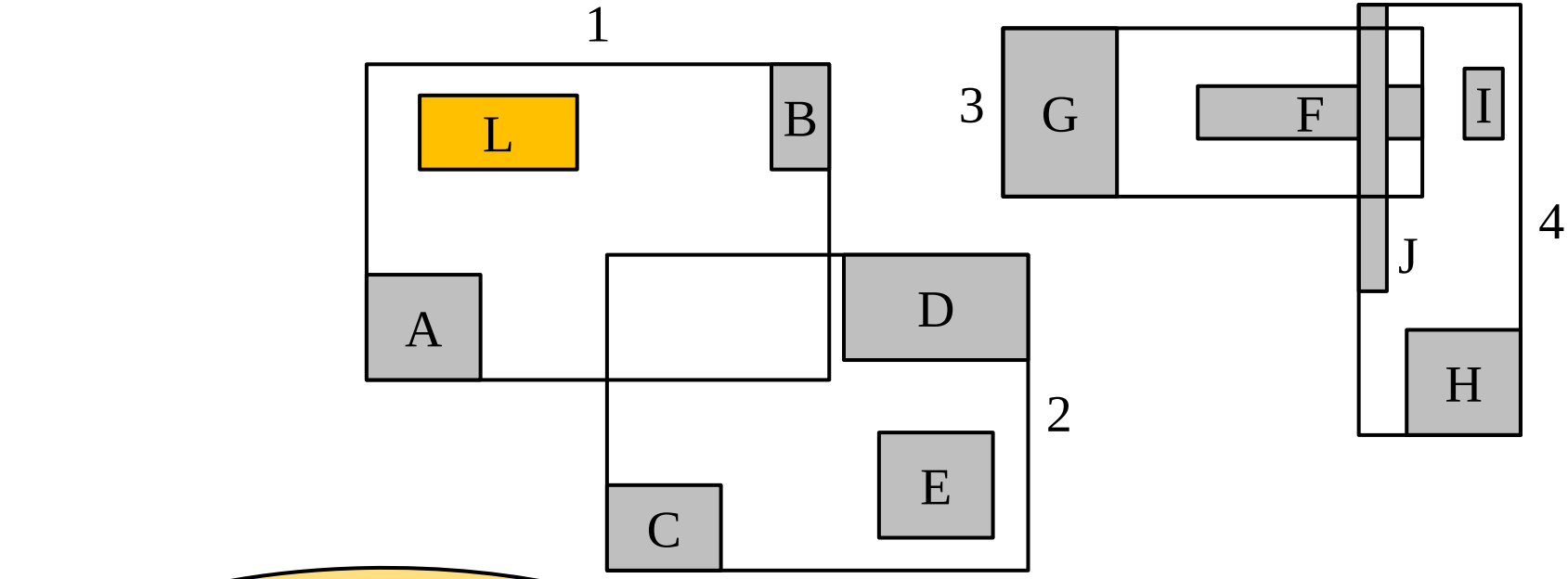
Inserção – Exemplo: R(2, 4)



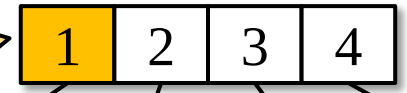
escolher a entrada cujo MBR necessita do menor aumento de área

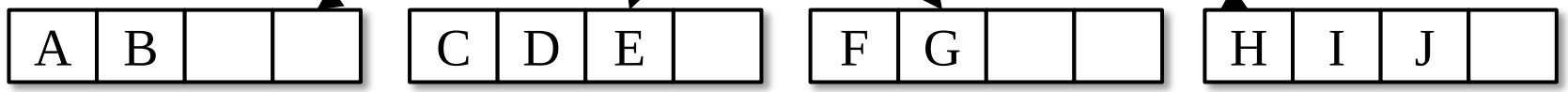


Inserção – Exemplo: R(2, 4)

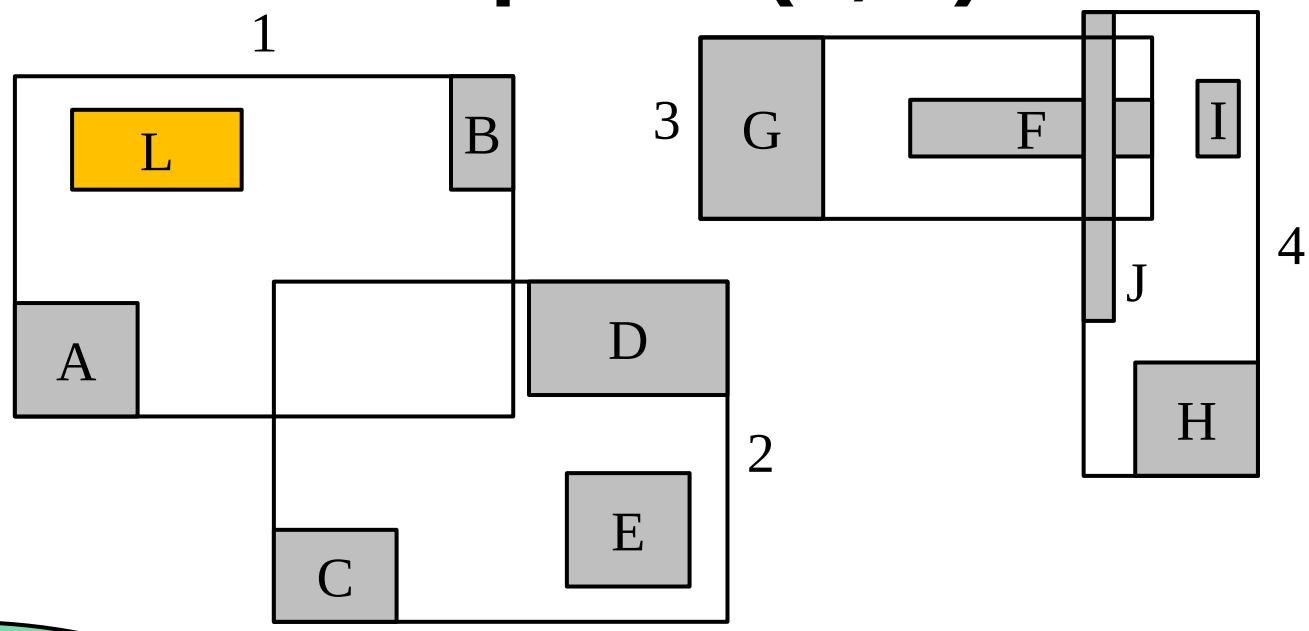


escolhe a entrada 1, pois seu MBR necessita do menor aumento de área: 0

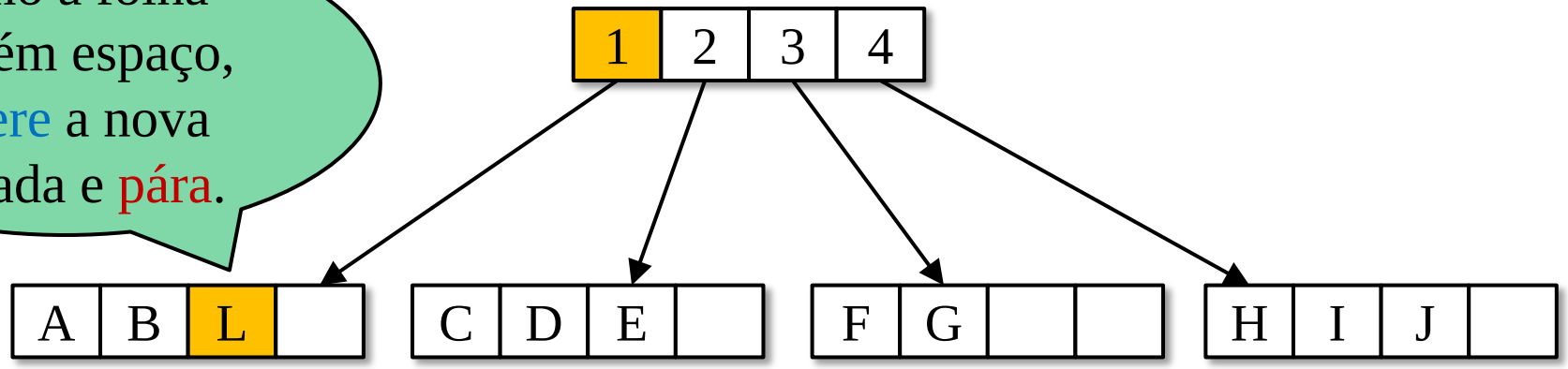




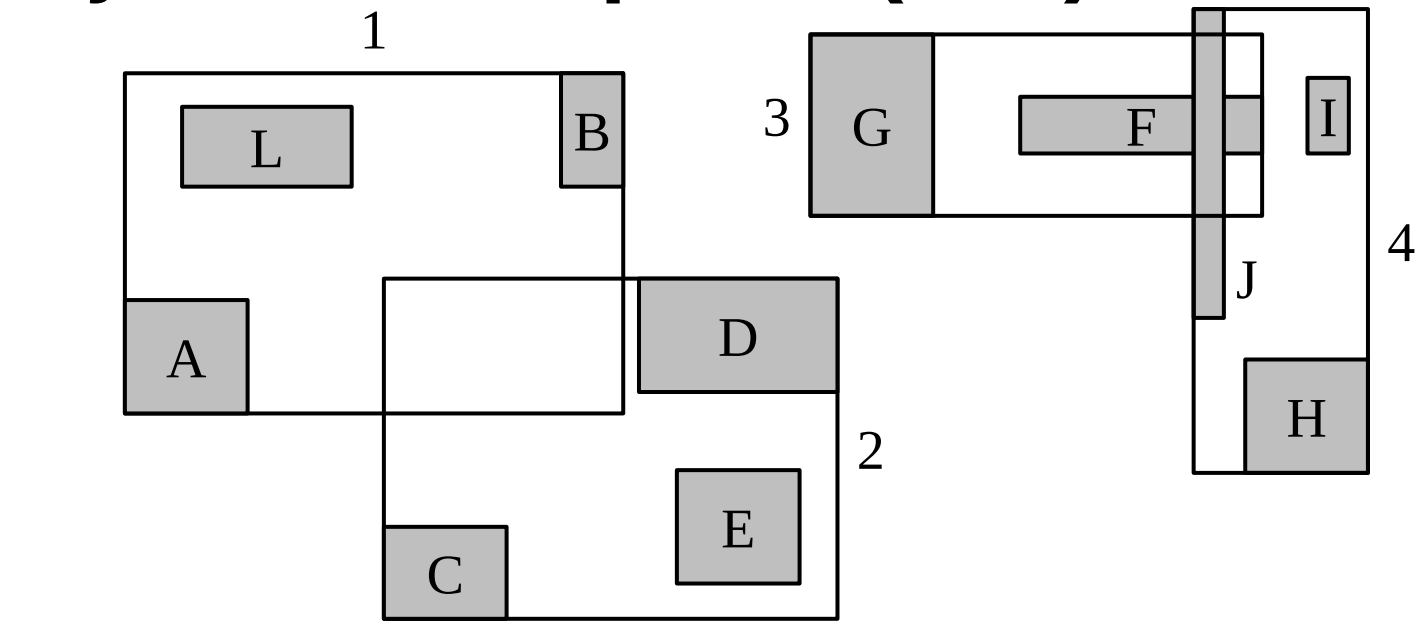
Inserção – Exemplo: R(2, 4)

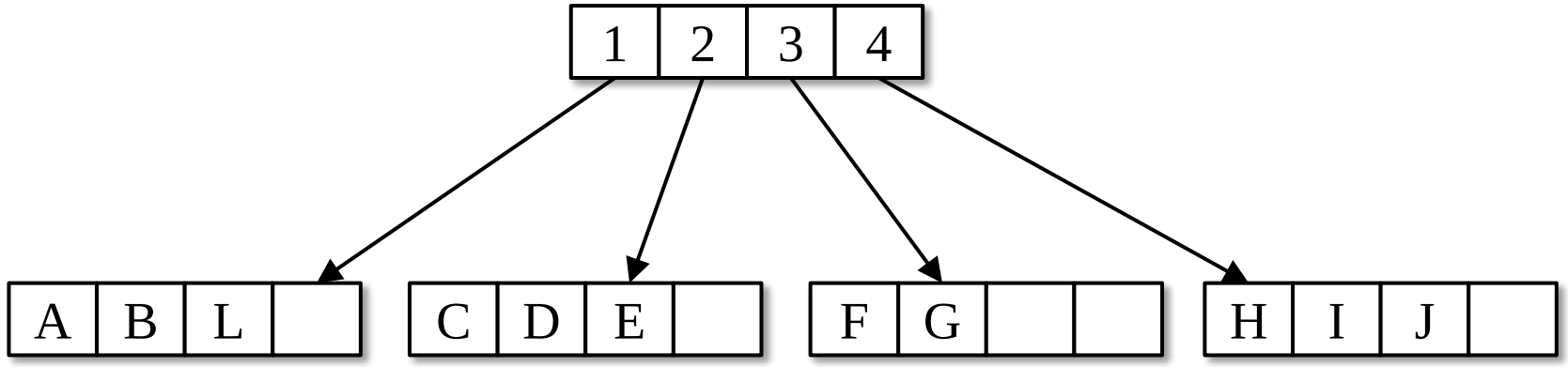


como a folha contém espaço, insere a nova entrada e pára.

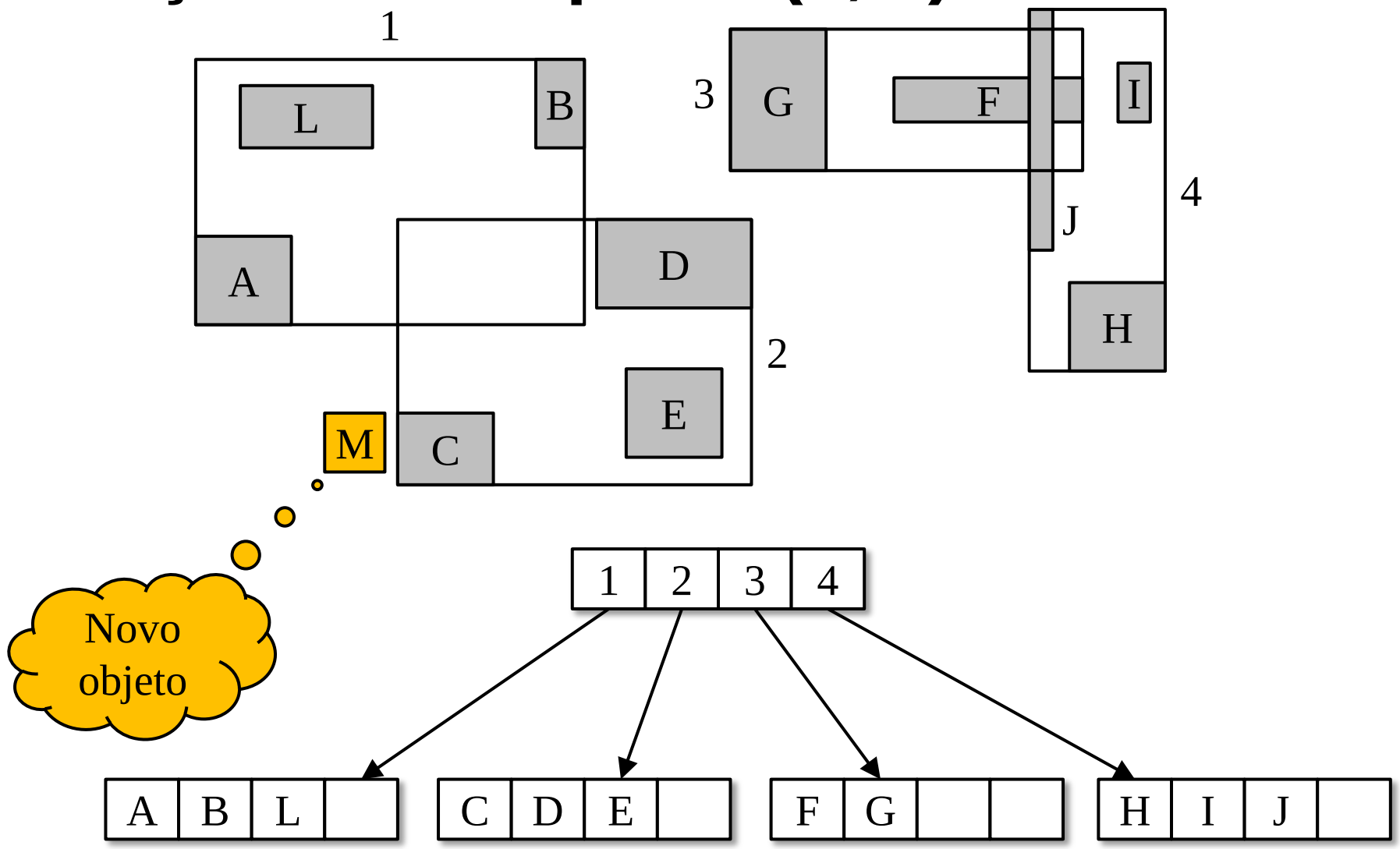


Inserção – Exemplo: R(2, 4)

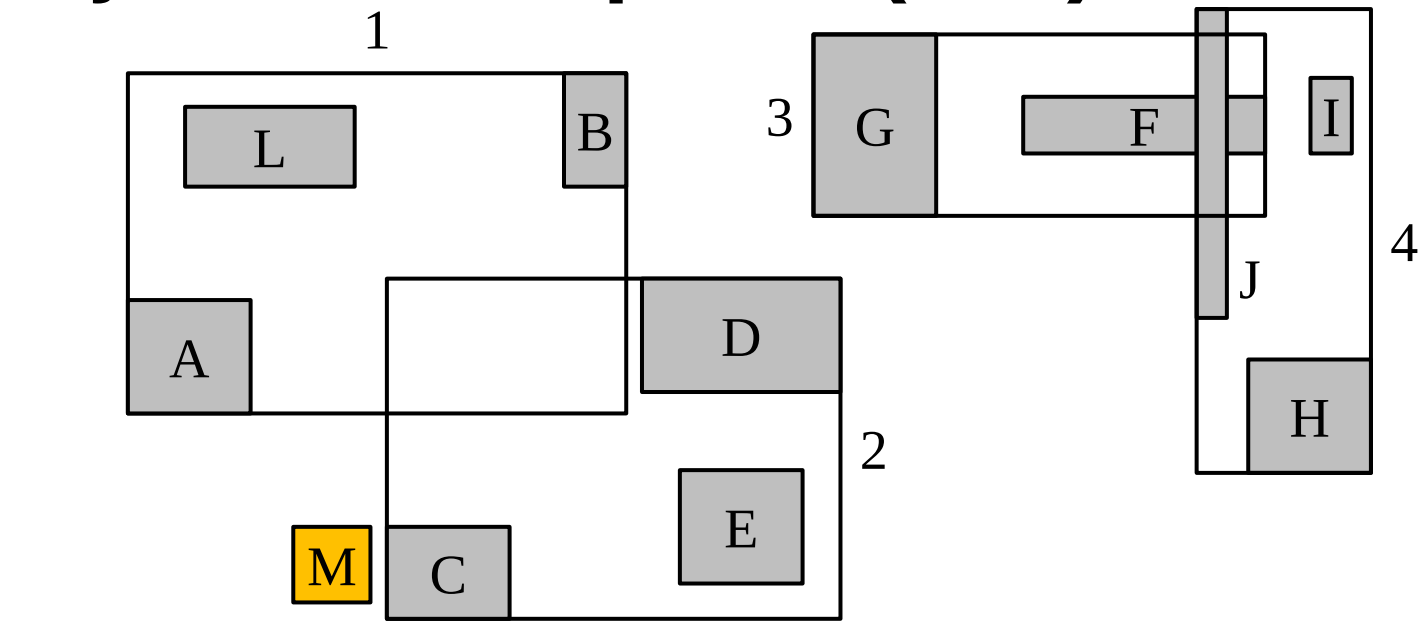


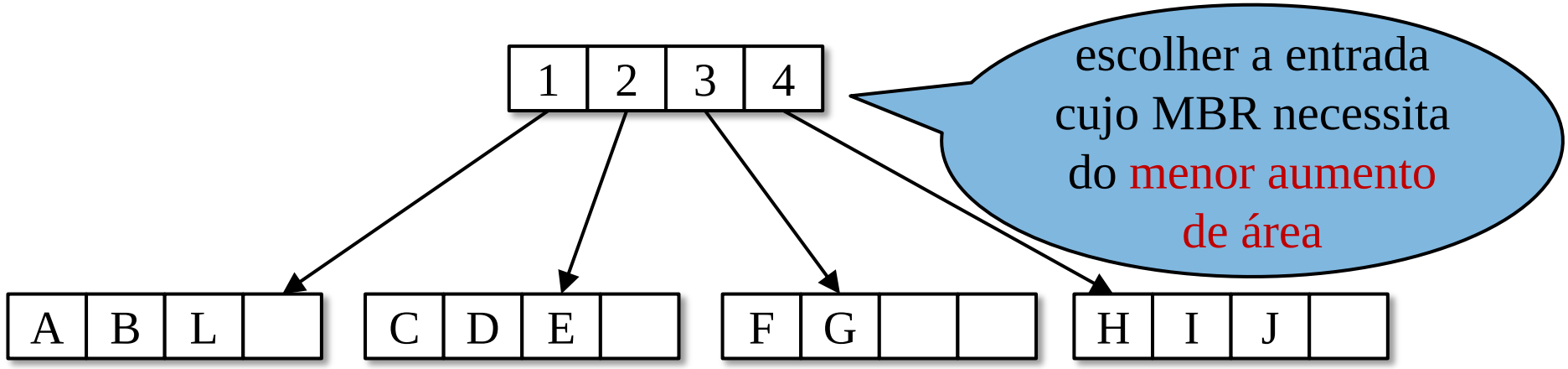


Inserção – Exemplo: R(2, 4)

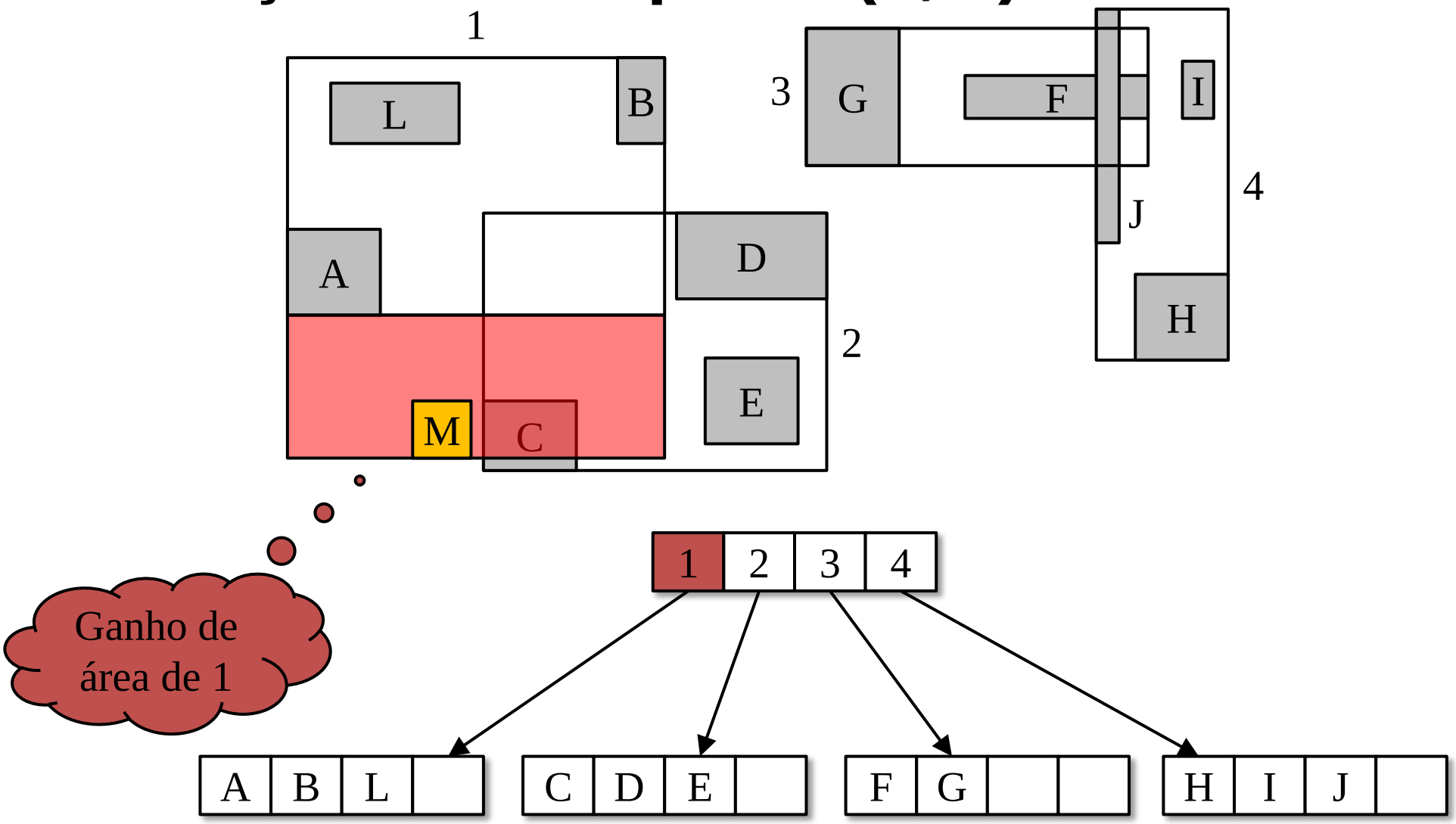


Inserção – Exemplo: R(2, 4)

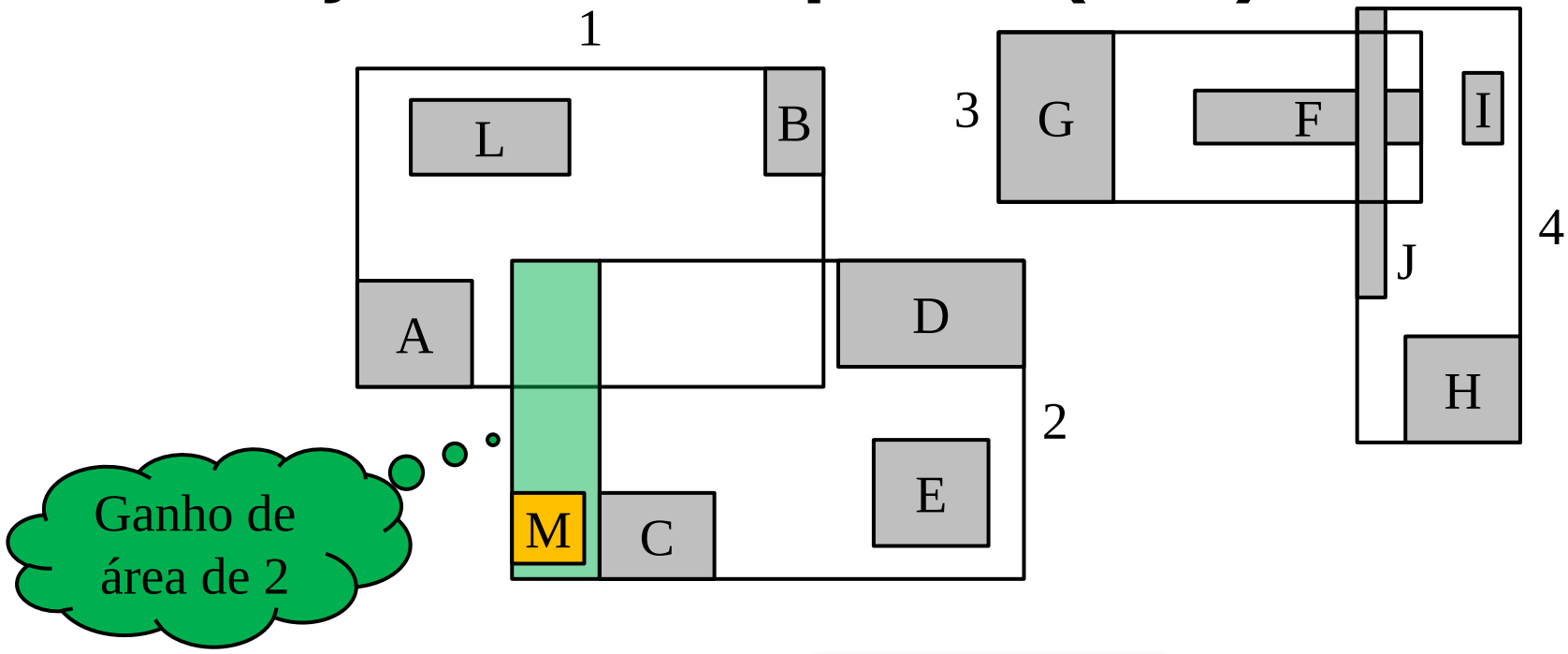


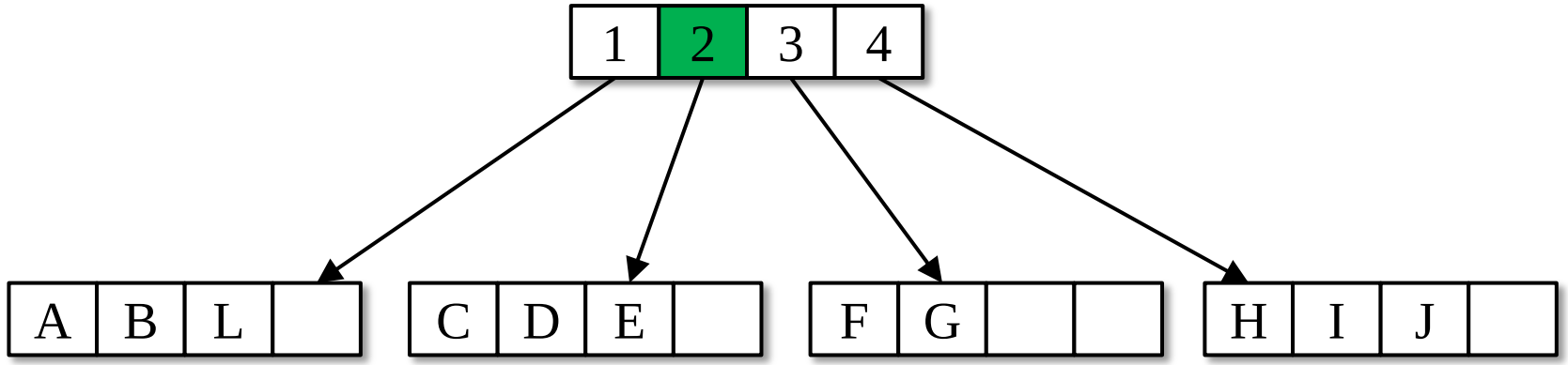


Inserção – Exemplo: R(2, 4)

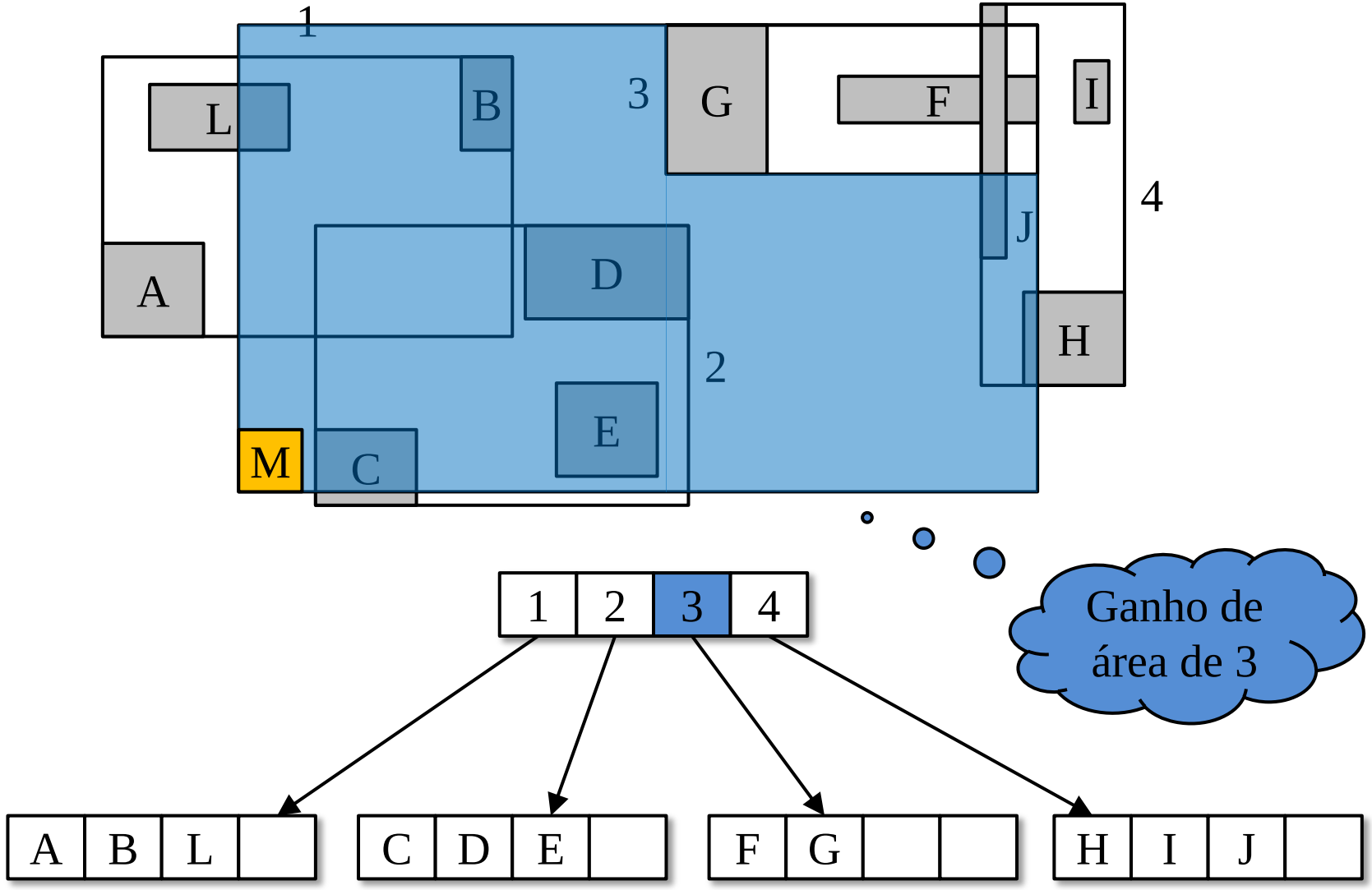


Inserção – Exemplo: R(2, 4)

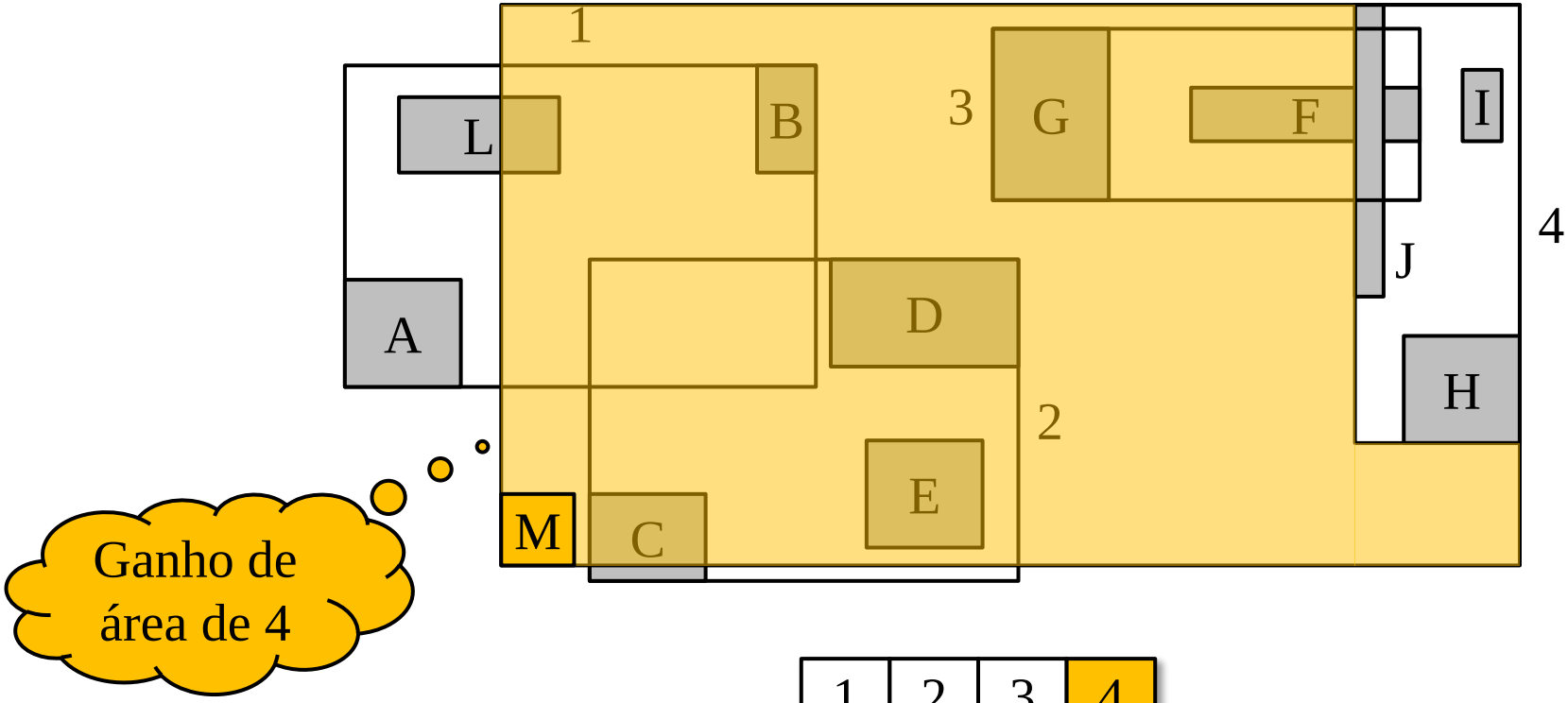


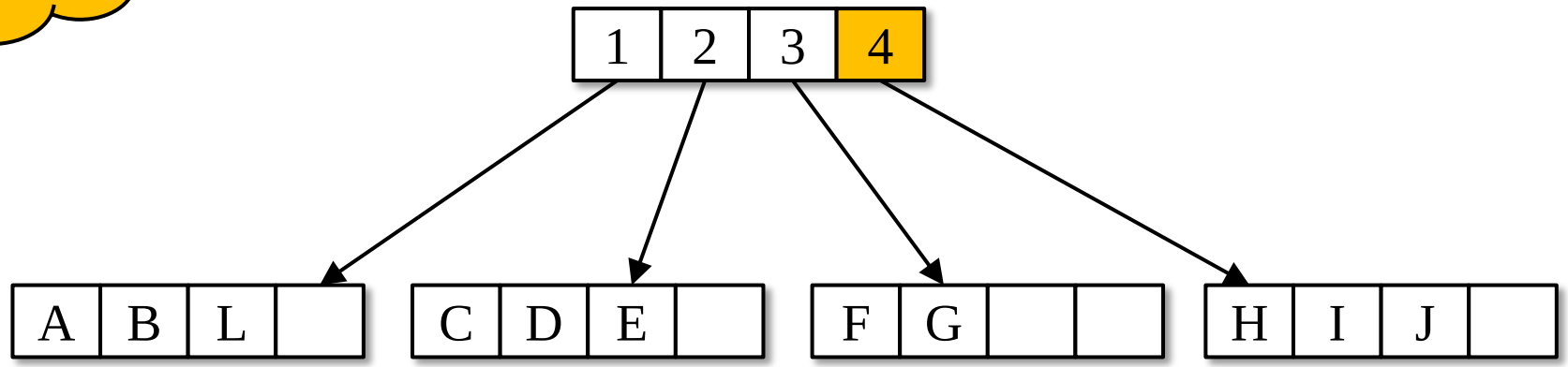


Inserção – Exemplo: R(2, 4)

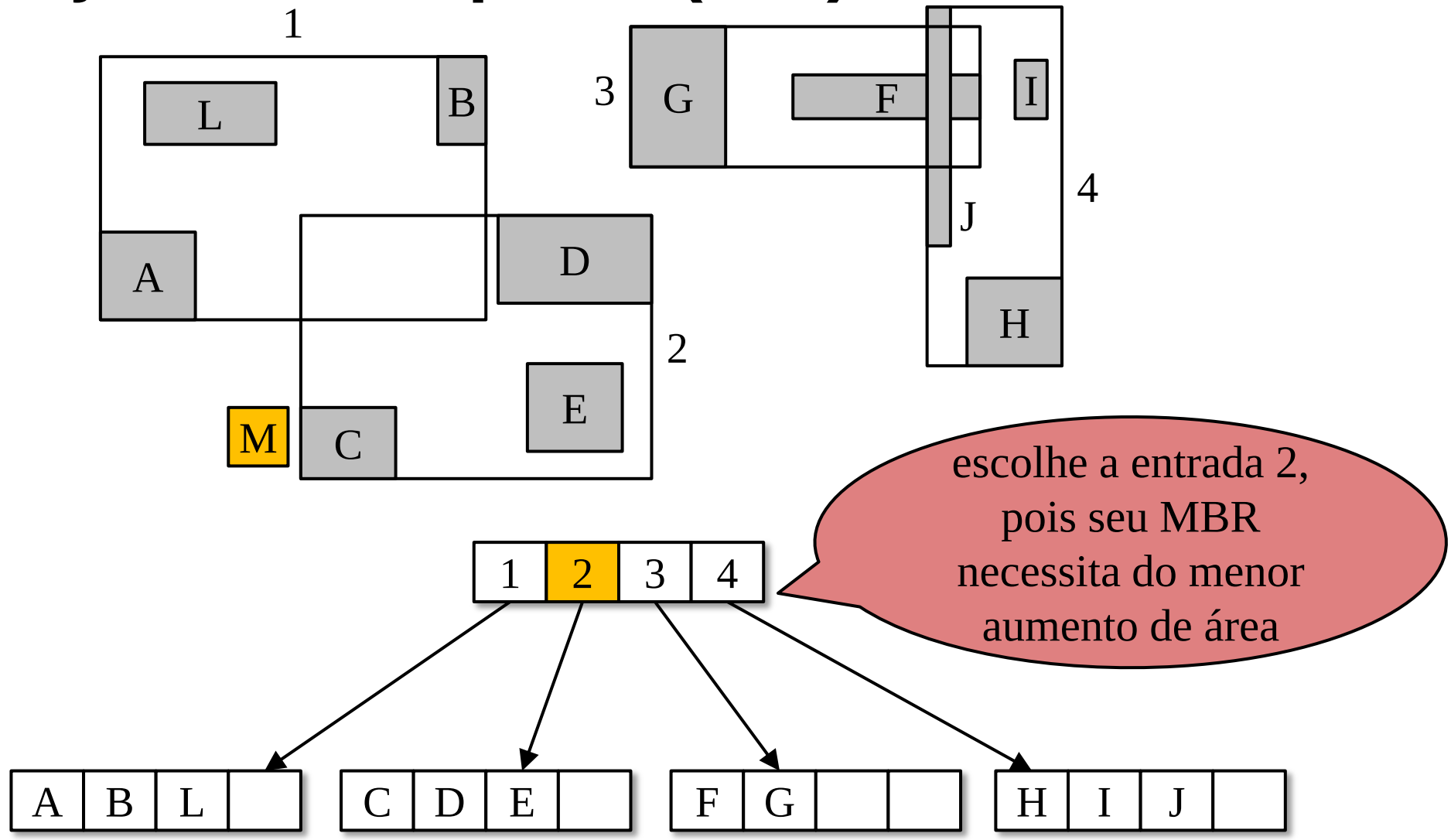


Inserção – Exemplo: R(2, 4)

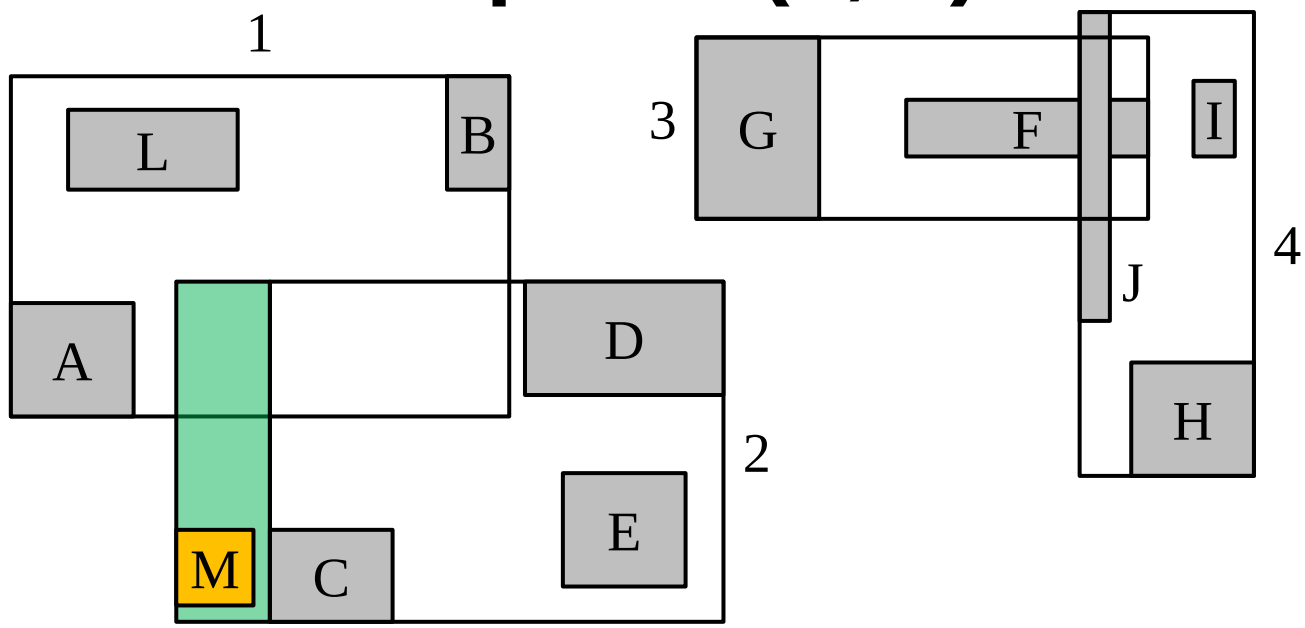




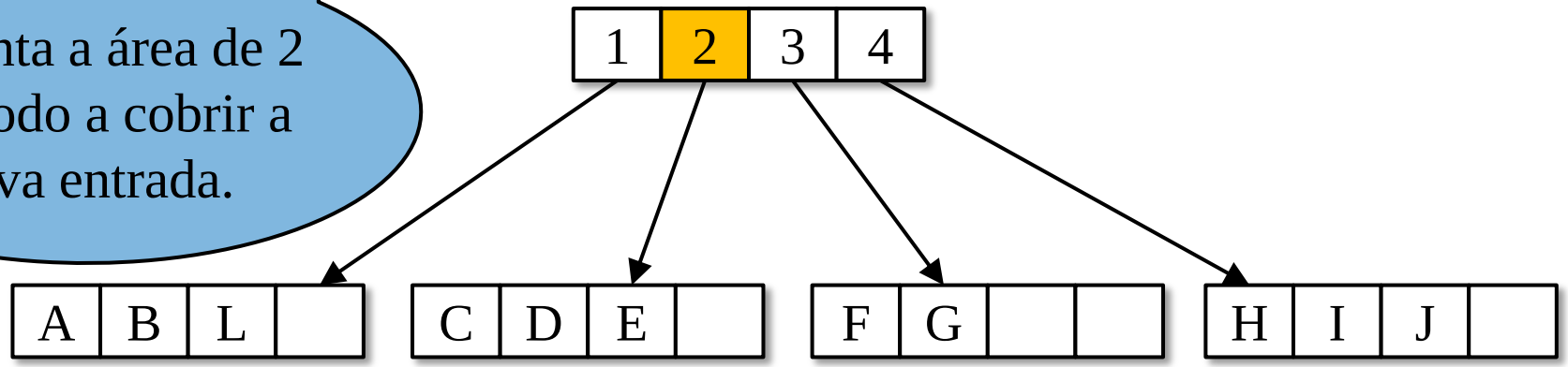
Inserção – Exemplo: R(2, 4)



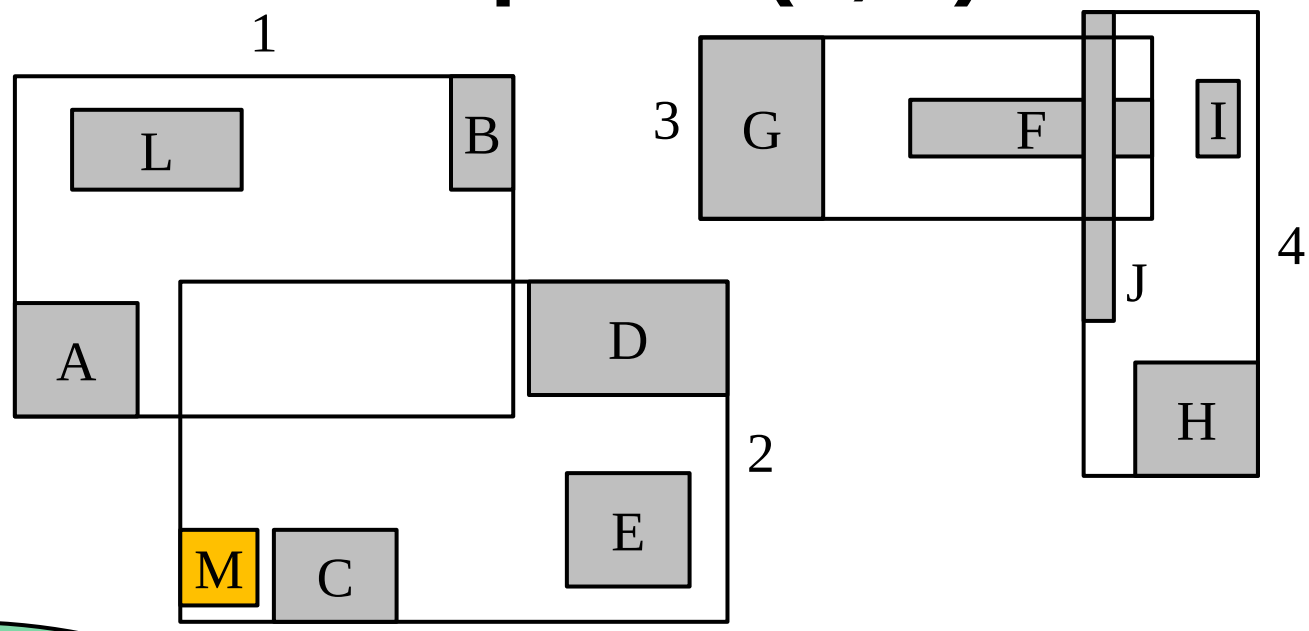
Inserção – Exemplo: R(2, 4)



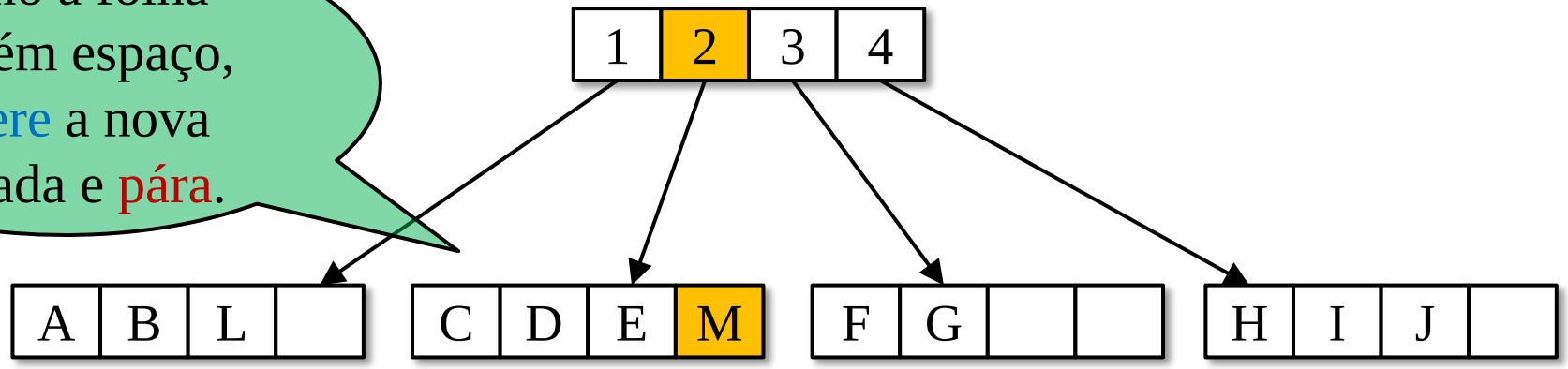
aumenta a área de 2 de modo a cobrir a nova entrada.



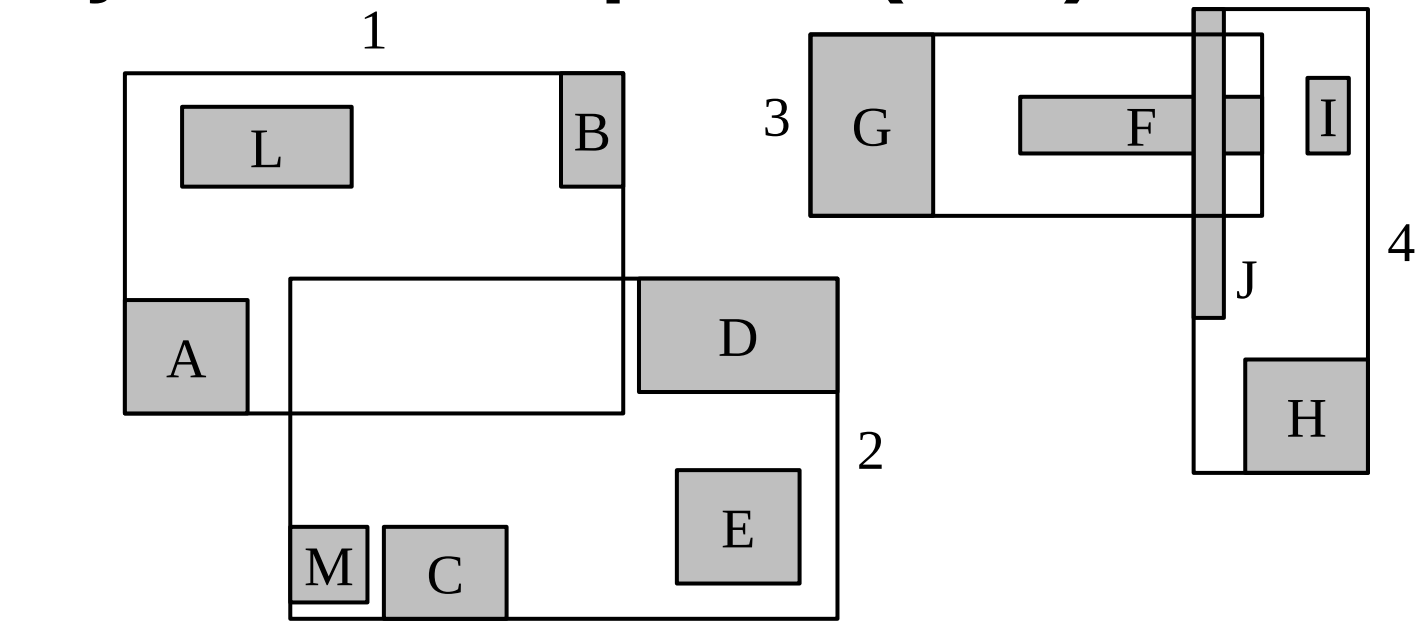
Inserção – Exemplo: R(2, 4)

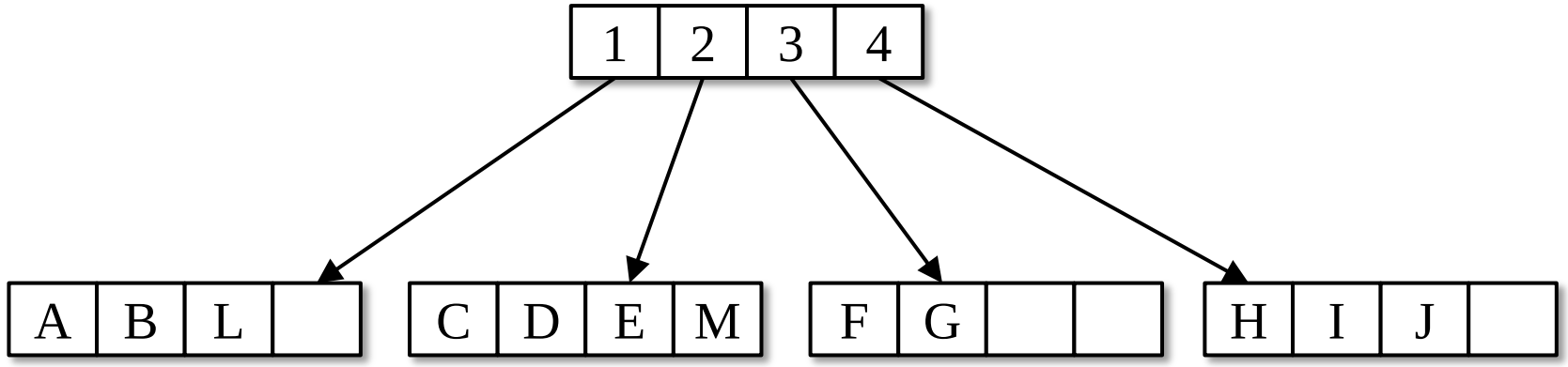


como a folha contém espaço, insere a nova entrada e pára.

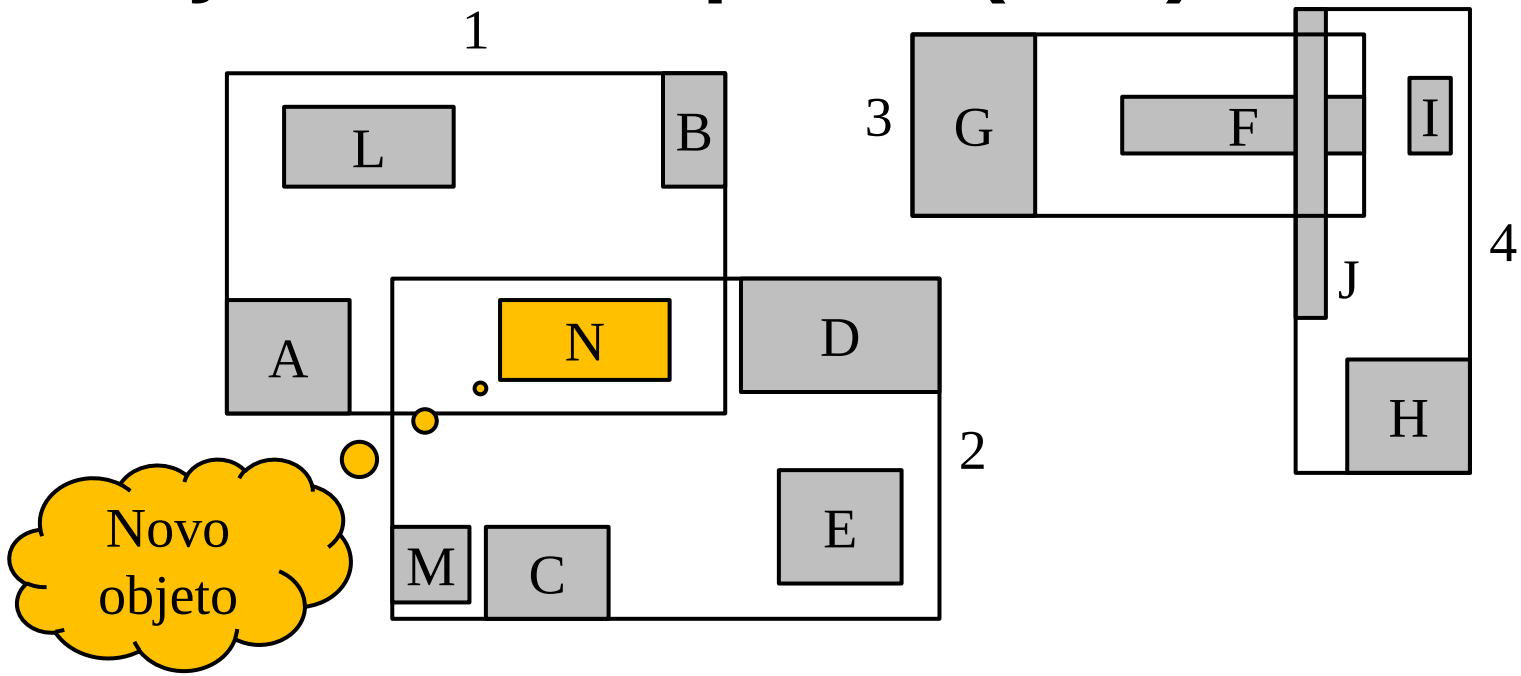


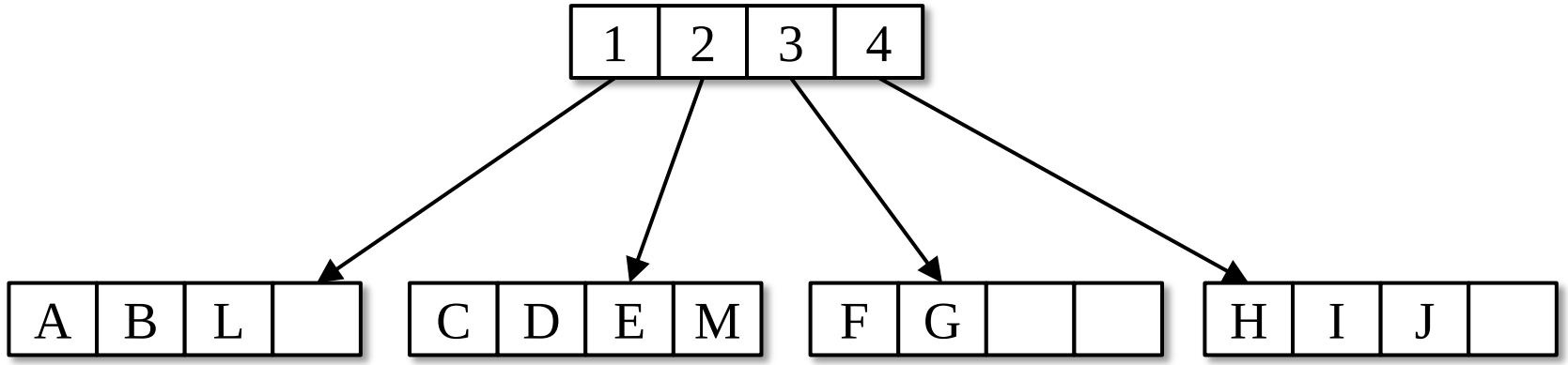
Inserção – Exemplo: R(2, 4)



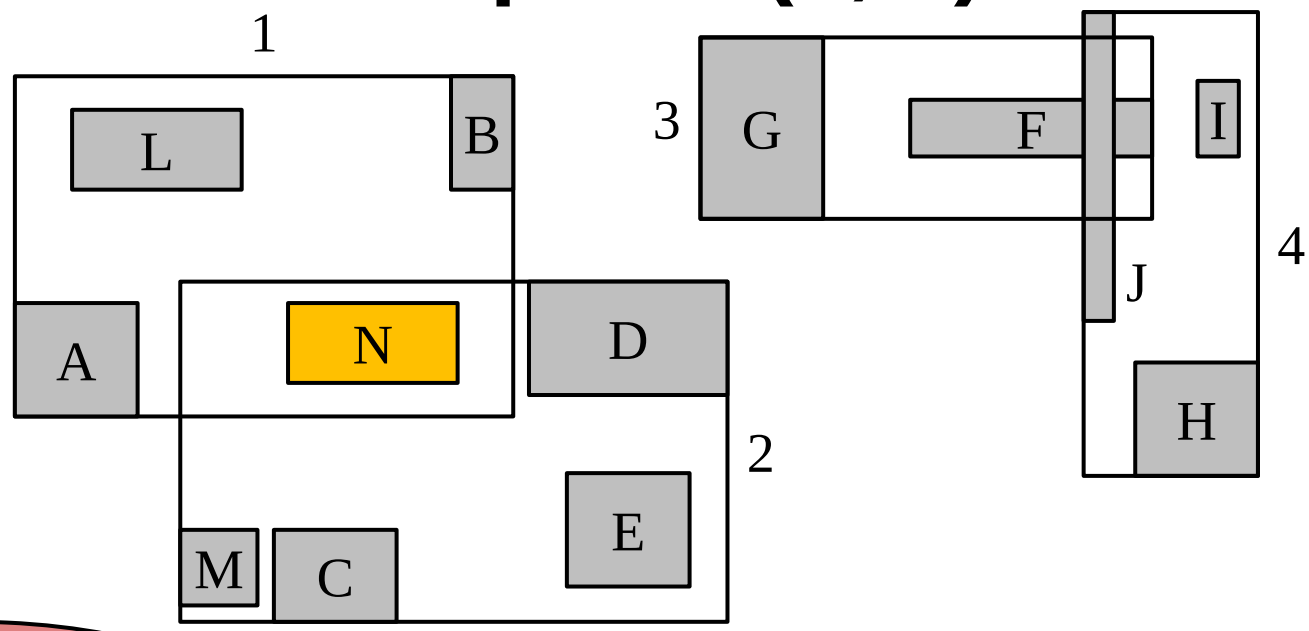


Inserção – Exemplo: R(2, 4)

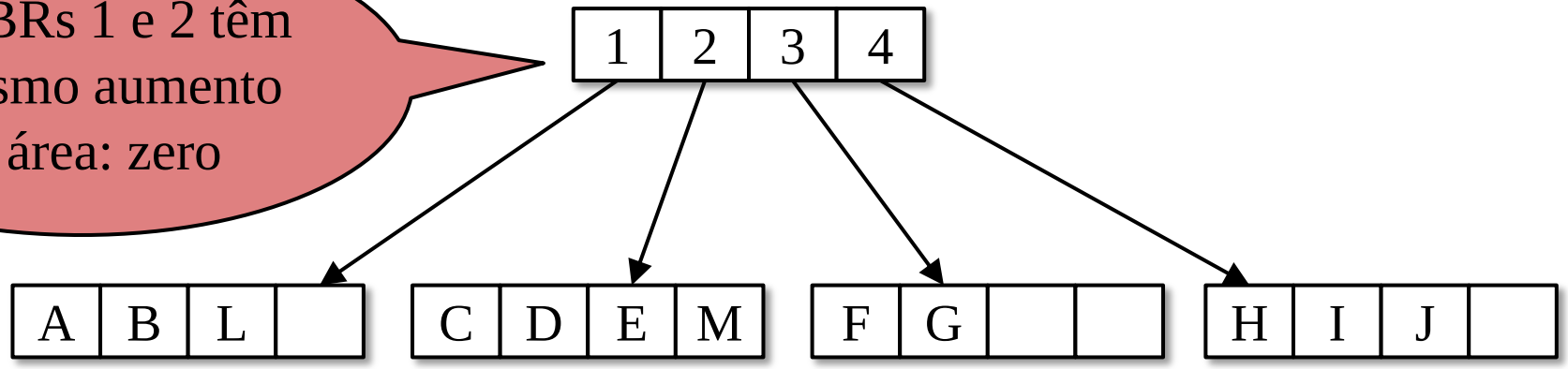




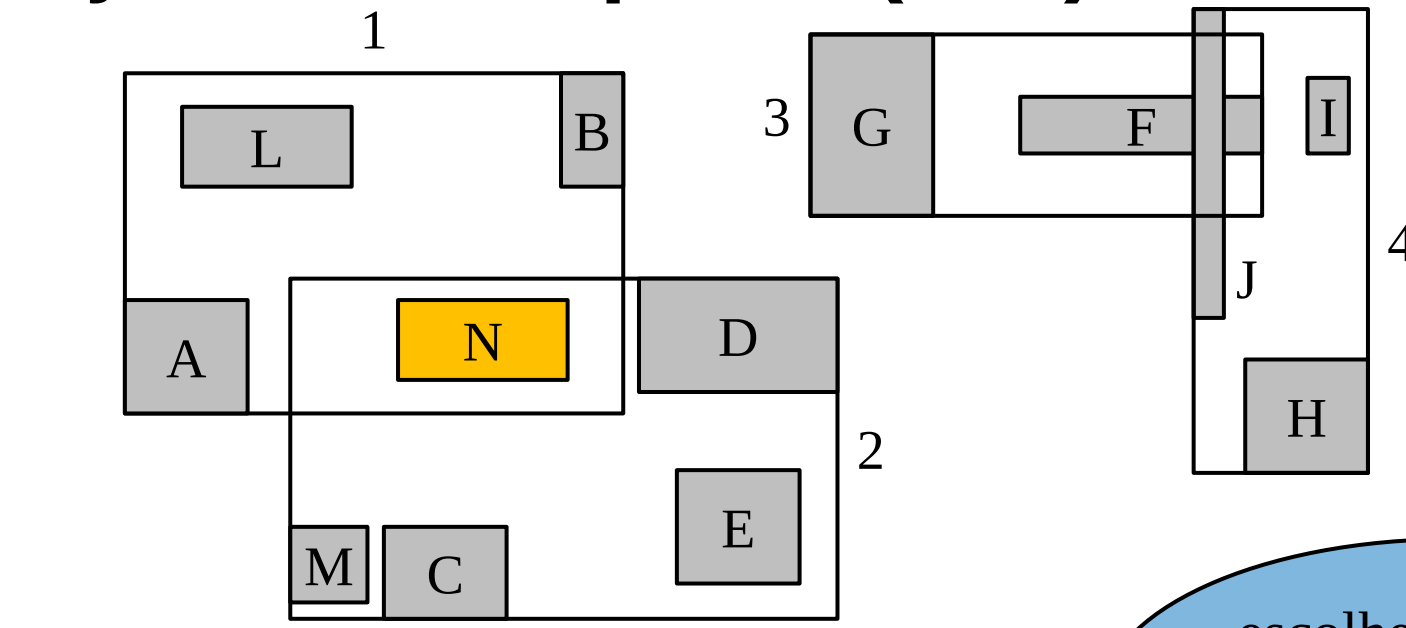
Inserção – Exemplo: R(2, 4)

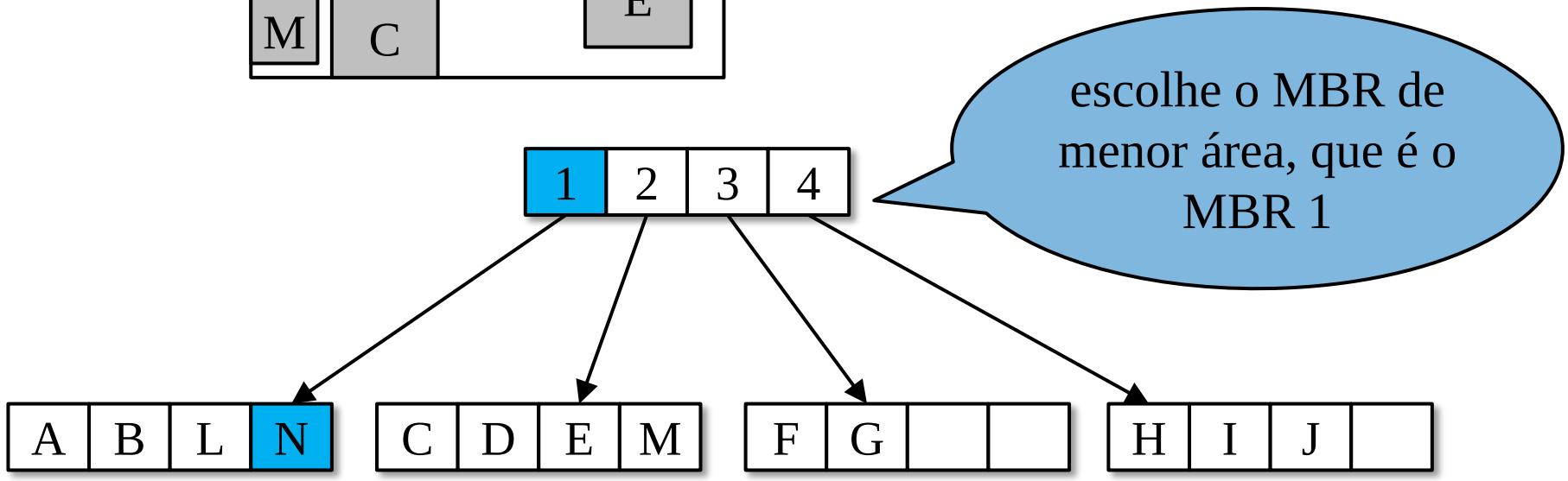


os MBRs 1 e 2 têm o mesmo aumento de área: zero

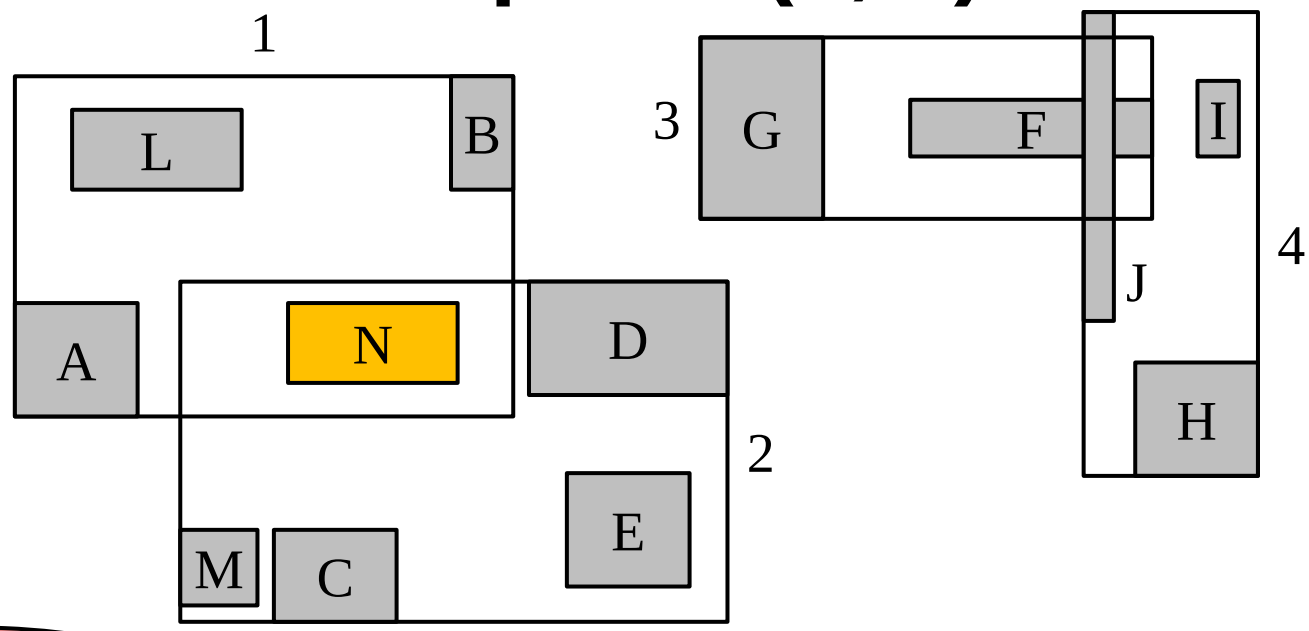


Inserção – Exemplo: R(2, 4)

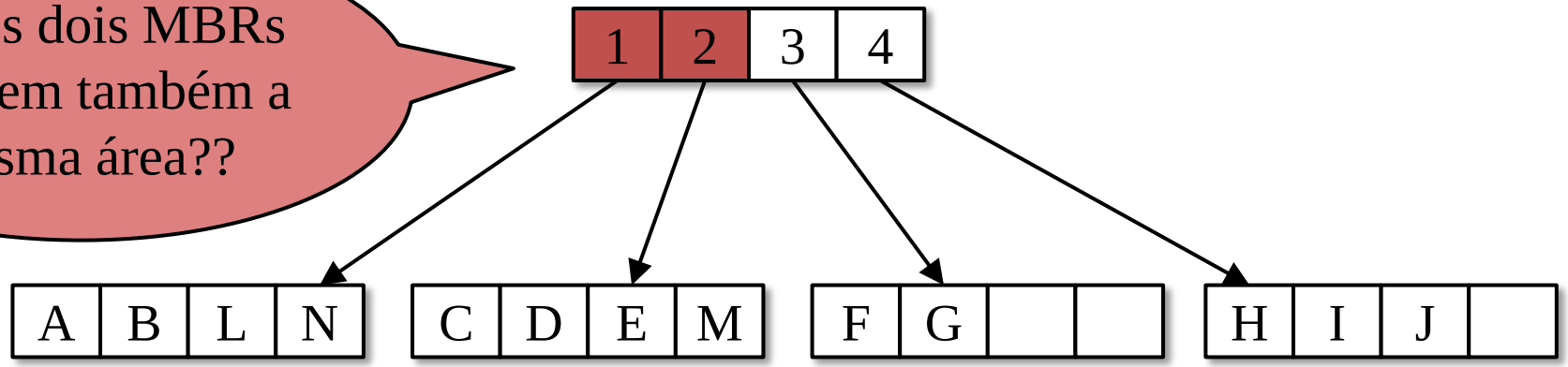




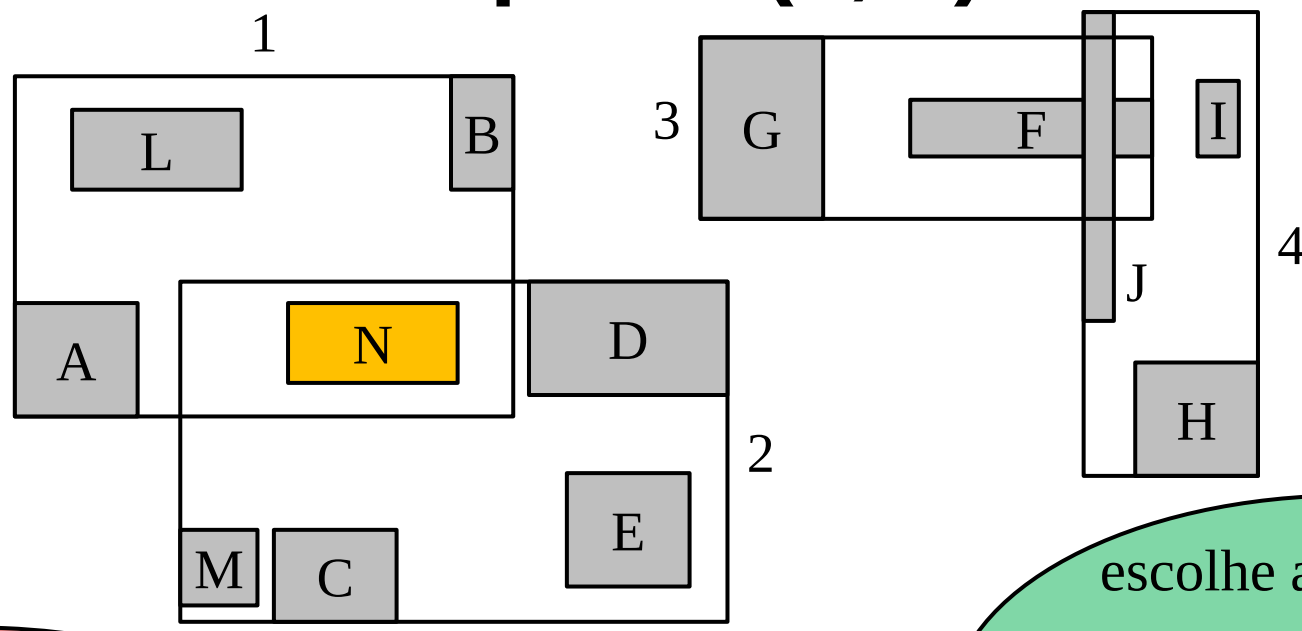
Inserção – Exemplo: R(2, 4)



e se os dois MBRs tivessem também a mesma área??

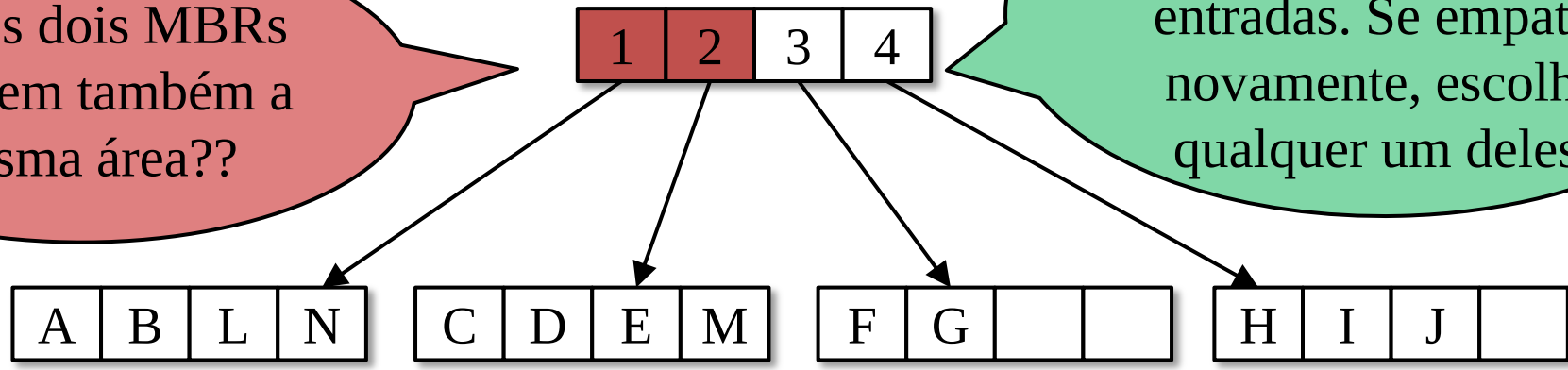


Inserção – Exemplo: R(2, 4)

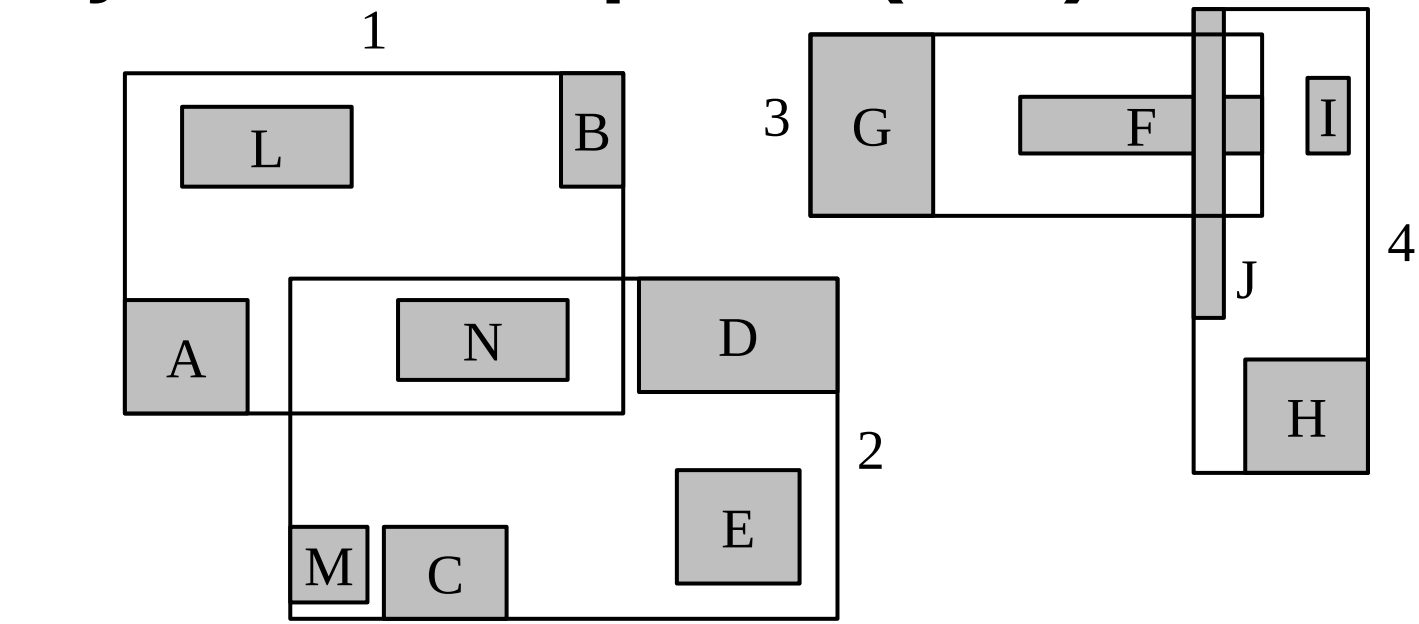


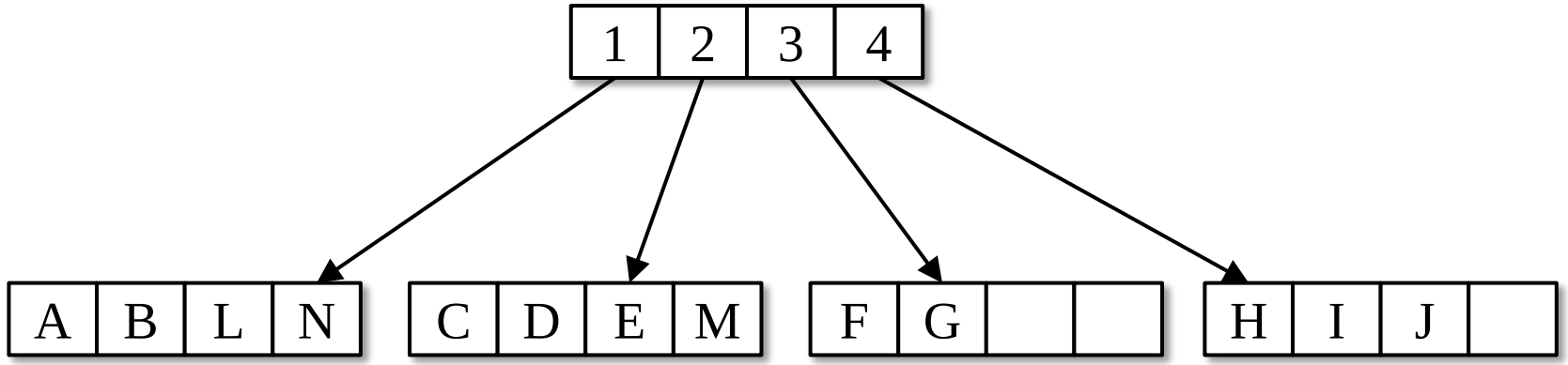
e se os dois MBRs tivessem também a mesma área??

escolhe aquele com o menor número de entradas. Se empatar novamente, escolhe qualquer um deles.

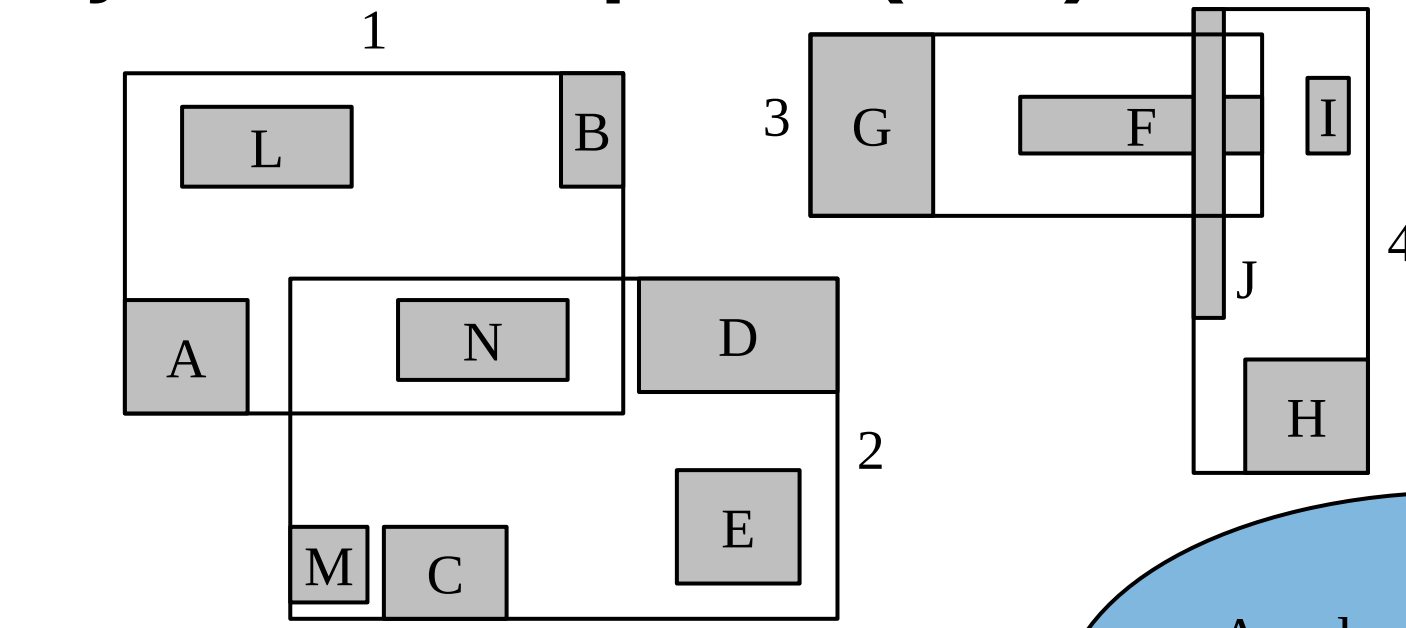


Inserção – Exemplo: R(2, 4)

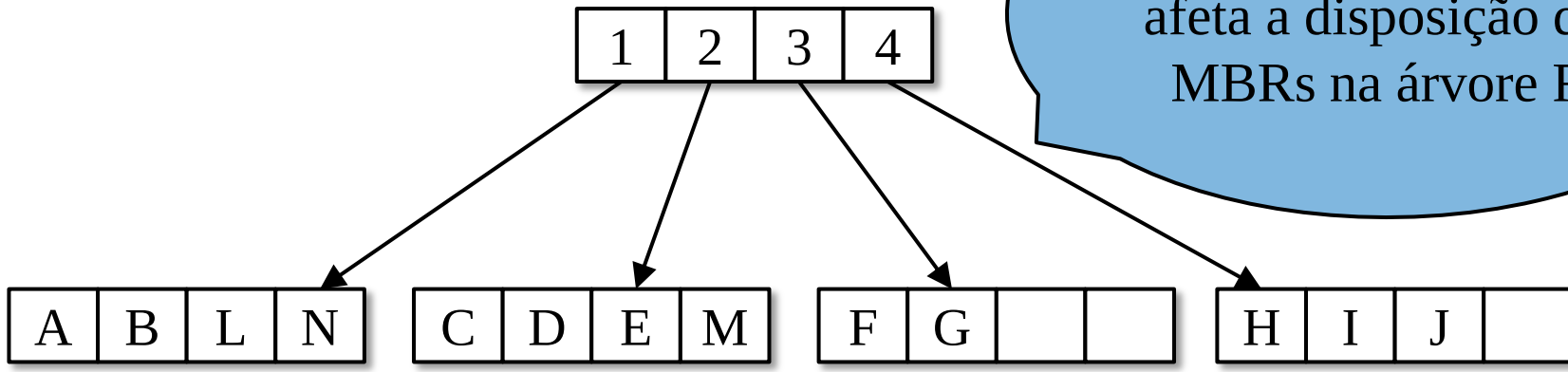




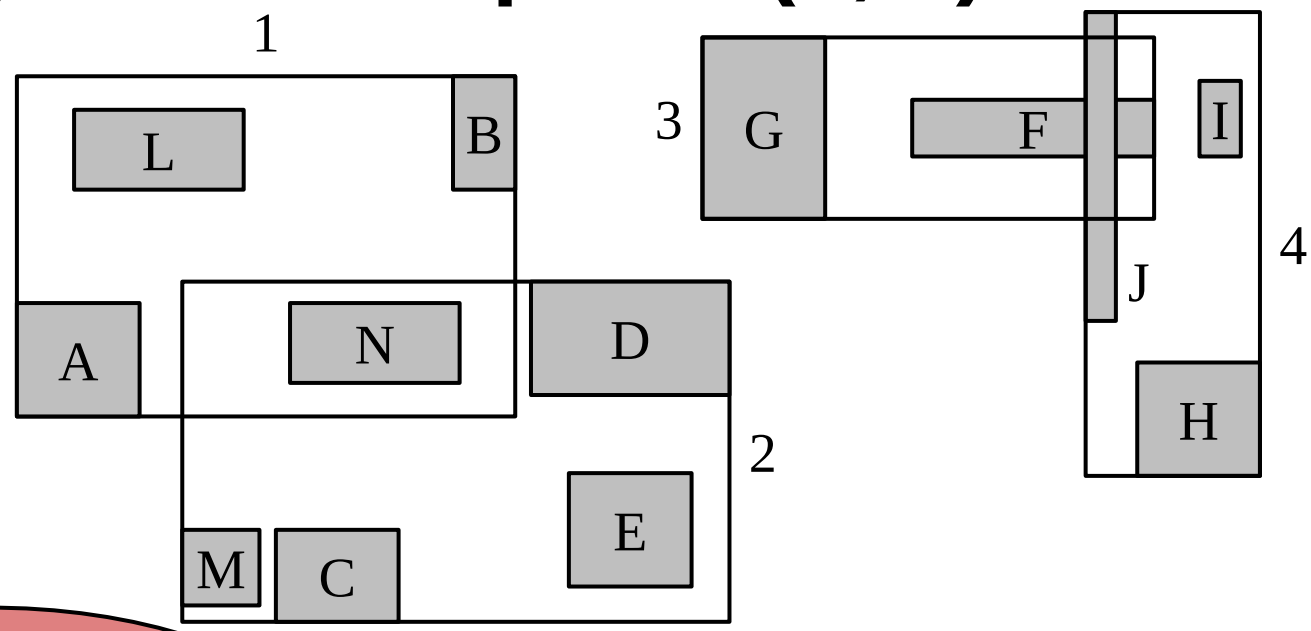
Inserção – Exemplo: R(2, 4)



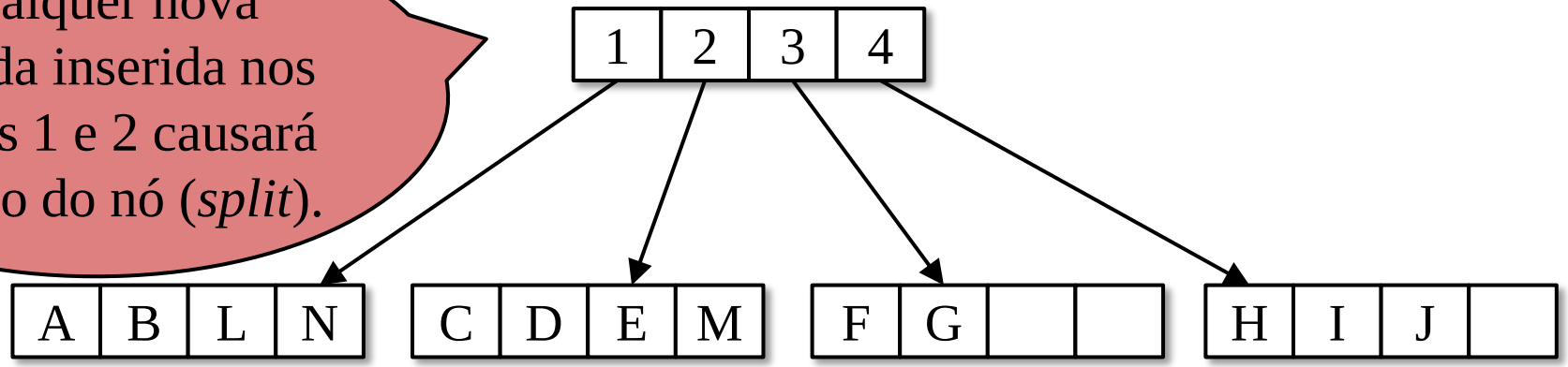
A ordem de inserção afeta a disposição dos MBRs na árvore R.



Inserção – Exemplo: R(2, 4)



Qualquer nova entrada inserida nos MBRs 1 e 2 causará divisão do nó (*split*).



Árvore R

Split

Split

- Distribui as *M* entradas de um nó mais a nova entrada em dois nós.
- Reduzir a área de cobertura.
- Seleção dos primeiros objetos de cada grupo: *seeds*. Na árvore R, dois objetos são promovidos ao nó índice.
- Distribuição dos objetos restantes.
- Algoritmos: quadrático, linear, exaustivo.

Split: Algoritmo Quadrático

Parte 1: Seleção das seeds.

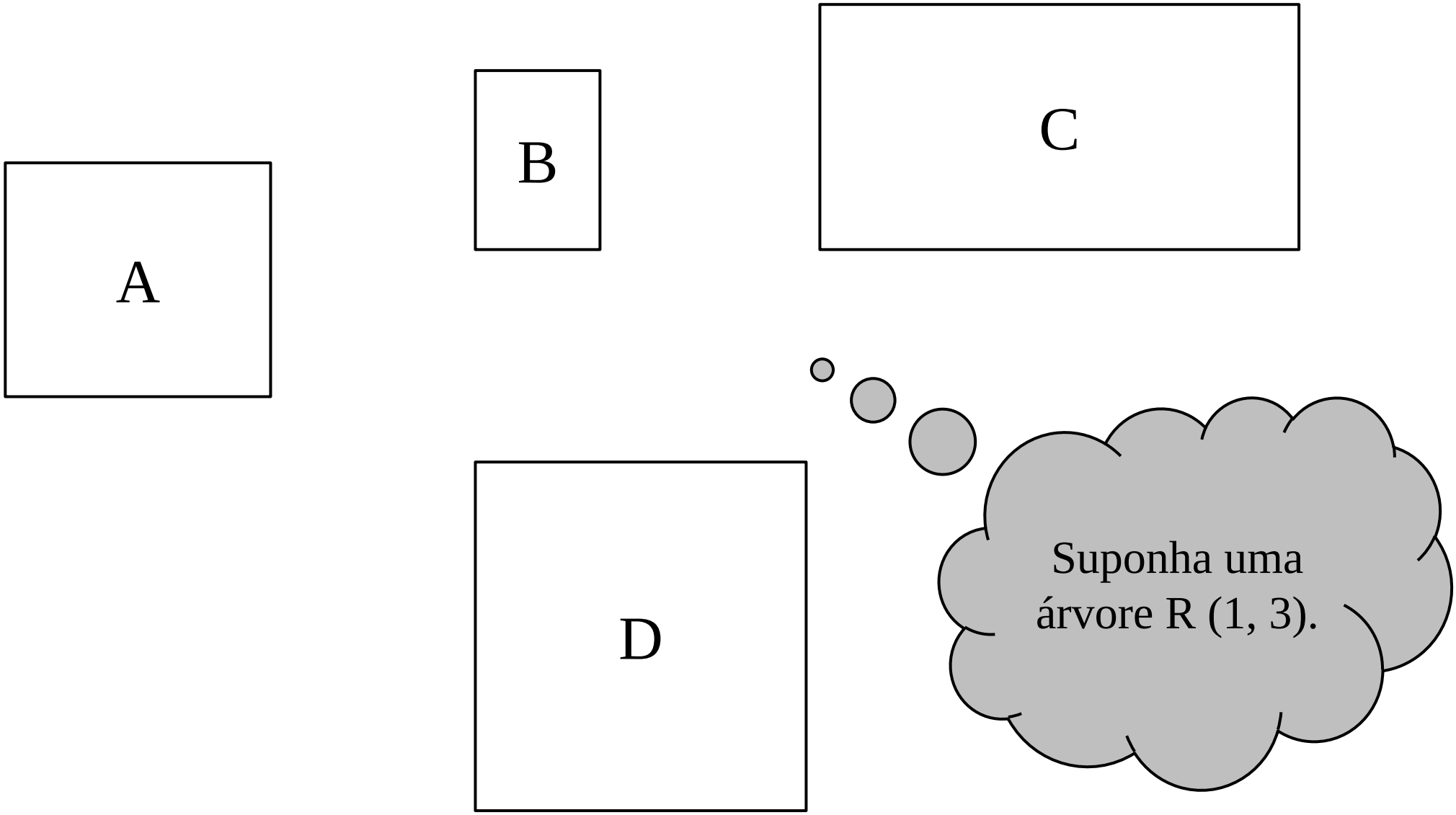
- Selecionar dois objetos como *seeds* de modo que esses objetos, se colocados juntos, criam o maior *dead space* possível. *Dead space* é a área restante no MBR se as áreas das *seeds* forem ignoradas. Complexidade de tempo: $O(M^2)$

Parte 2: Redistribuição das M – 1 entradas.

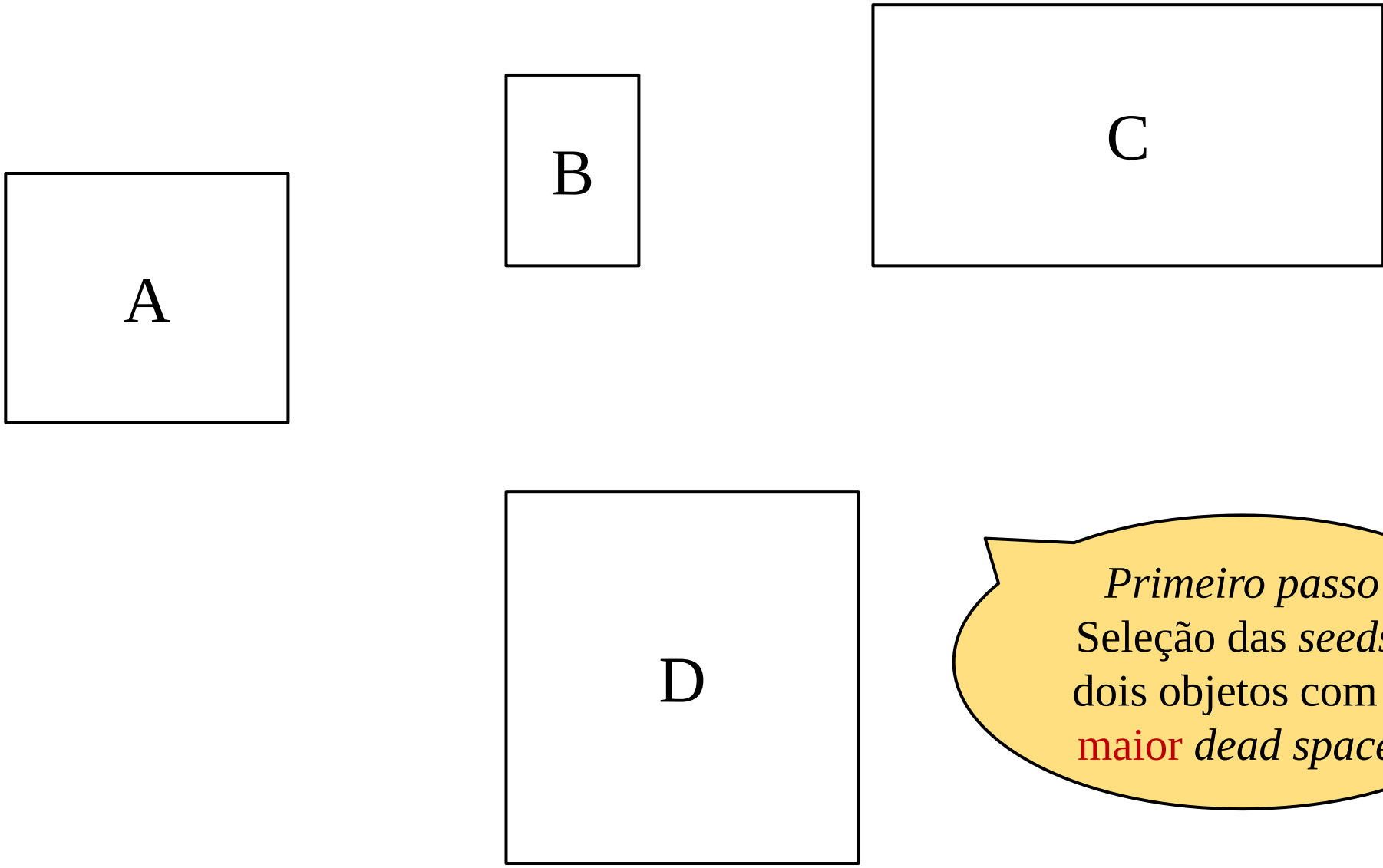
- Até que não reste mais entradas ($O(M)$), selecionar a entrada *E* cuja diferença de *dead space* para cada um dos dois nós *N1* e *N2* seja máxima ($O(M^2)$).
- Inserir *E* no nó que requer o menor aumento de seu MBR ($O(1)$).

Complexidade total de tempo: $O(M^2 + M + M^2 + 1) = O(M^2)$

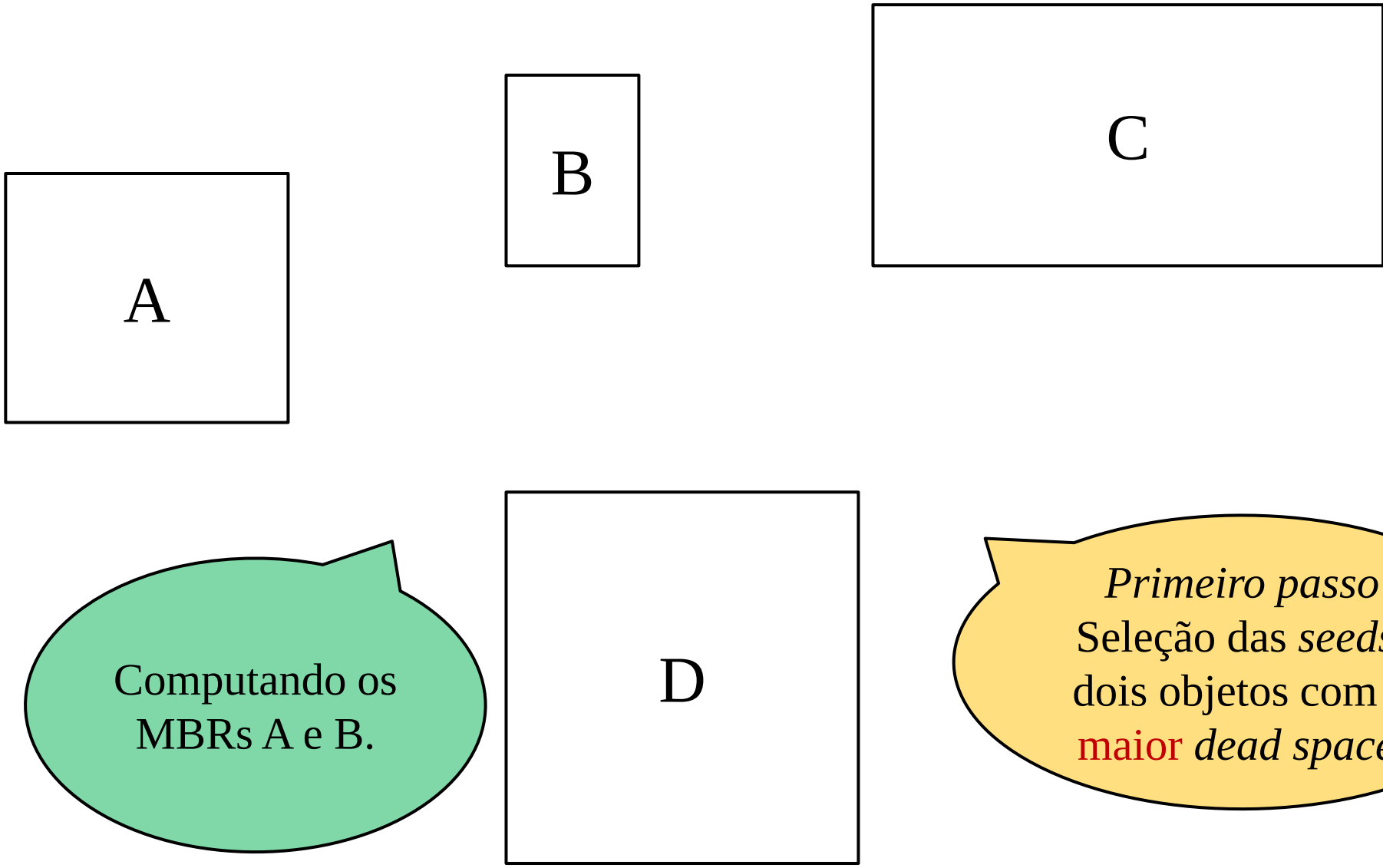
Split: Algoritmo Quadrático – Exemplo



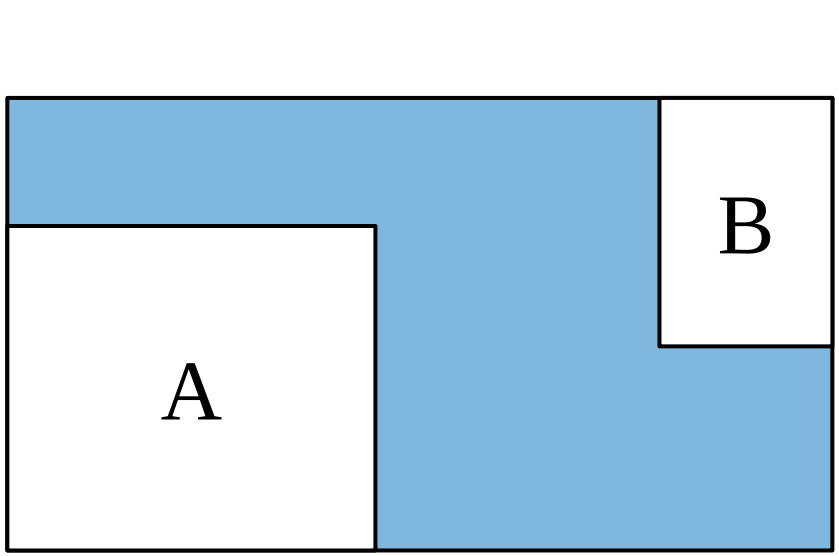
Split: Algoritmo Quadrático – Exemplo



Split: Algoritmo Quadrático – Exemplo



Split: Algoritmo Quadrático – Exemplo



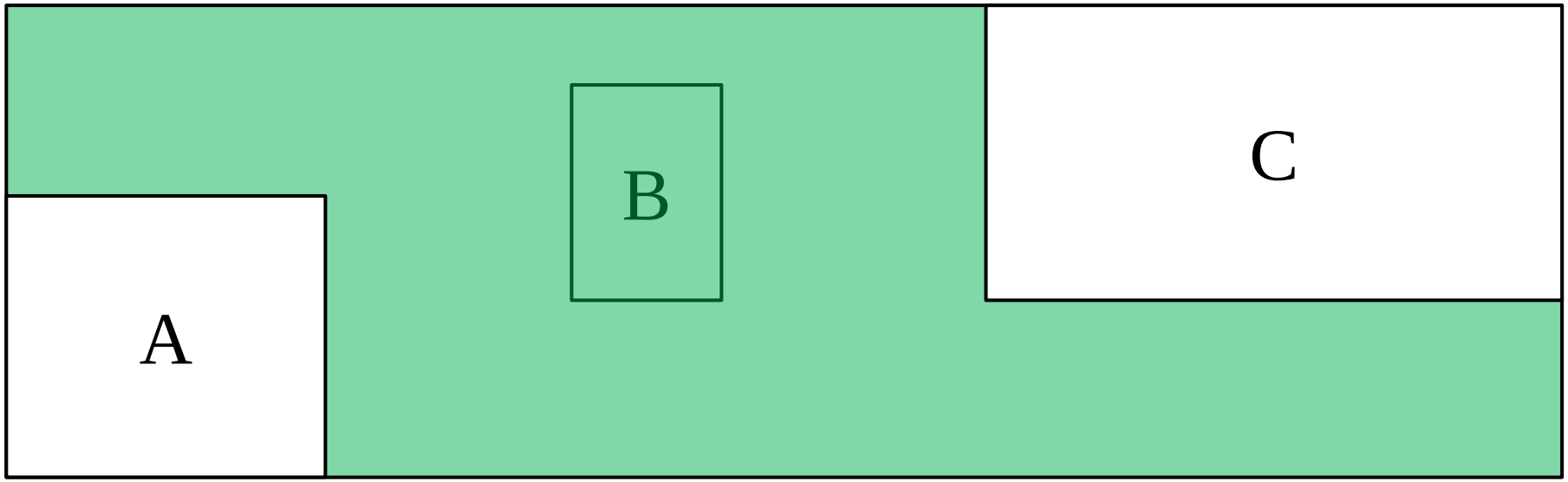
C

Dead space (A, B) = Área(MBR(A, B)) – Área(A) – Área(B)

Computando os MBRs A e B.

D

Split: Algoritmo Quadrático – Exemplo

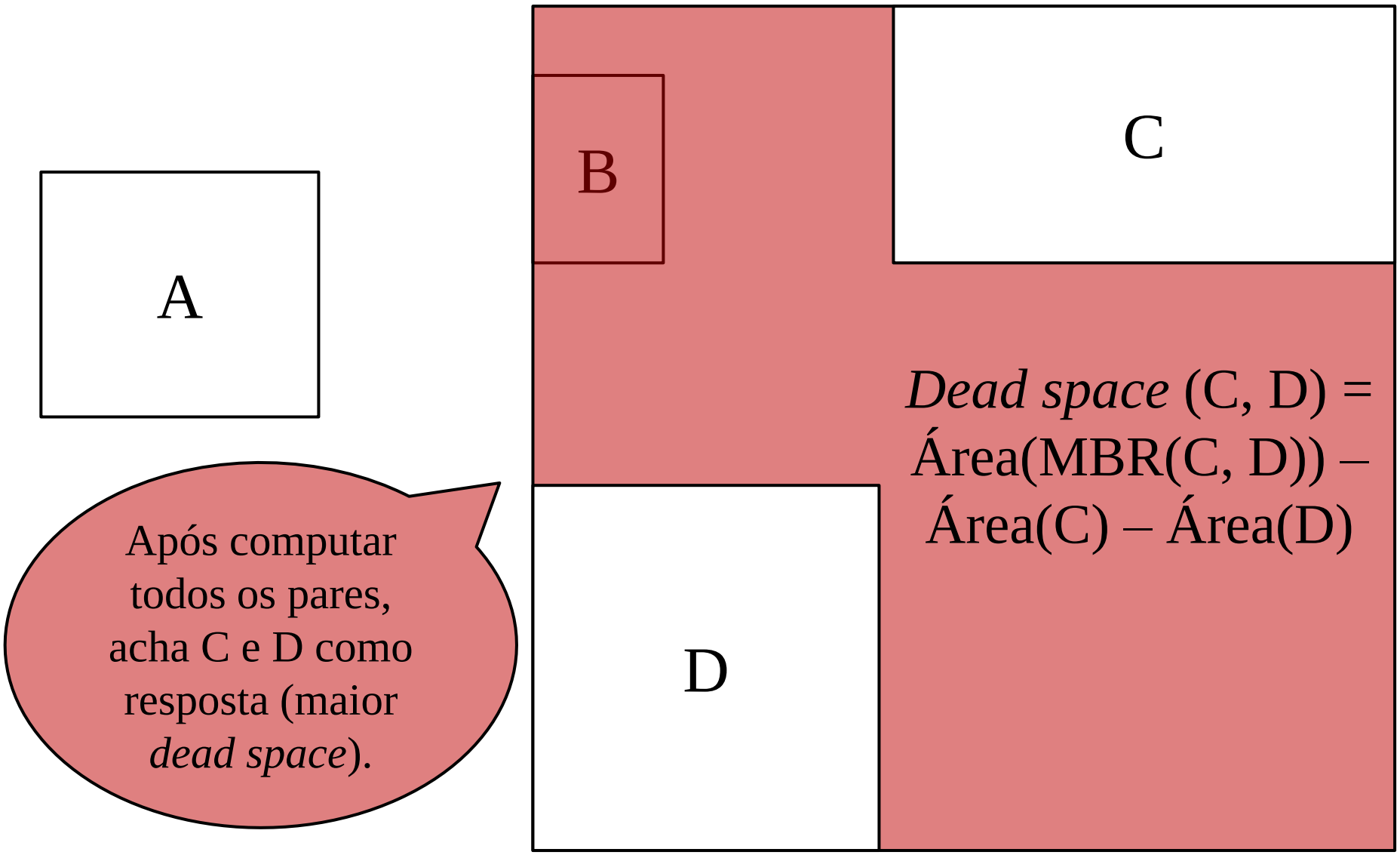


Computando os MBRs A e C.

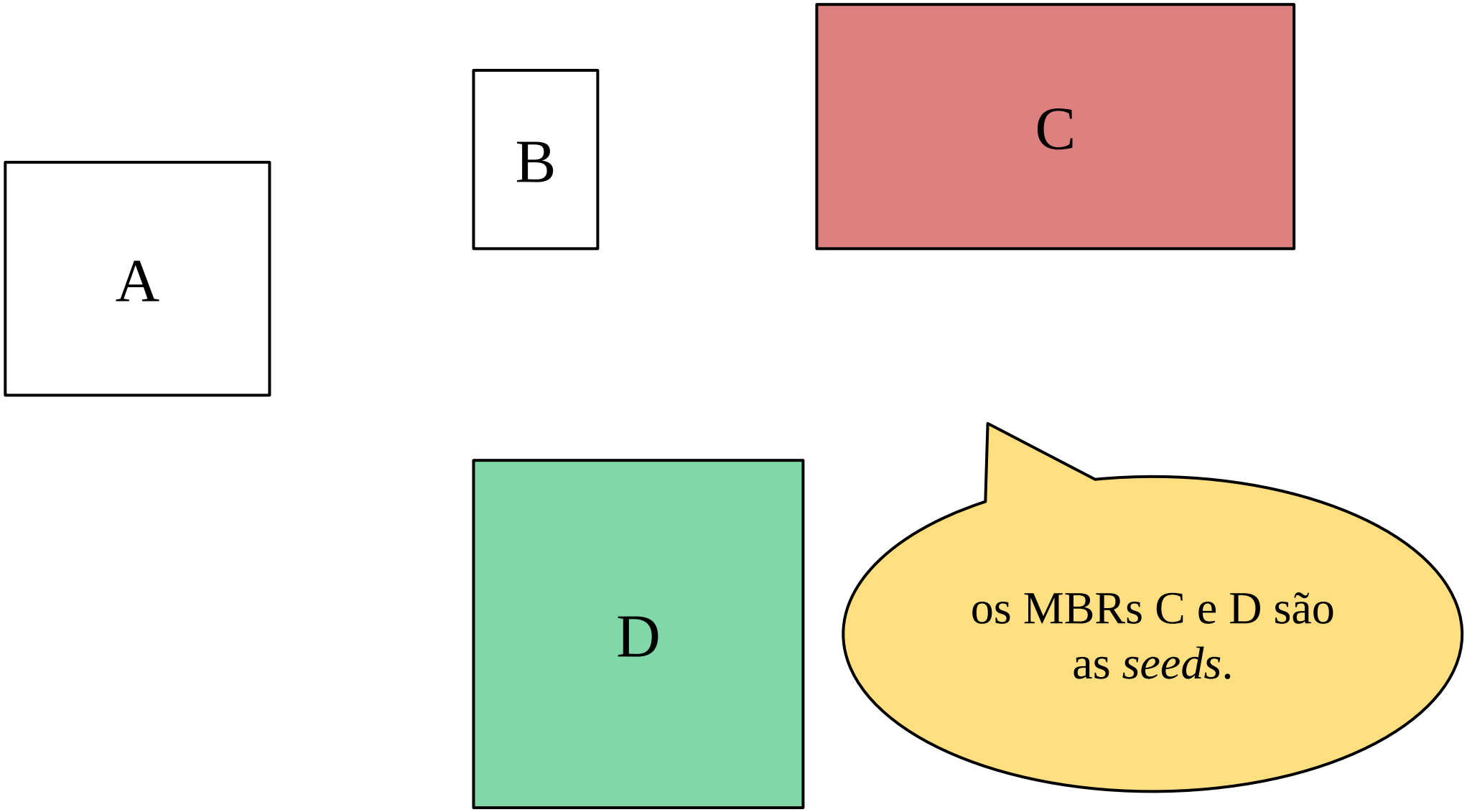
D

Dead space (A, C) = Área(MBR(A, C)) – Área(A) – Área(C)

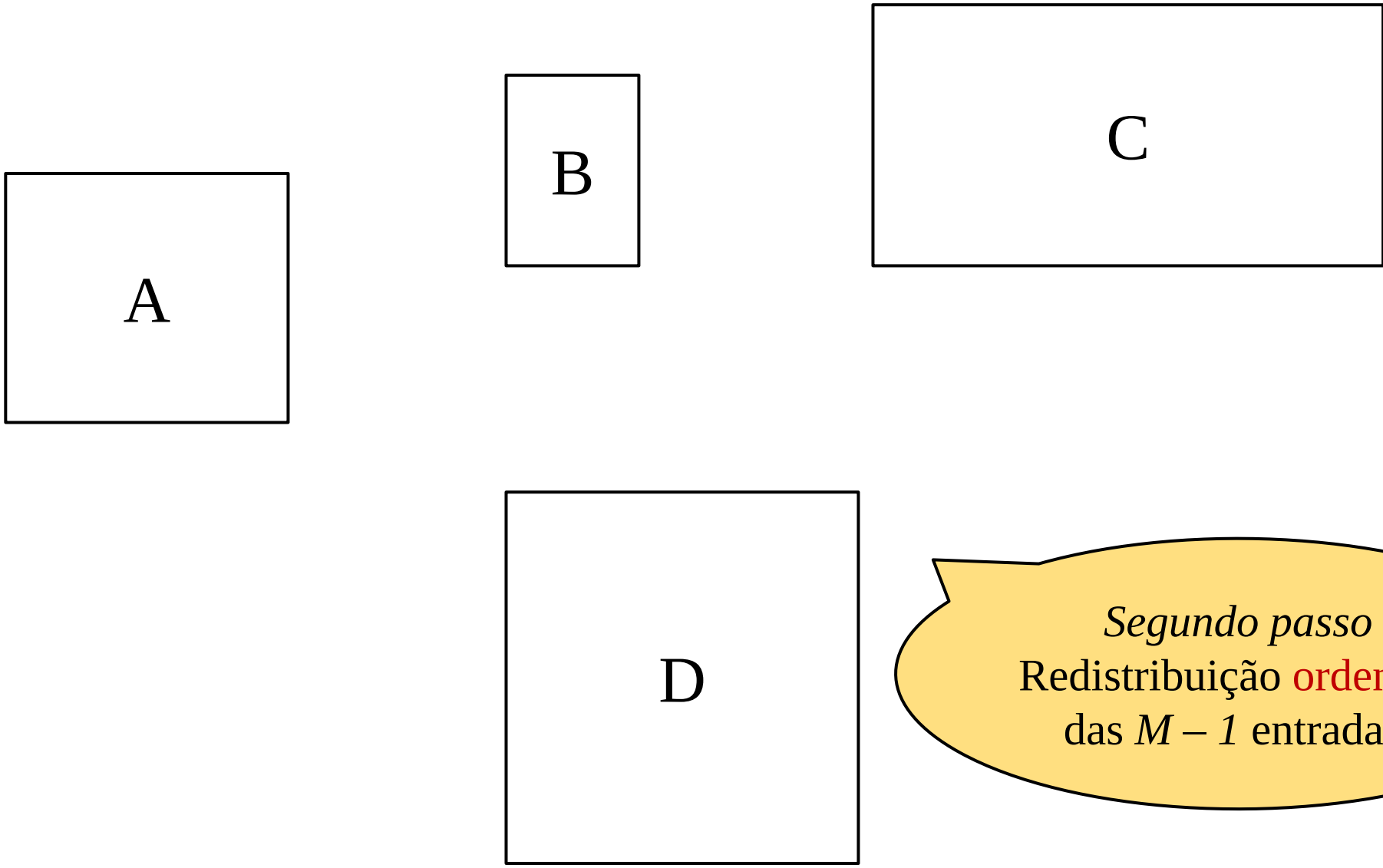
Split: Algoritmo Quadrático – Exemplo



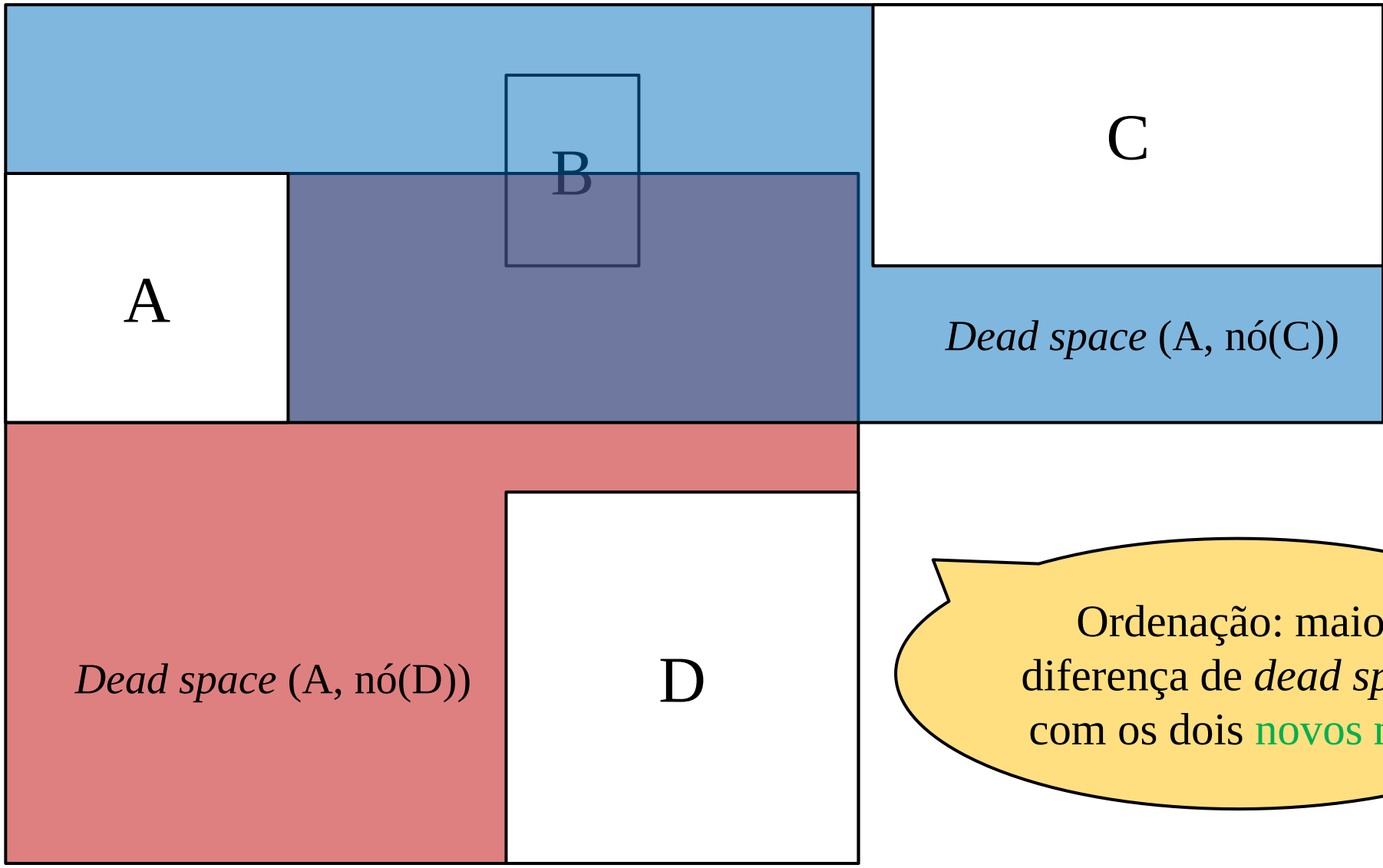
Split: Algoritmo Quadrático – Exemplo



Split: Algoritmo Quadrático – Exemplo

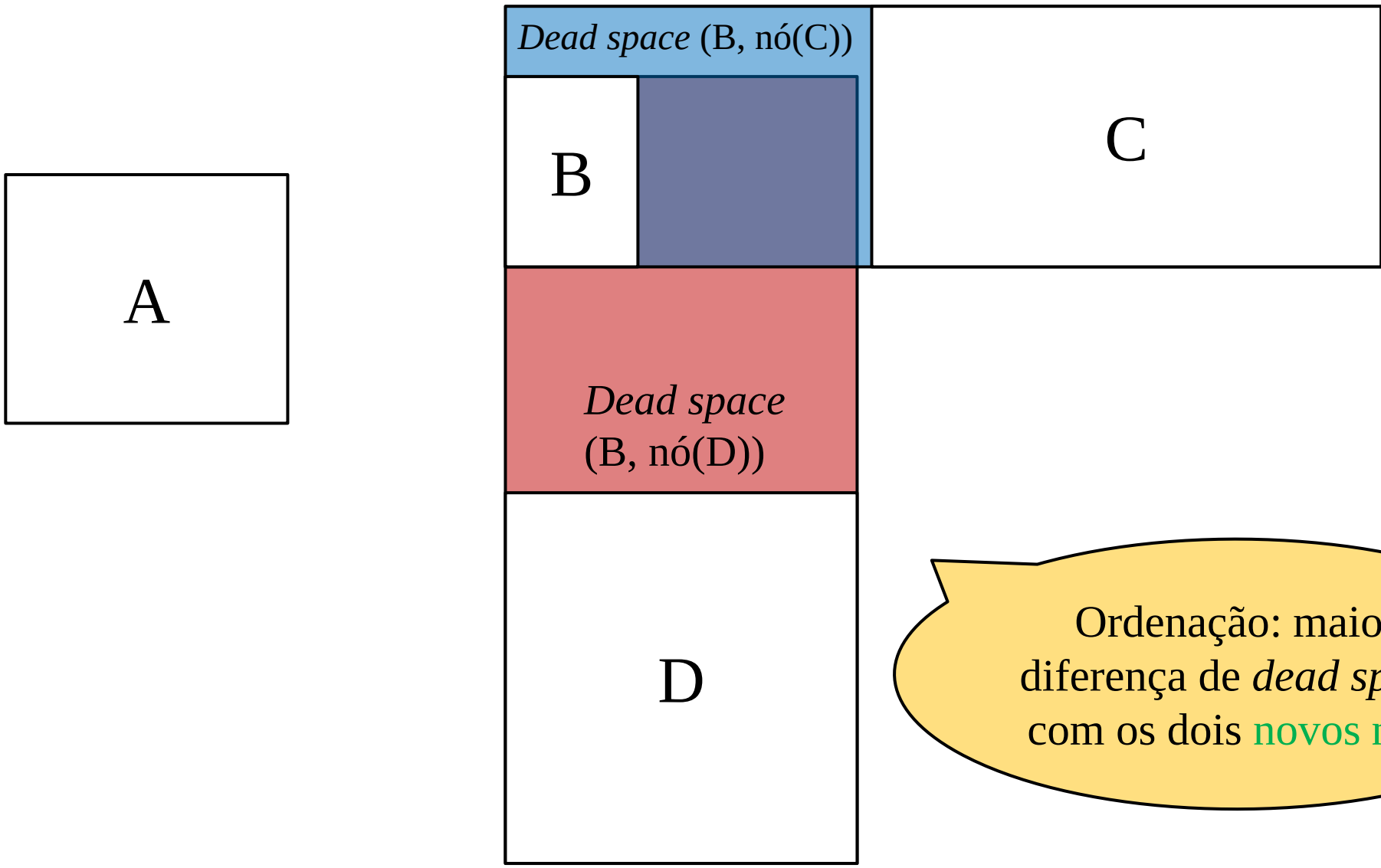


Split: Algoritmo Quadrático – Exemplo

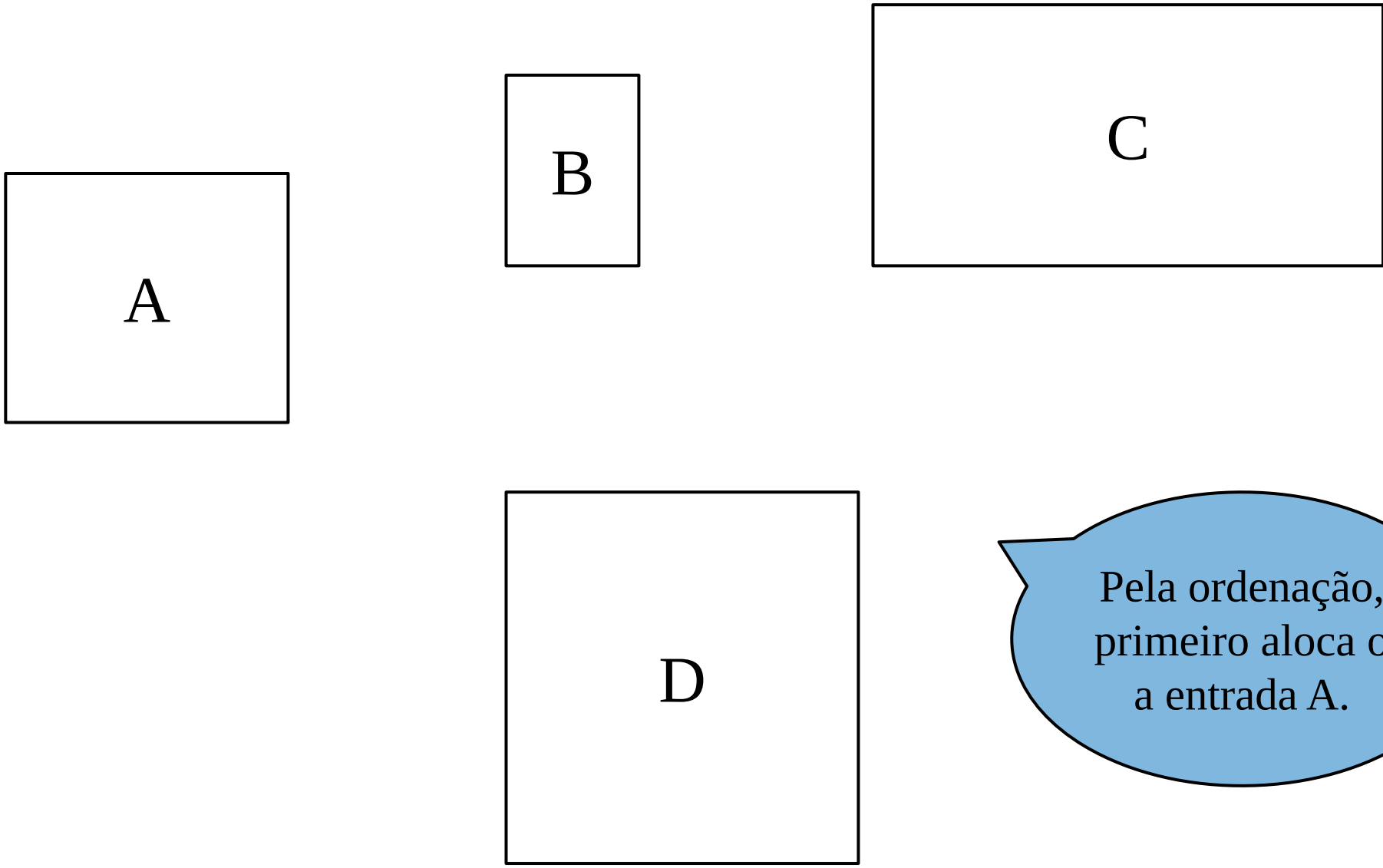


Ordenação: maior diferença de *dead space* com os dois novos nós.

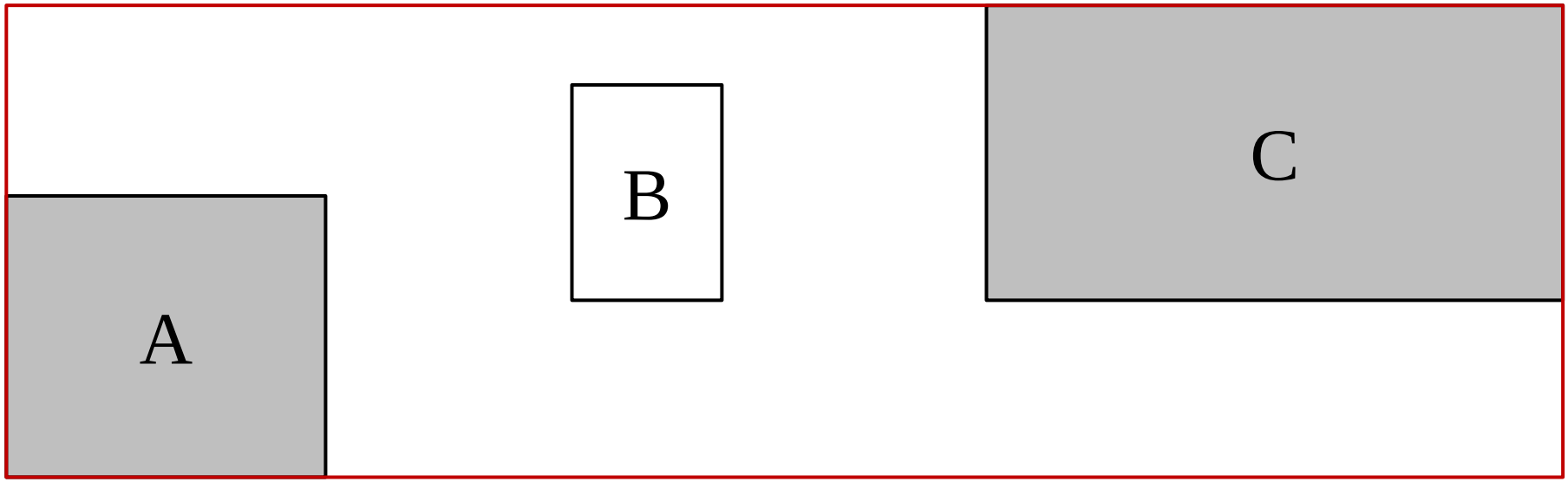
Split: Algoritmo Quadrático – Exemplo



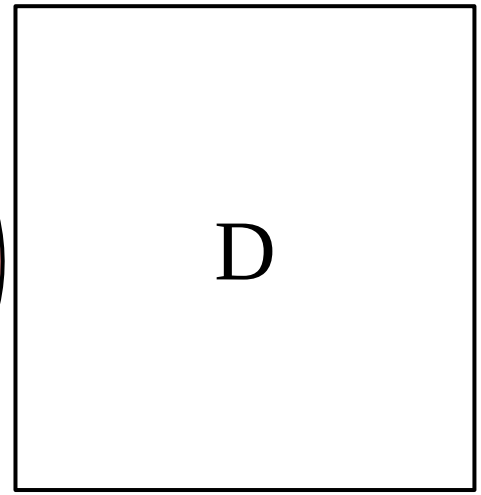
Split: Algoritmo Quadrático – Exemplo



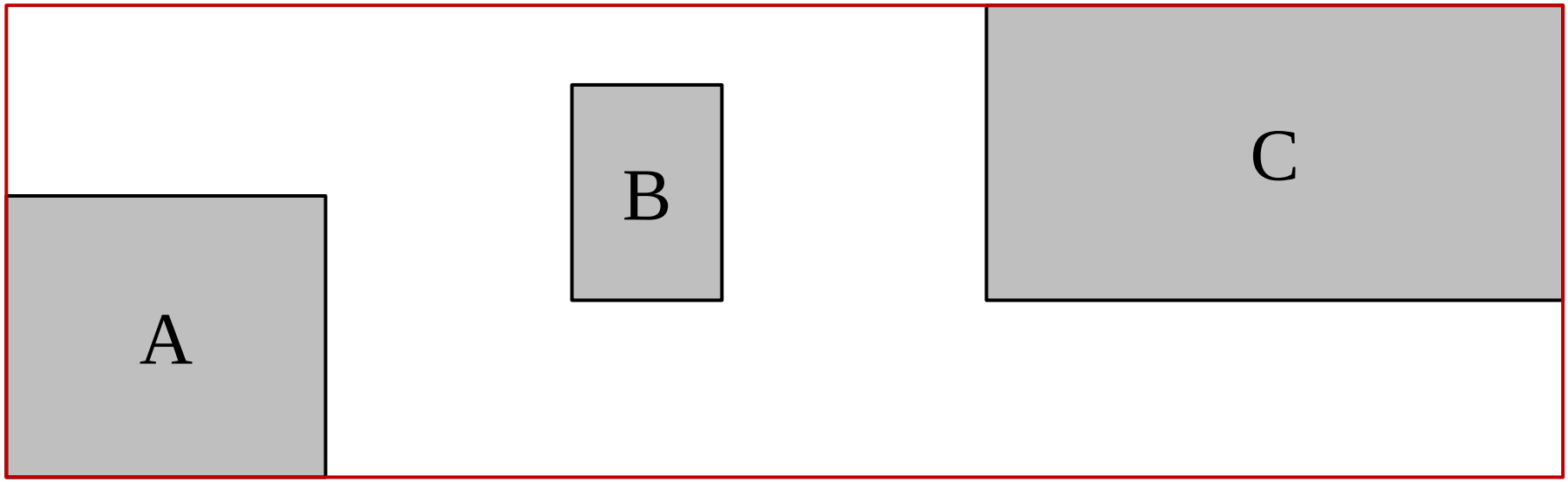
Split: Algoritmo Quadrático – Exemplo

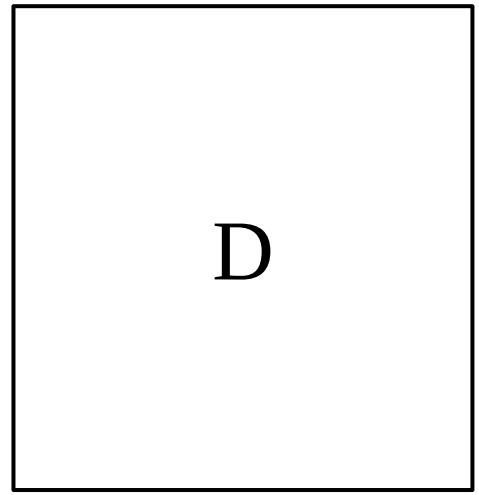


entre os nós de C e D, A vai para o nó de C, que é o que sofre menor aumento.



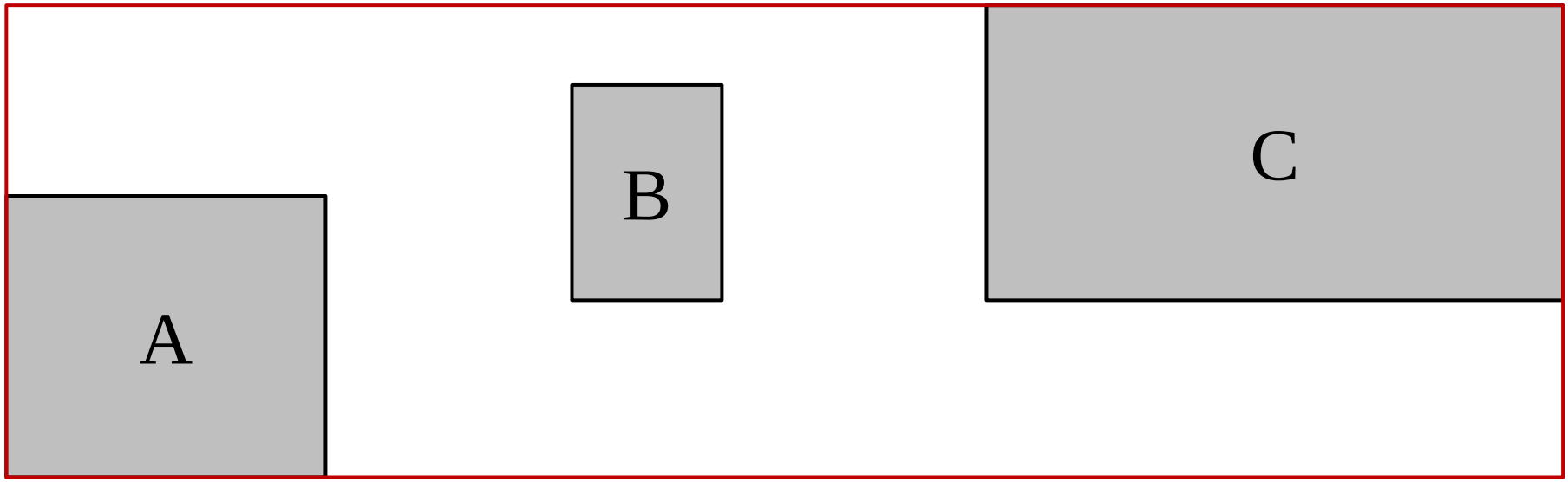
Split: Algoritmo Quadrático – Exemplo



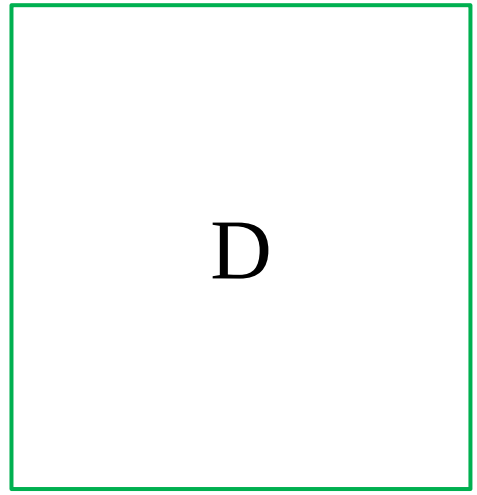


entre os nós de C e D, B também vai para o nó de C, que é o que sofre menor aumento (zero).

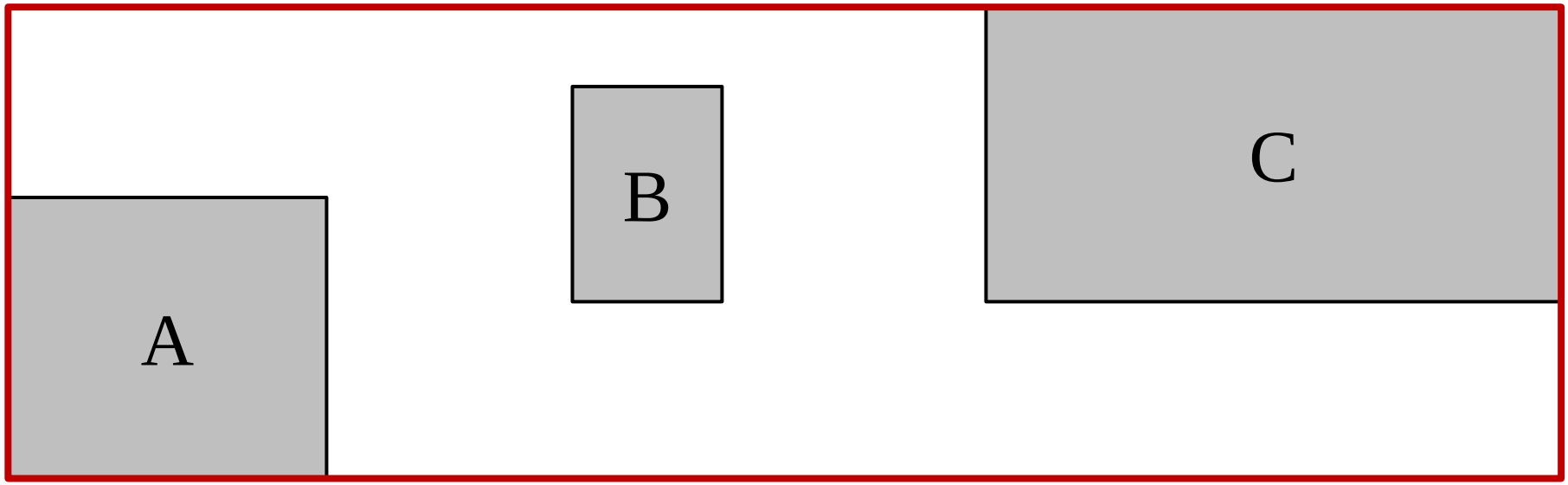
Split: Algoritmo Quadrático – Exemplo

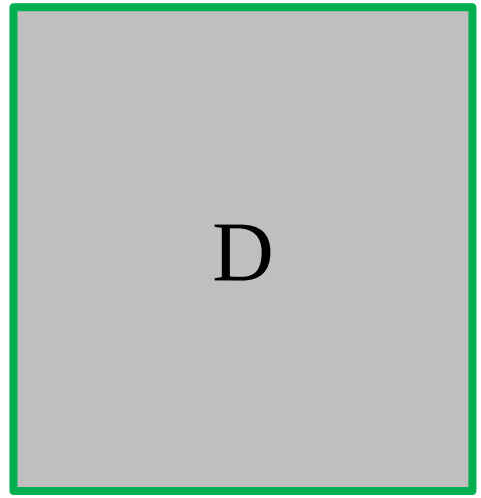


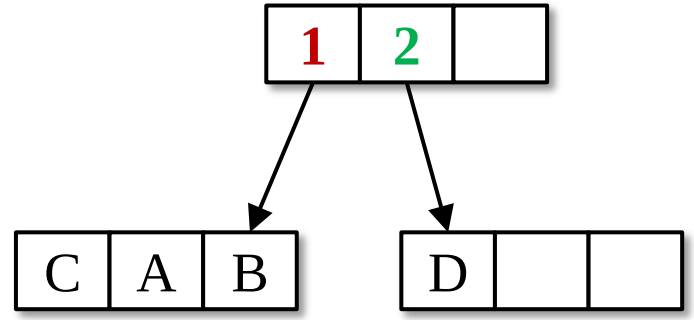
D, que é *seed*, fica sozinho em seu nó.



Split: Algoritmo Quadrático – Exemplo







Split: Algoritmo Linear

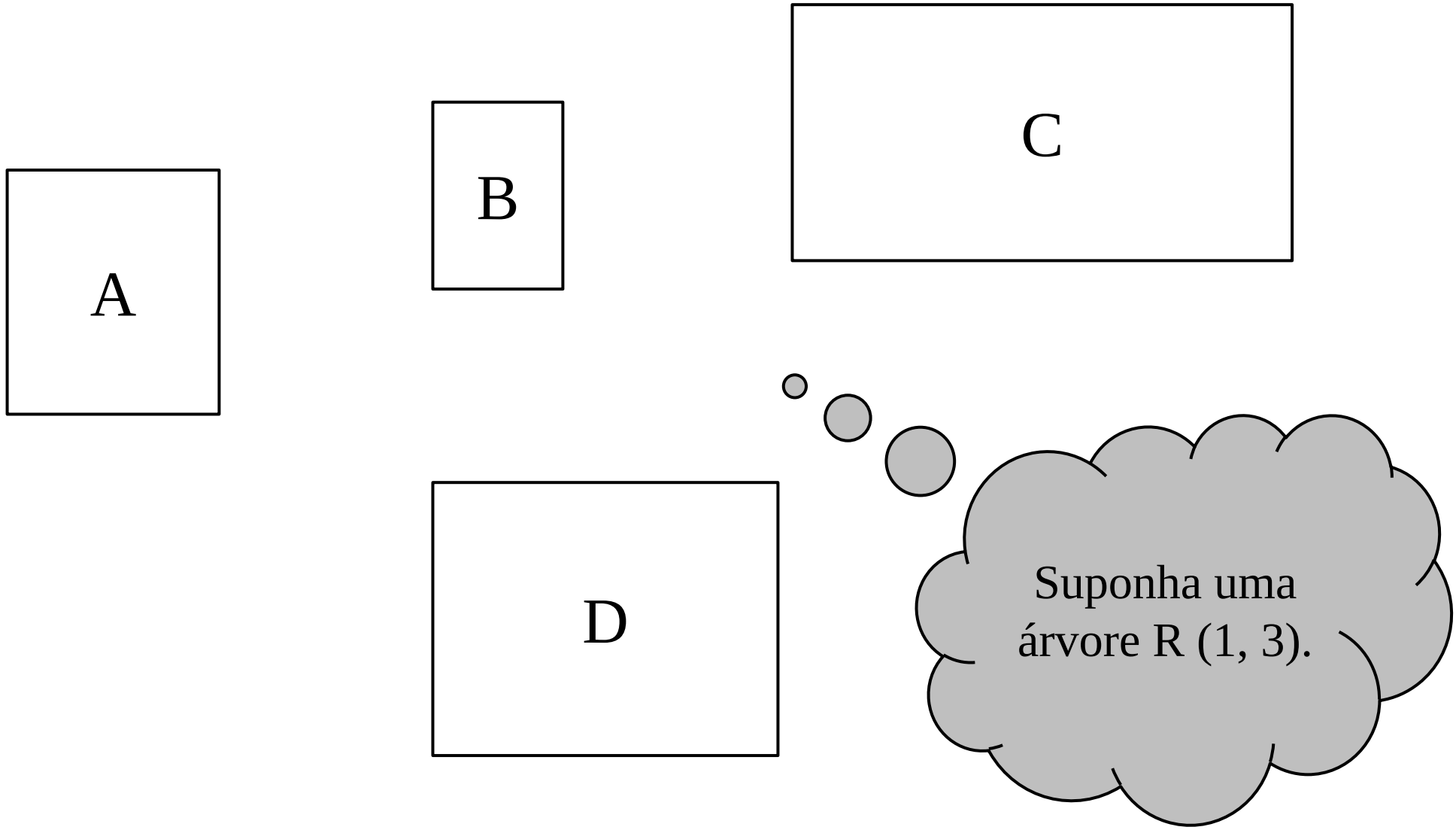
Parte 1: Seleção das seeds.

- Along each dimension, find the entry whose rectangle has the highest low side, and the one with the lowest high side. Record the separation. Complexidade de tempo: O(M * *d*) *d = nº dimensões*
- Normalize the separations by divinding the width of the entire set along the corresponding dimension (O(*d*)).
- Choose the pair with the greatest normalized separation along any dimension (O(*d*)).

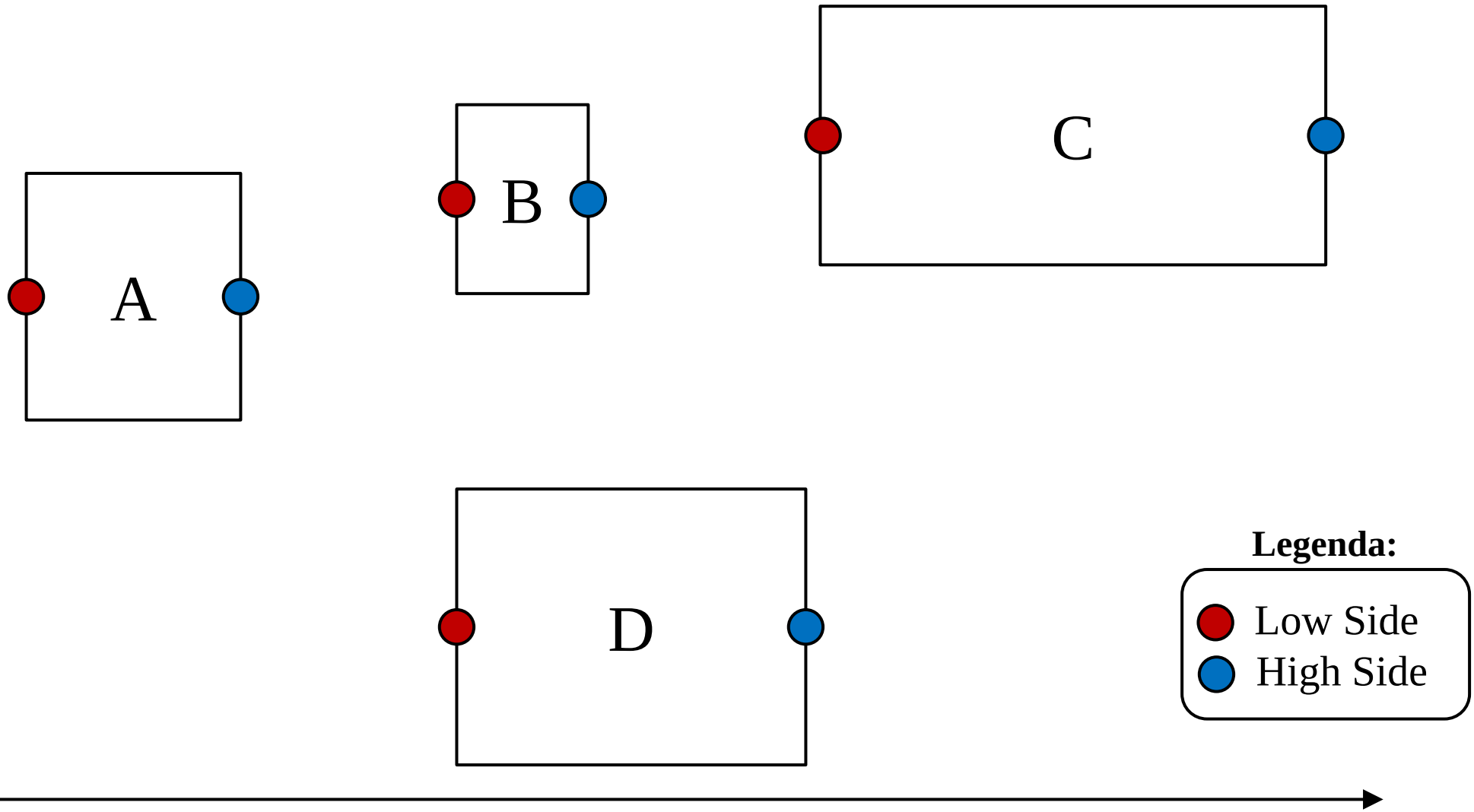
Parte 2: Redistribuição das M – 1 entradas.

- Até que não reste mais entradas (O(1)), selecionar uma entrada *E* e inserir no nó que requer o menor aumento de seu MBR (O(M)). Complexidade total de tempo: O(M * *d*)

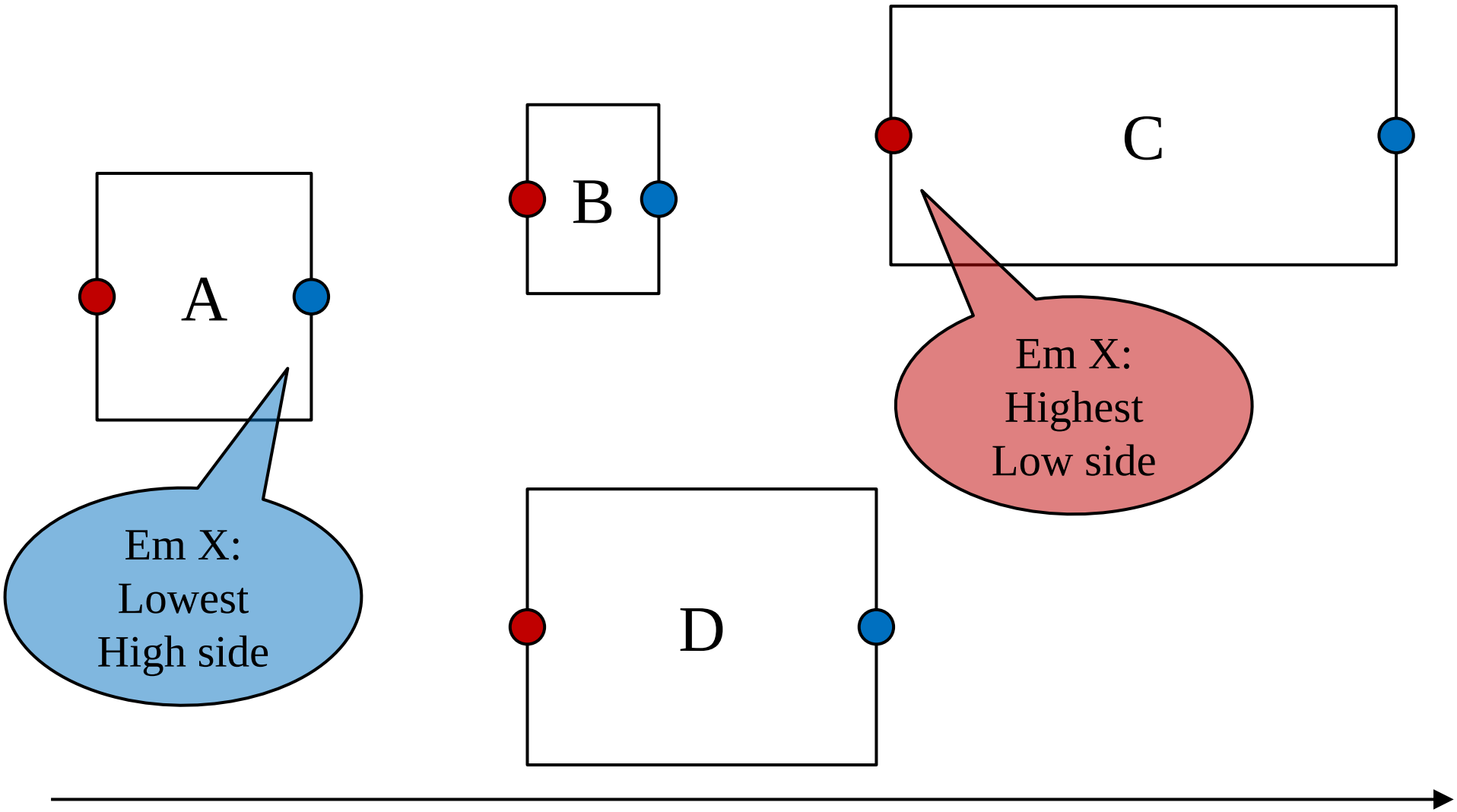
Split: Algoritmo Linear – Exemplo



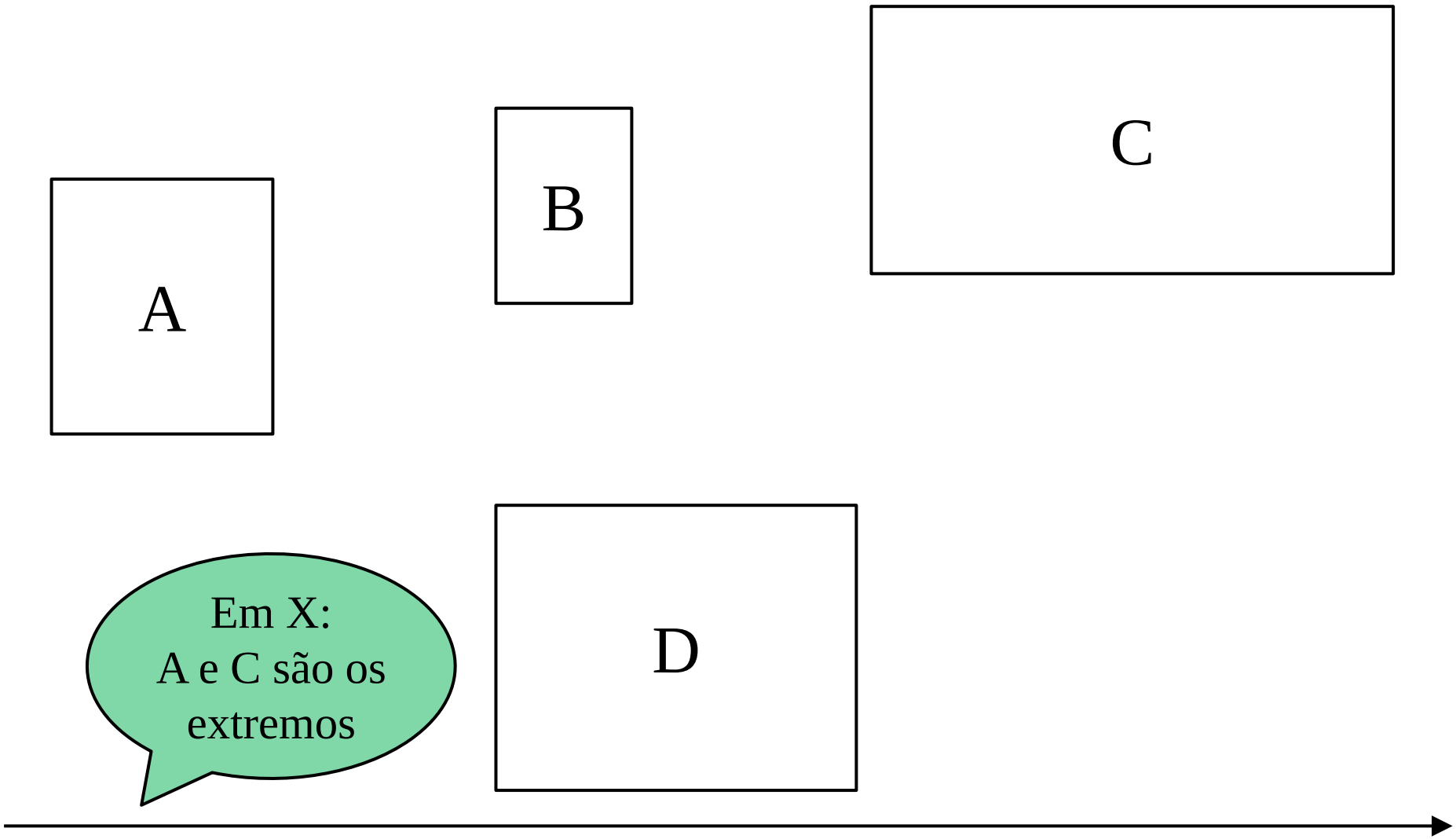
Split: Algoritmo Linear – Exemplo



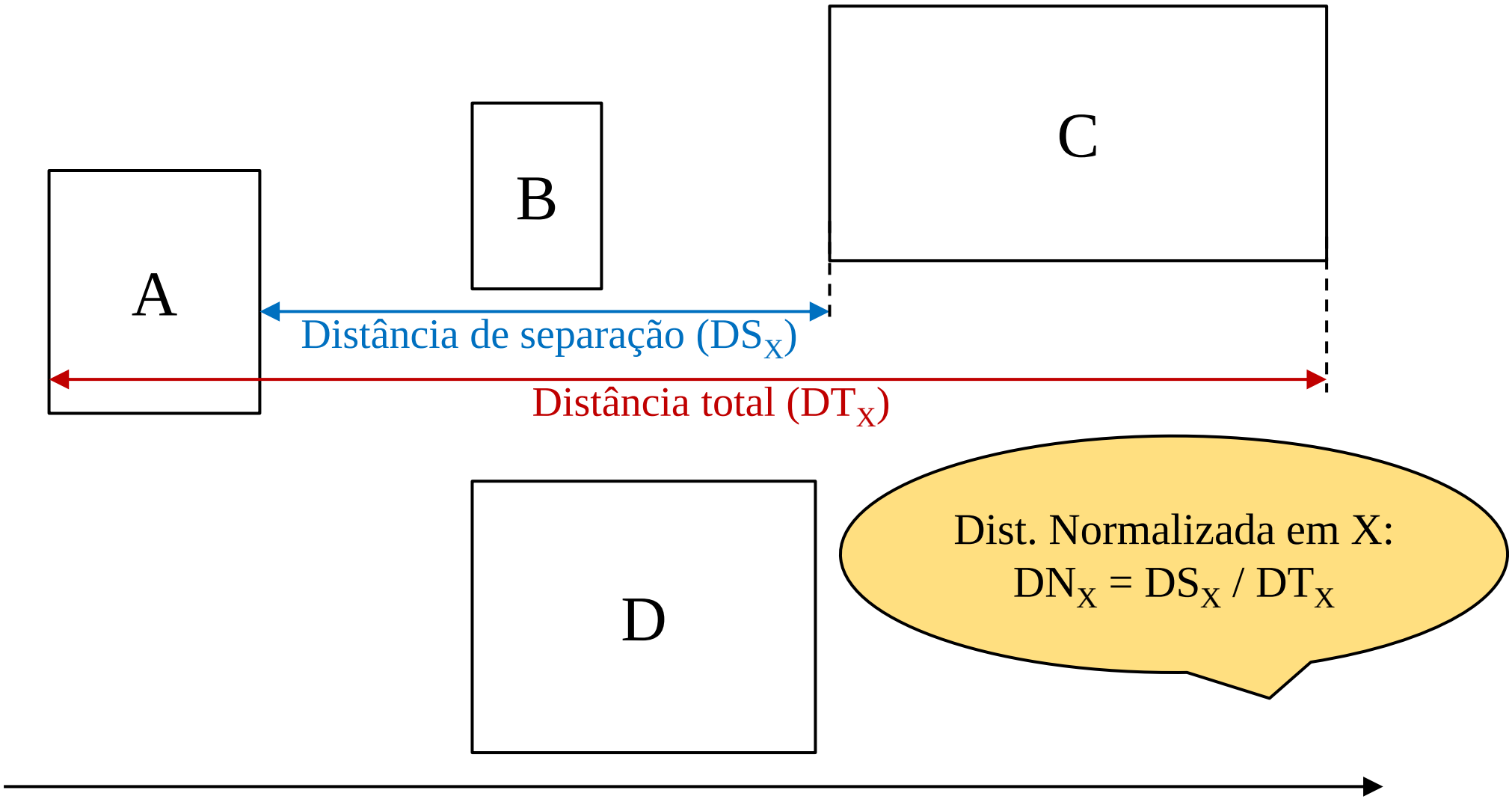
Split: Algoritmo Linear – Exemplo



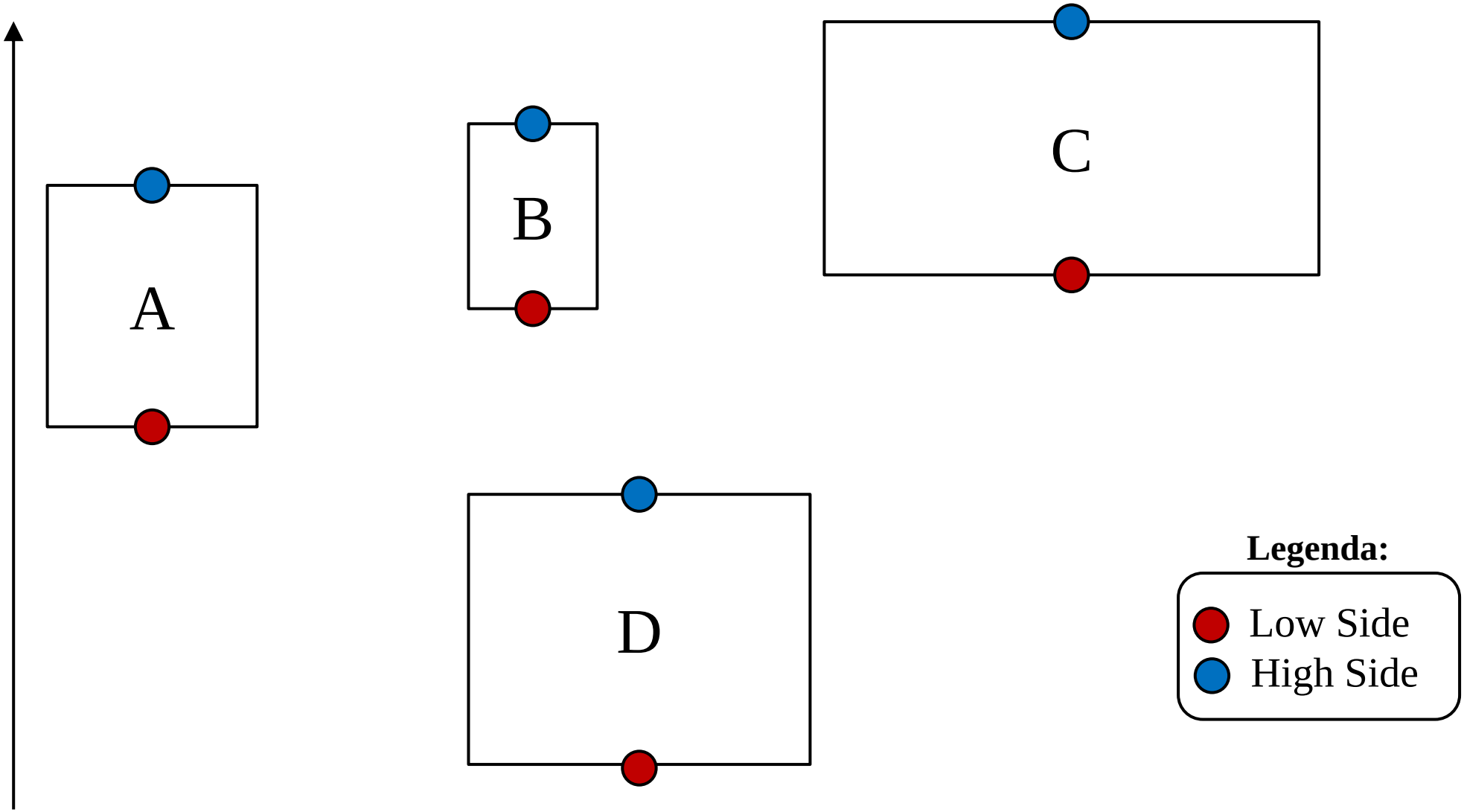
Split: Algoritmo Linear – Exemplo



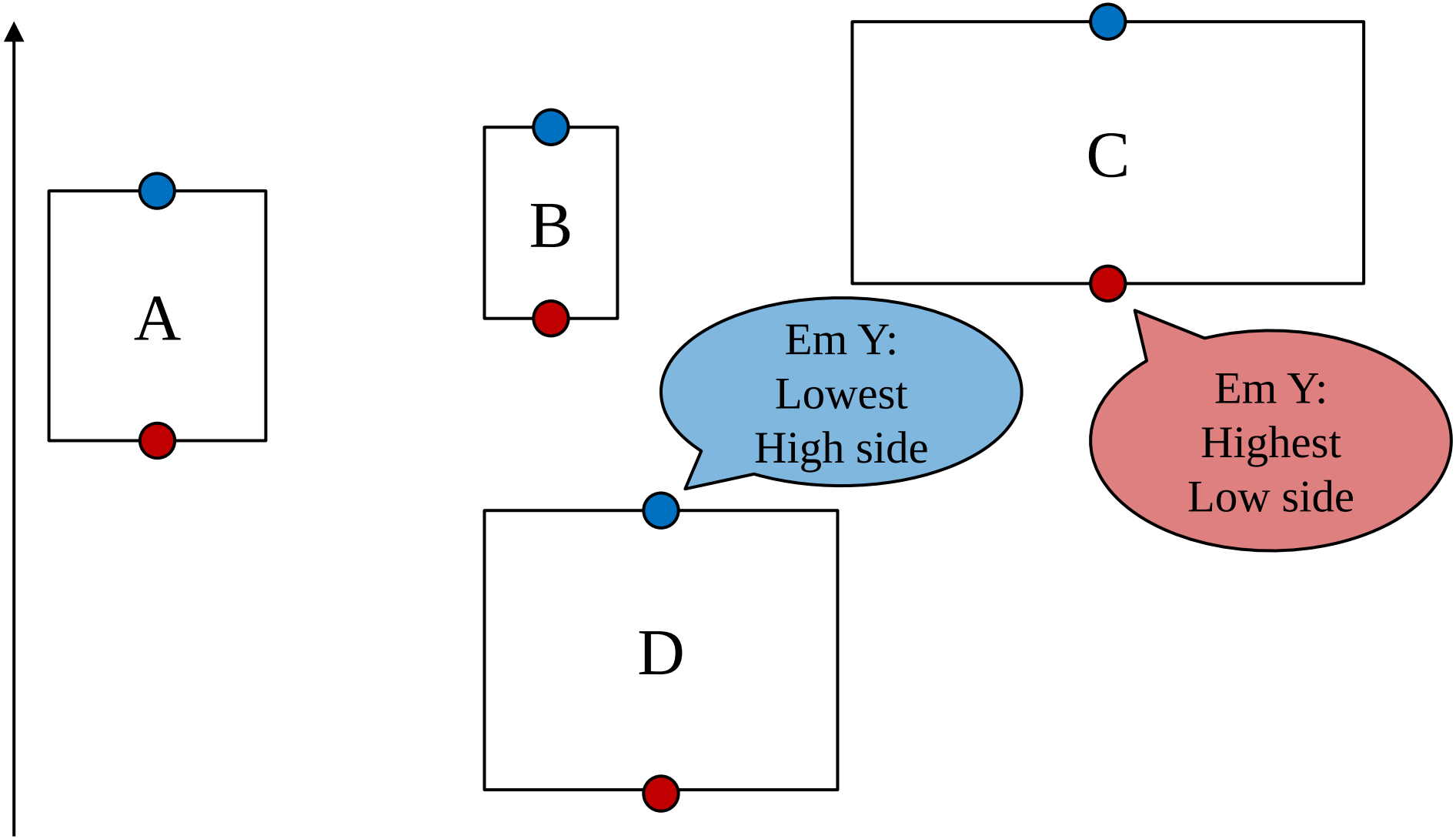
Split: Algoritmo Linear – Exemplo



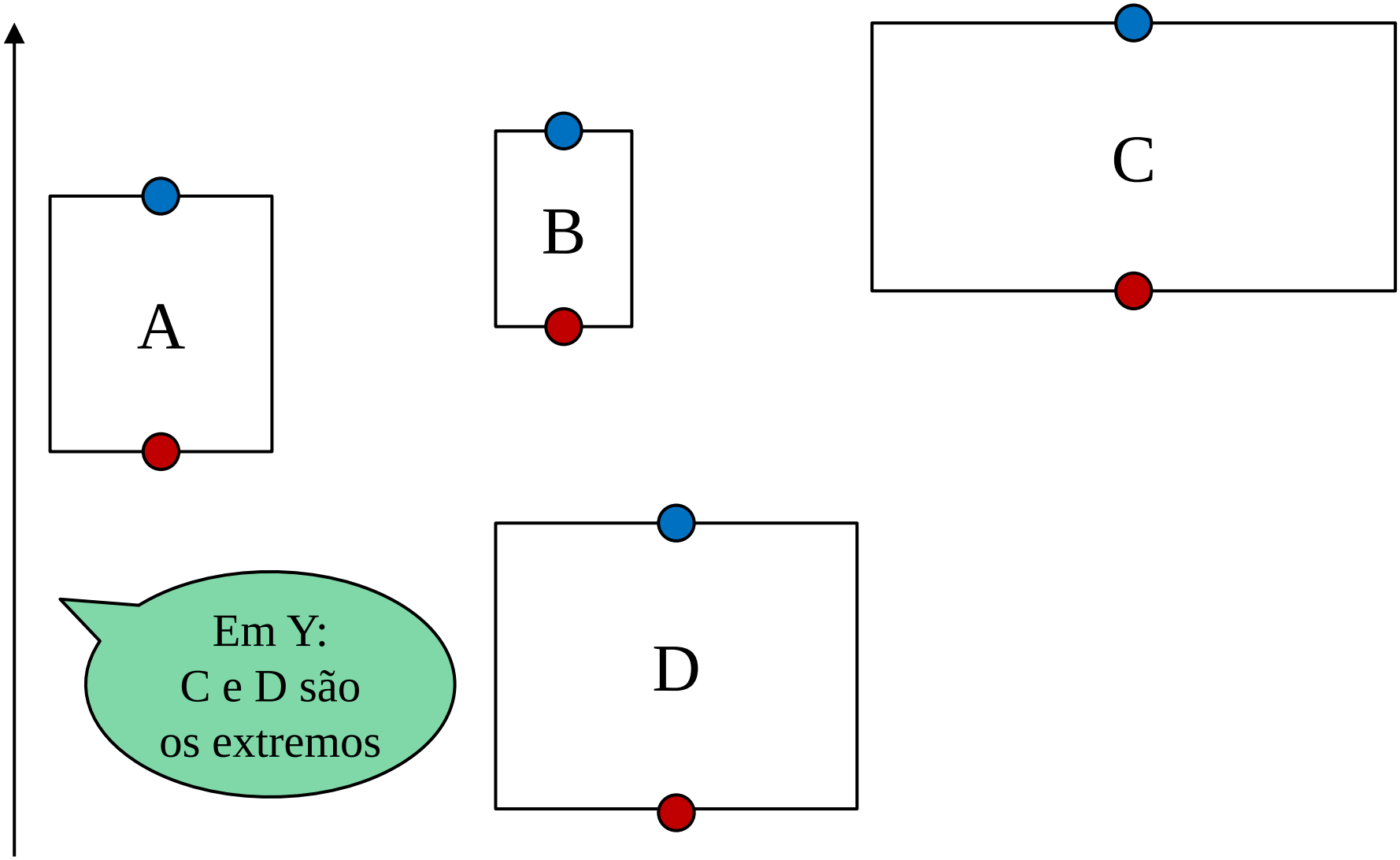
Split: Algoritmo Linear – Exemplo



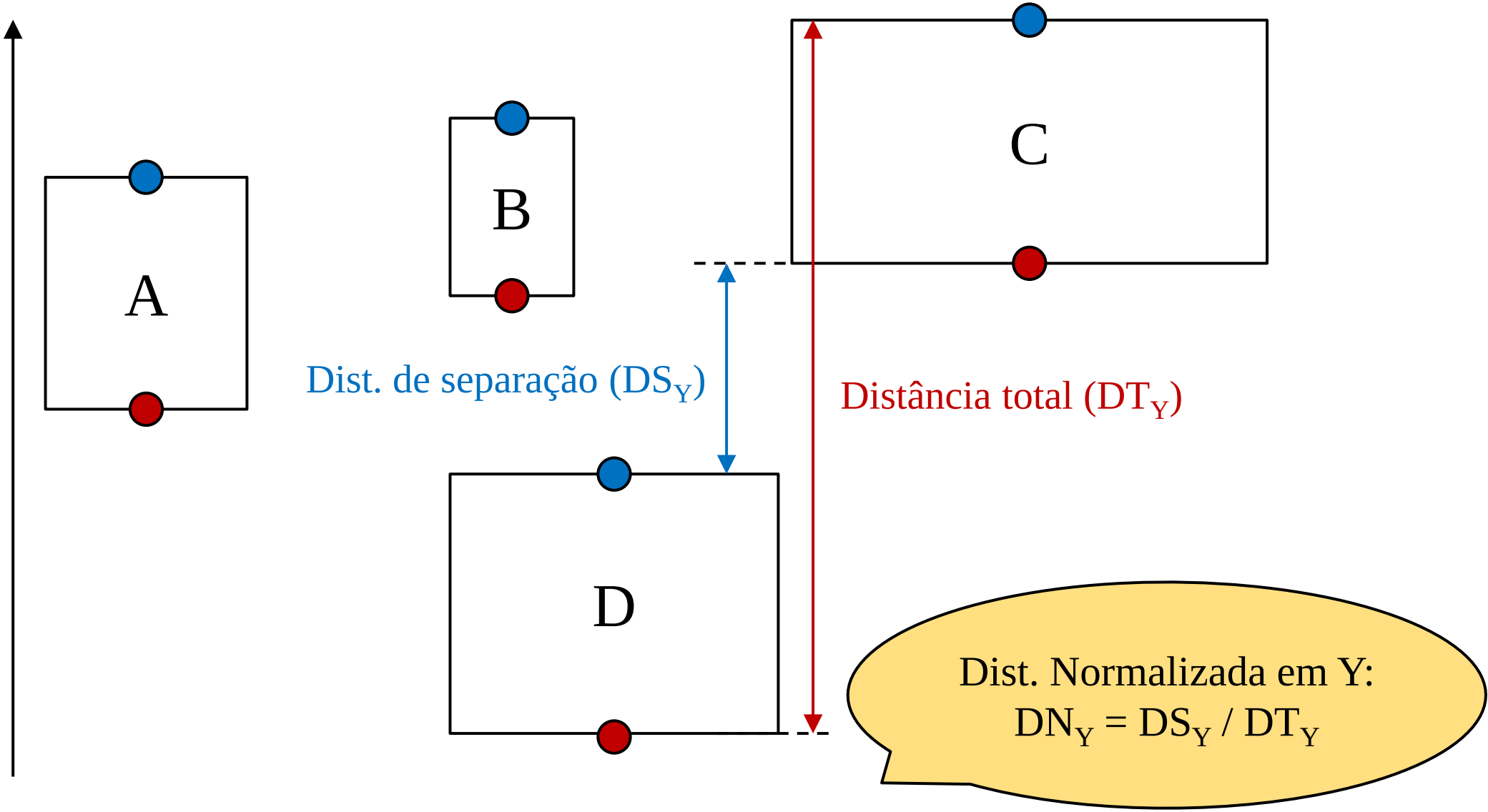
Split: Algoritmo Linear – Exemplo



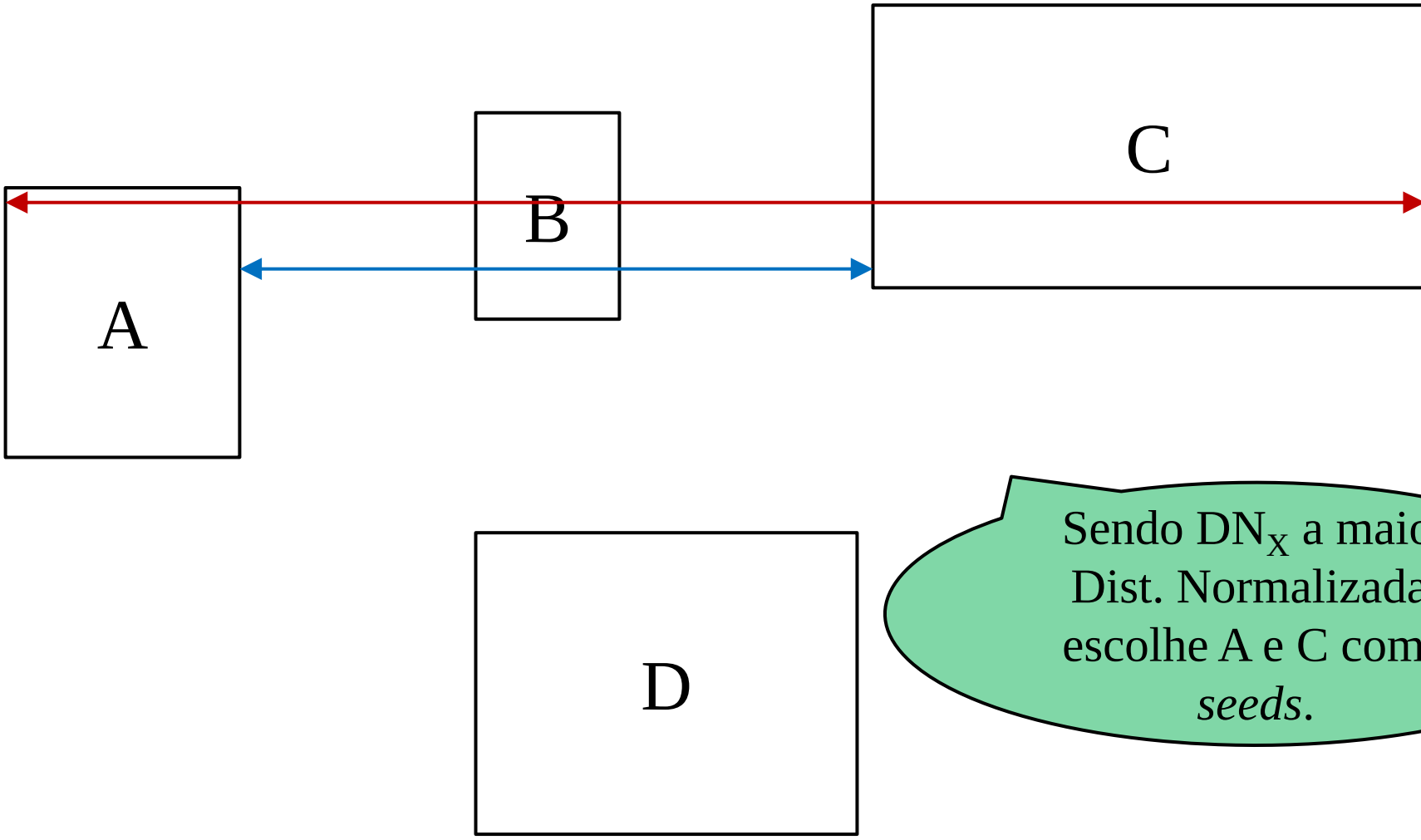
Split: Algoritmo Linear – Exemplo



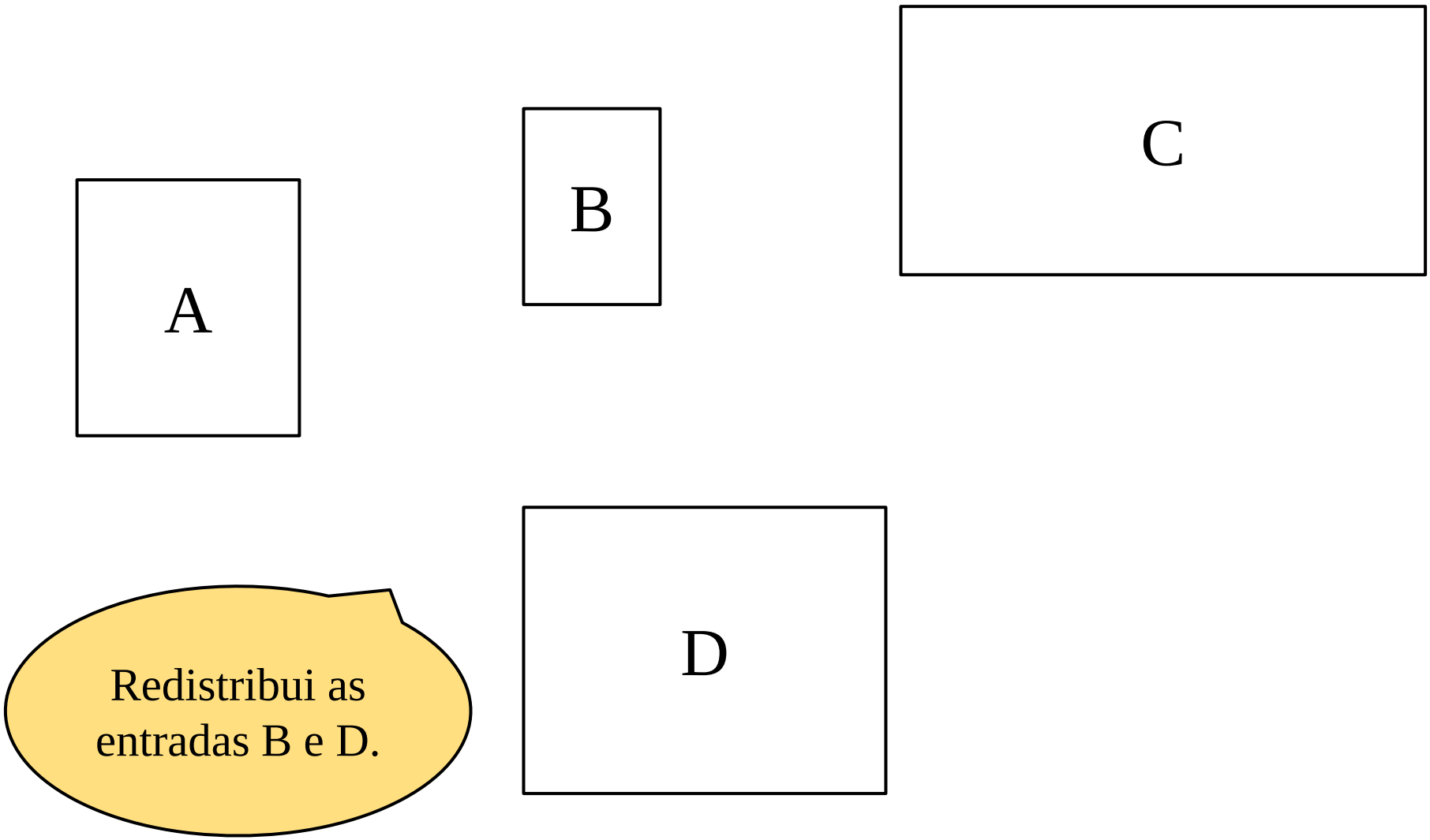
Split: Algoritmo Linear – Exemplo



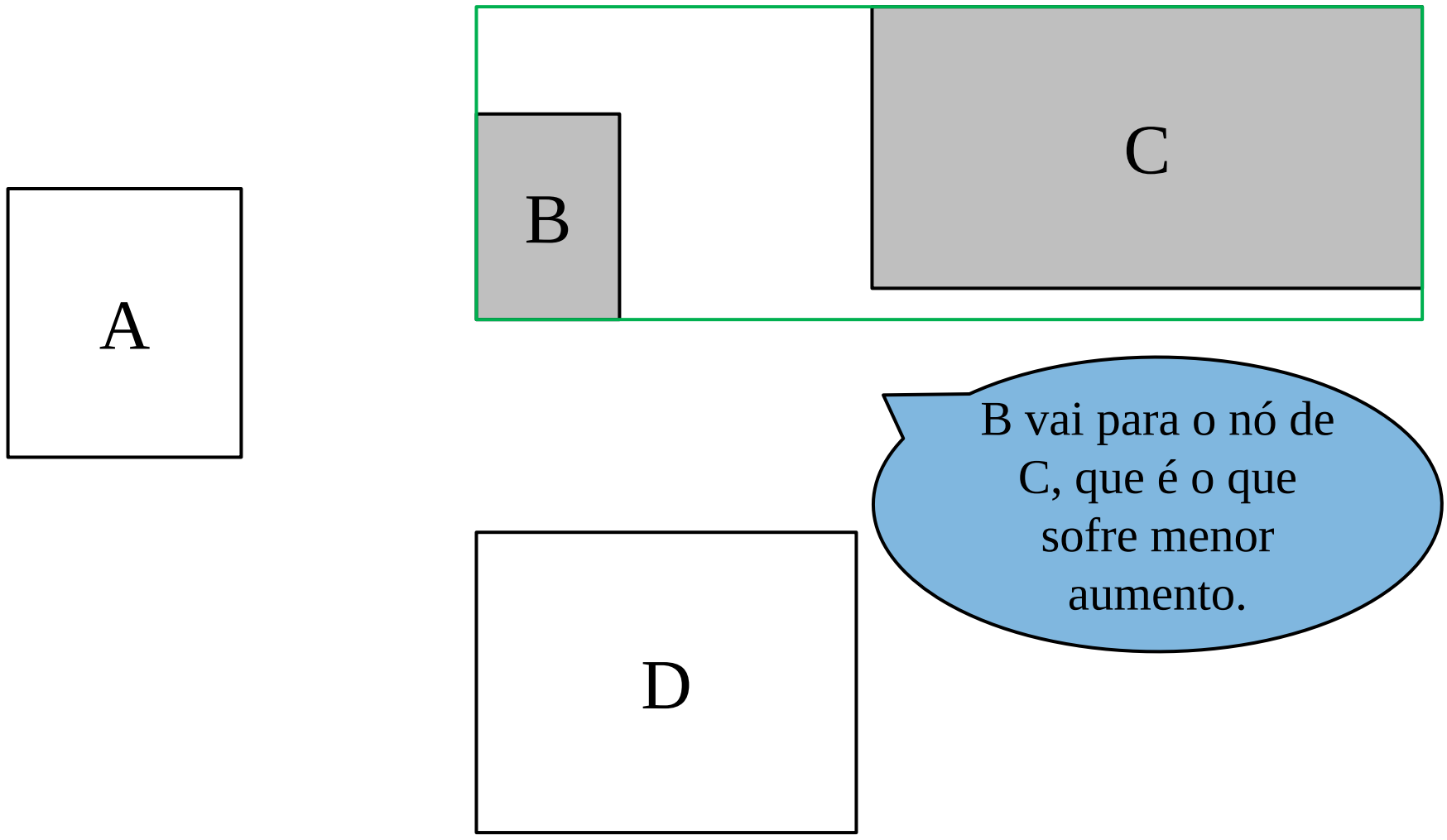
Split: Algoritmo Linear – Exemplo



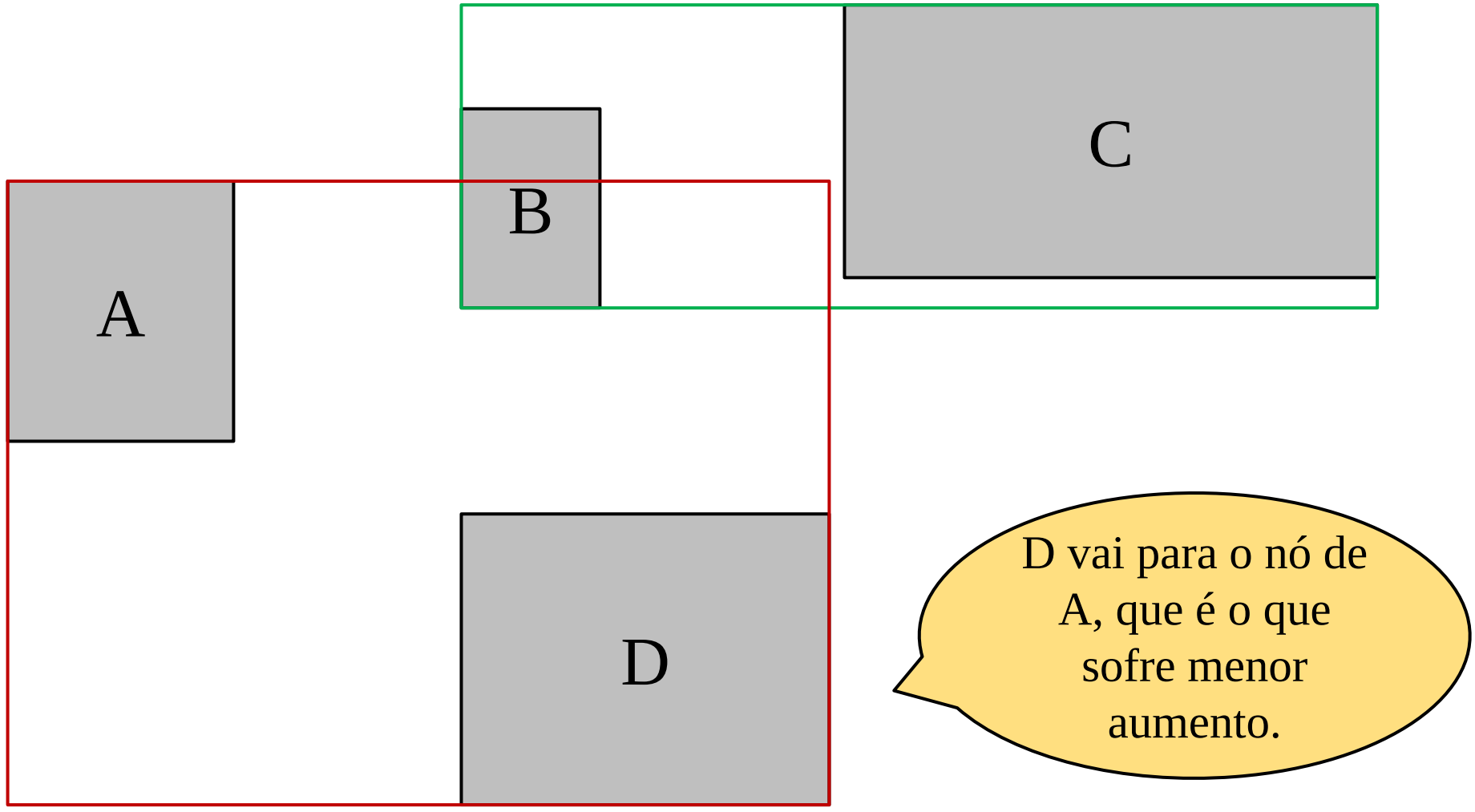
Split: Algoritmo Linear – Exemplo



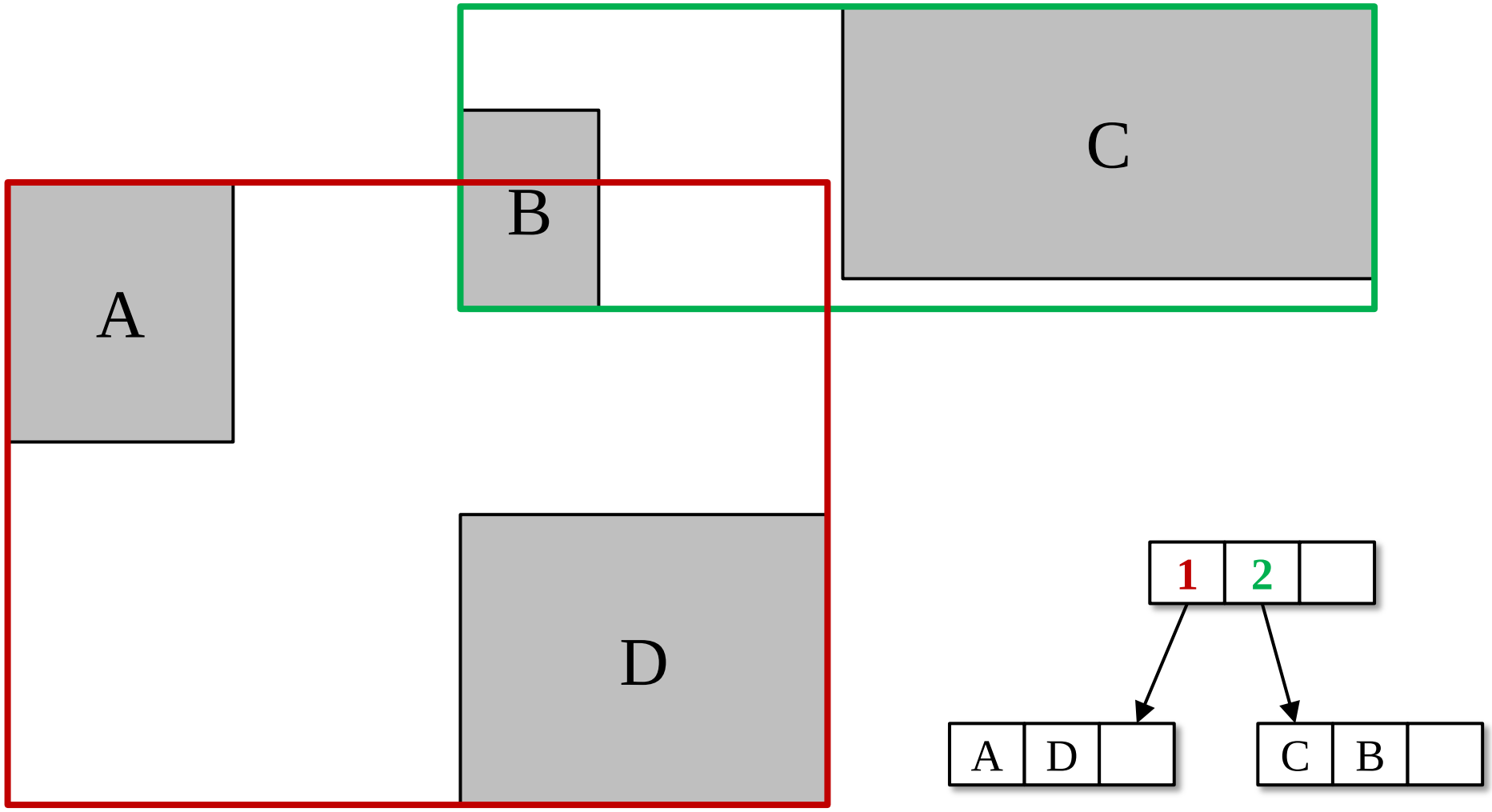
Split: Algoritmo Linear – Exemplo



Split: Algoritmo Linear – Exemplo



Split: Algoritmo Linear – Exemplo



Split: Algoritmo Exaustivo

- Testa todos os agrupamentos possíveis com relação ao menor aumento de MBRs e área de sobreposição.

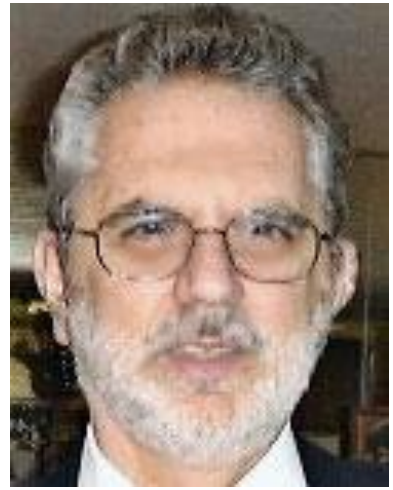
- Complexidade temporal: $O(2^{M-1})$.

- Fins de comparação.

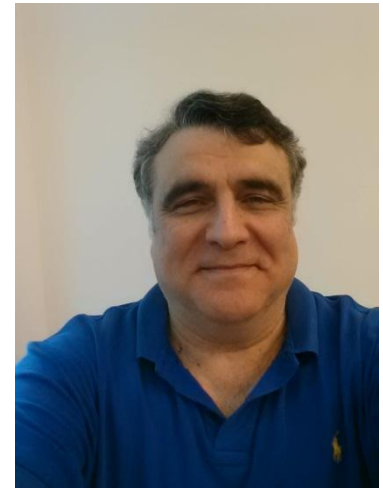
Variações: R^+, R*

Árvore R^+ (1987)

Timos Sellis



Nick Roussopoulos

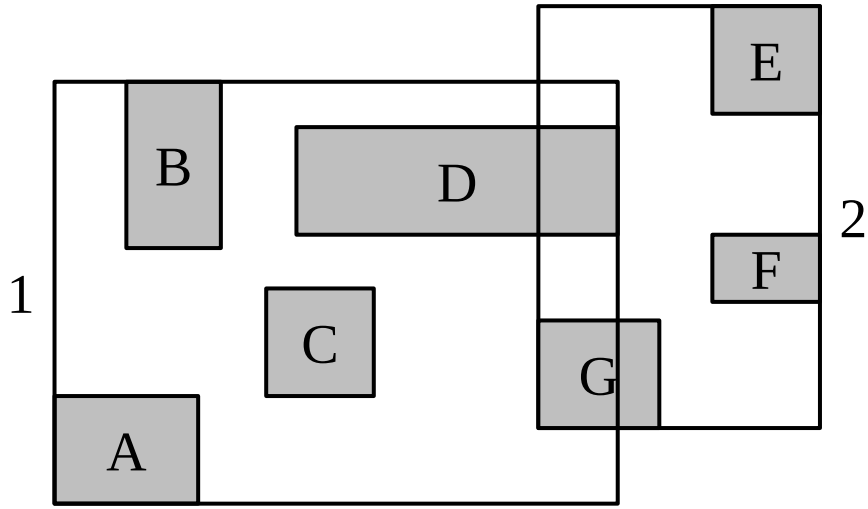


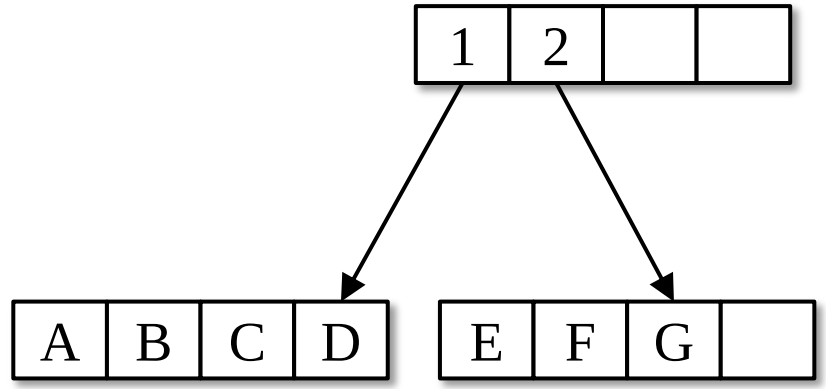
Christos Faloustos

Árvore R^+

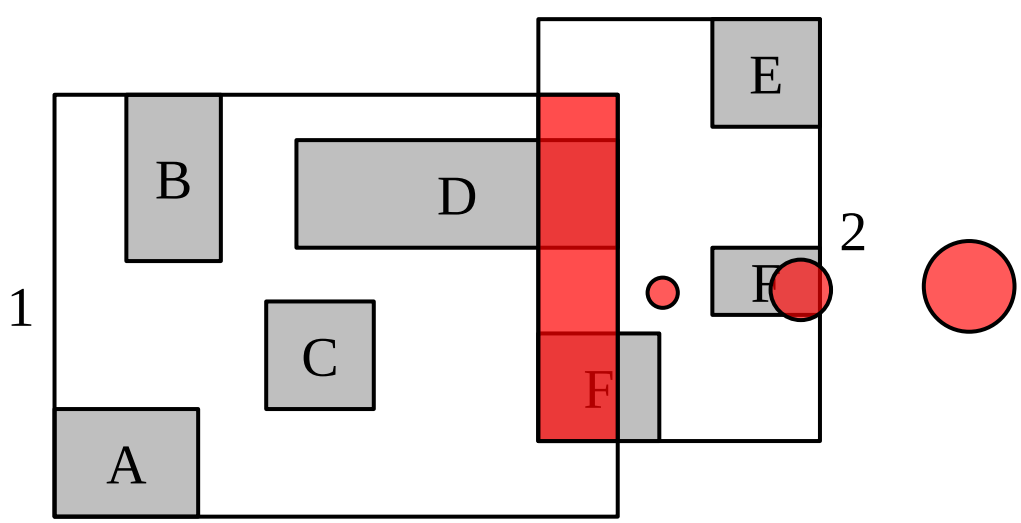
- Motivação:
  - Uma consulta pontual na Árvore R pode percorrer vários caminhos, da raiz até as folhas.
  - Alguns MBRs grandes podem aumentar o grau de sobreposição significativamente, devido ao *dead space*.
- Estas características causam uma degradação de desempenho das consultas, especialmente quando a sobreposição dos MBRs é significativa.
- Solução: a Árvore R^+ não permite sobreposição de MBRs no mesmo nível: técnica de *clipping*.

Árvore R+

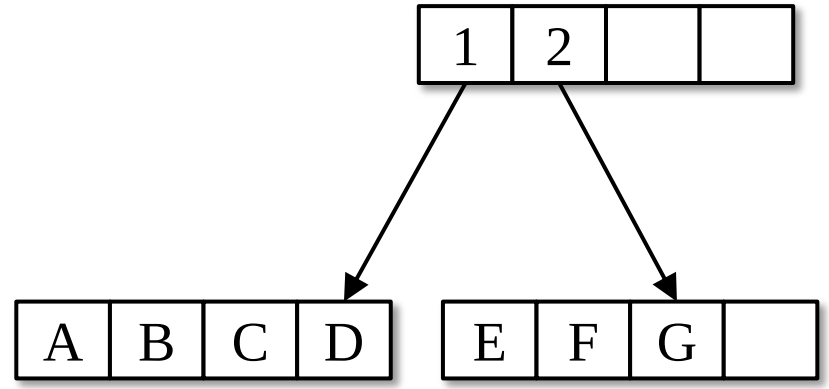




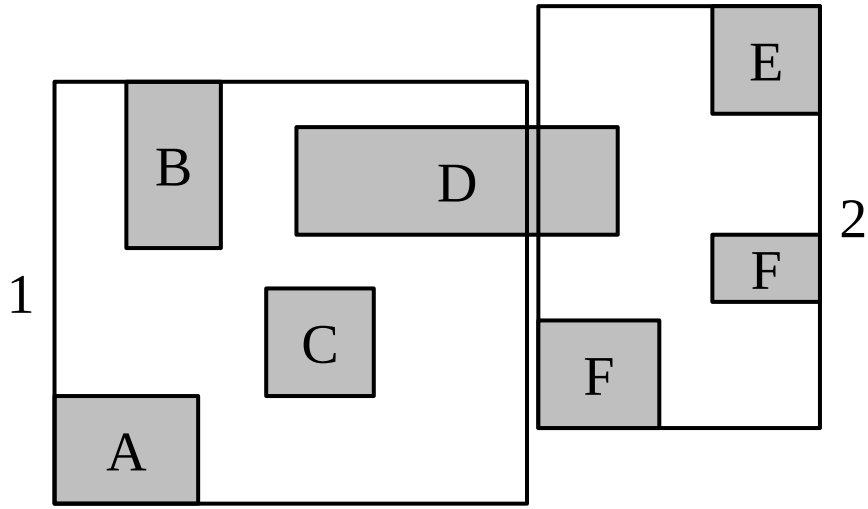
Árvore R+



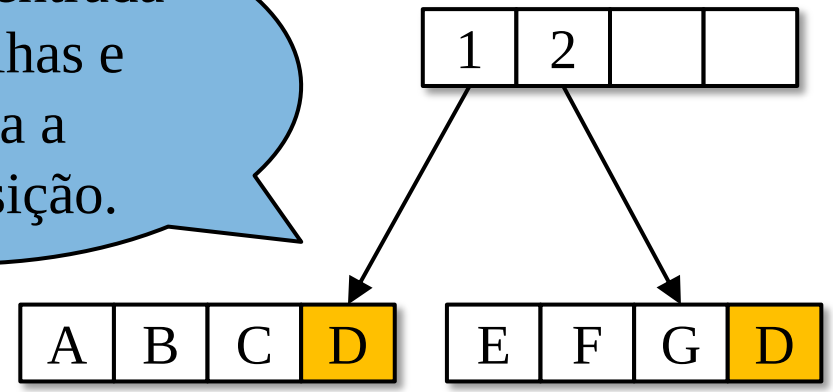
Região de sobreposição no mesmo nível.



Árvore R+



Duplica a entrada D nas folhas e elimina a sobreposição.



Árvore R^+

- Considerações:
- Busca: deve eliminar as duplicatas.
- Inserção: maior complexidade em relação à árvore R.
- Remoção: deve eliminar o objeto de todas as folhas em que aparece.
- Consumo de espaço físico: é problema?

Árvore R* (1990)

Norbert Beckmann

Hans-Peter Kriegel

Ralf Schneider

Bernhard Seeger

Árvore R*

- A árvore R é baseada na minimização de área de seus MBRs.
- A árvore R* considera minimizar:
  - área dos MBRs.
  - área de sobreposição entre MBRs.
  - margens dos MBRs: deixar os MBRs mais "quadrados" ajusta melhor o espaço nos níveis superiores.
- Principal alteração em relação à árvore R original:
  - inserção.
  - reinserção.
  - parâmetro: cerca de 30% da capacidade do nó.

Referências

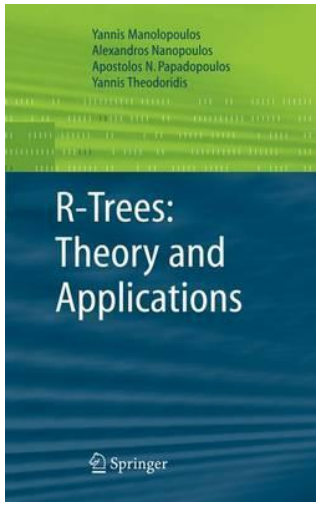
- GUTTMAN, A. R-trees: A dynamic index structure for spatial searching. *ACM SIGMOD Record*, ACM, New York, NY, USA, v. 14, n. 2, p. 47-57, jun. 1984.

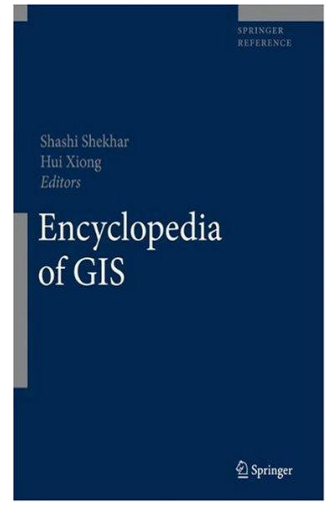
- SELLIS, T. K.; ROUSSOPOULOS, N.; FALOUTSOS, C. The R^+-tree: A dynamic index for multi-dimensional objects. In: *Proceedings of 13th International Conference on Very Large Data Bases*. Brighton, England: Morgan Kaufmann Publishers Inc., 1987. (VLDB '87), p. 507-518.

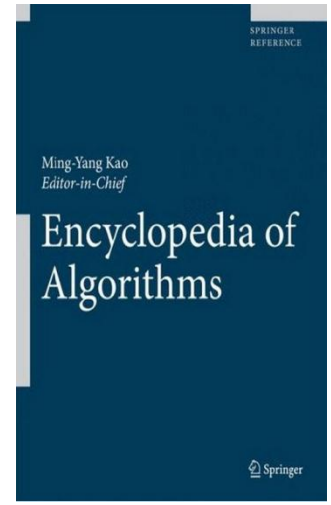
- BECKMANN, N.; KRIEGEL, H.-P.; SCHNEIDER, R.; SEEGER, B. The R*-tree: An efficient and robust access method for points and rectangles. *ACM SIGMOD Record*, ACM, New York, NY, USA, v. 19, n. 2, p. 322-331, maio 1990.

Referências

- MANOLOPOULOS, Y.; NANOPOULOS, A.; PAPADOPOULOS, A. N.; THEODORIDIS, Y. *R-Trees: Theory and Applications*. Springer, 2006.
- LIU, L; ÖZSU, M. T. *Encyclopedia of Database Systems*. Springer, 2009.
- SHEKHAR, S.; XIONG, H. *Encyclopedia of GIS*, Springer, 2008.
- KAO, M.-Y. *Encyclopedia of Algorithms*. Springer, 2008.







Demonstração

- http://gis.umb.no/gis/applets/rtree2/jdk1.1/