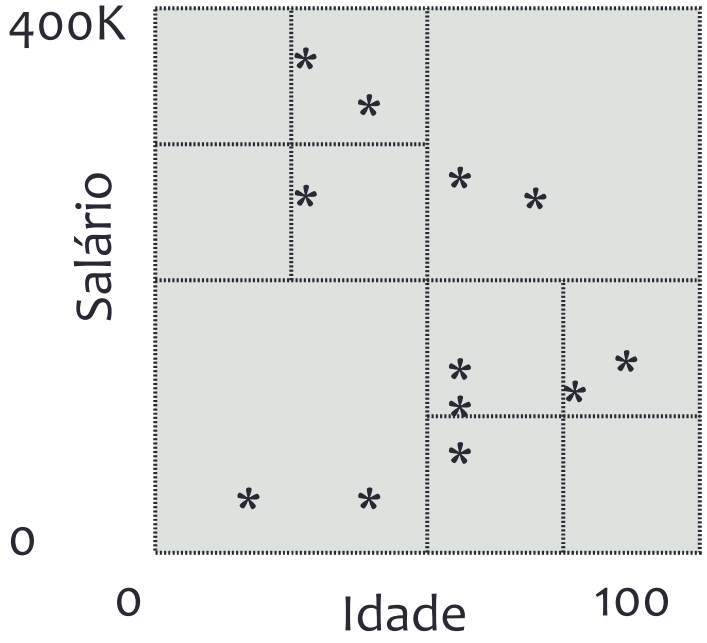
ÁRVORES QUADRANTES (QUAD TREES)
ÁRVORES R (R-TREES)

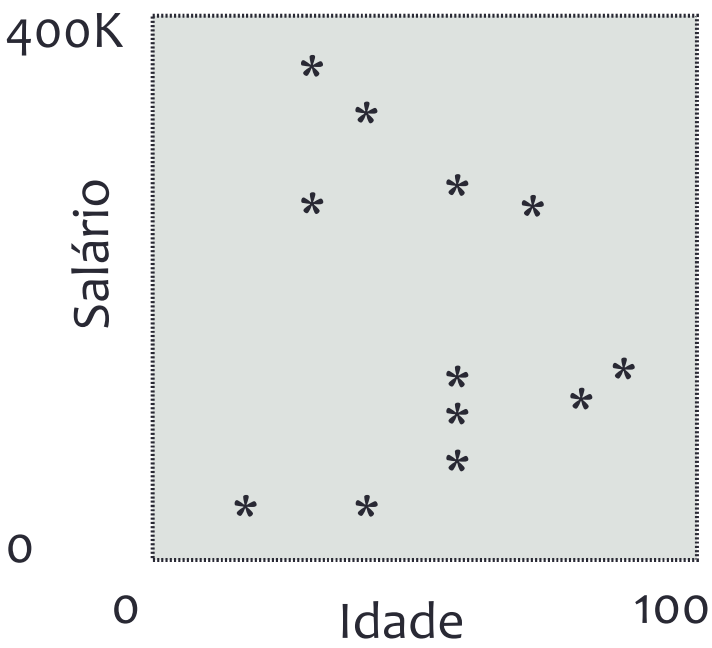
Sérgio Mergen

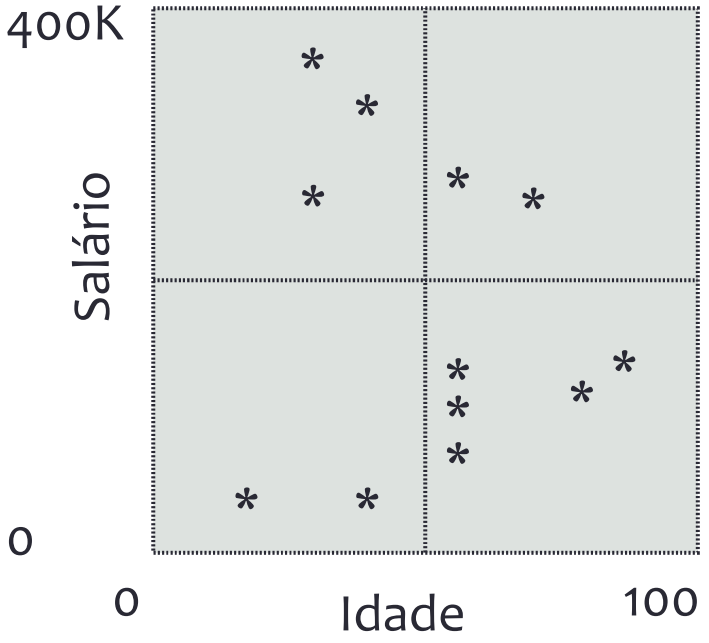
Árvores quadrante

- Cada nó interior corresponde a:
  - uma região quadrada ou
  - um cubo *k*-dimensional em *k*-dimensões
- Para fins didáticos, analisaremos o caso de regiões quadradas
- Se os pontos no quadrado couberem em um bloco
  - O bloco é um nó folha
- Se não couberem
  - Dividir em mais quatro quadrantes

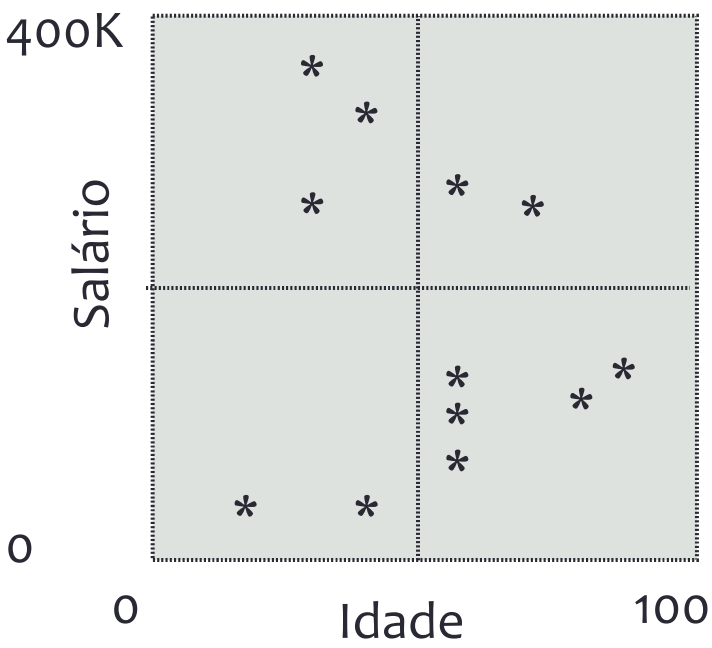


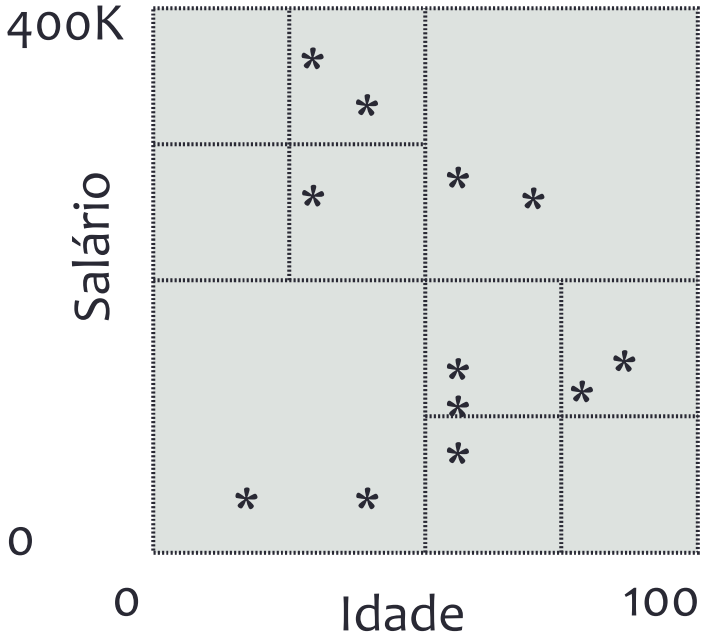
Árvores quadrante



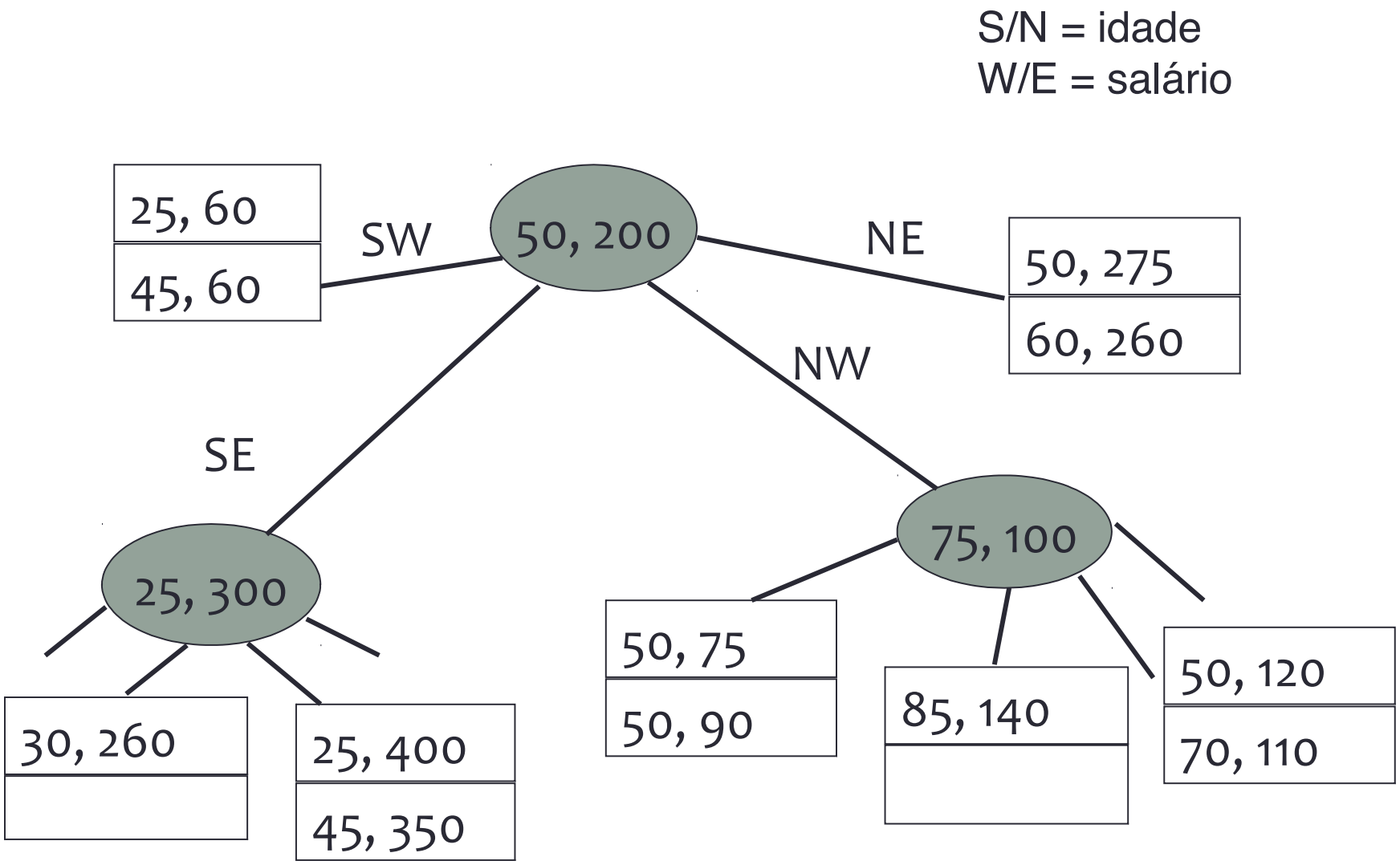


Árvores quadrante





Árvores de quadrante



Melhor ocupação de um nó

- Em uma árvore com k dimensões
  - Cada nó interno possui filhos
  - Ex. Se couberem 128 ponteiros para filhos em um bloco
    - Pode-se trabalhar com uma árvore de 7 dimensões

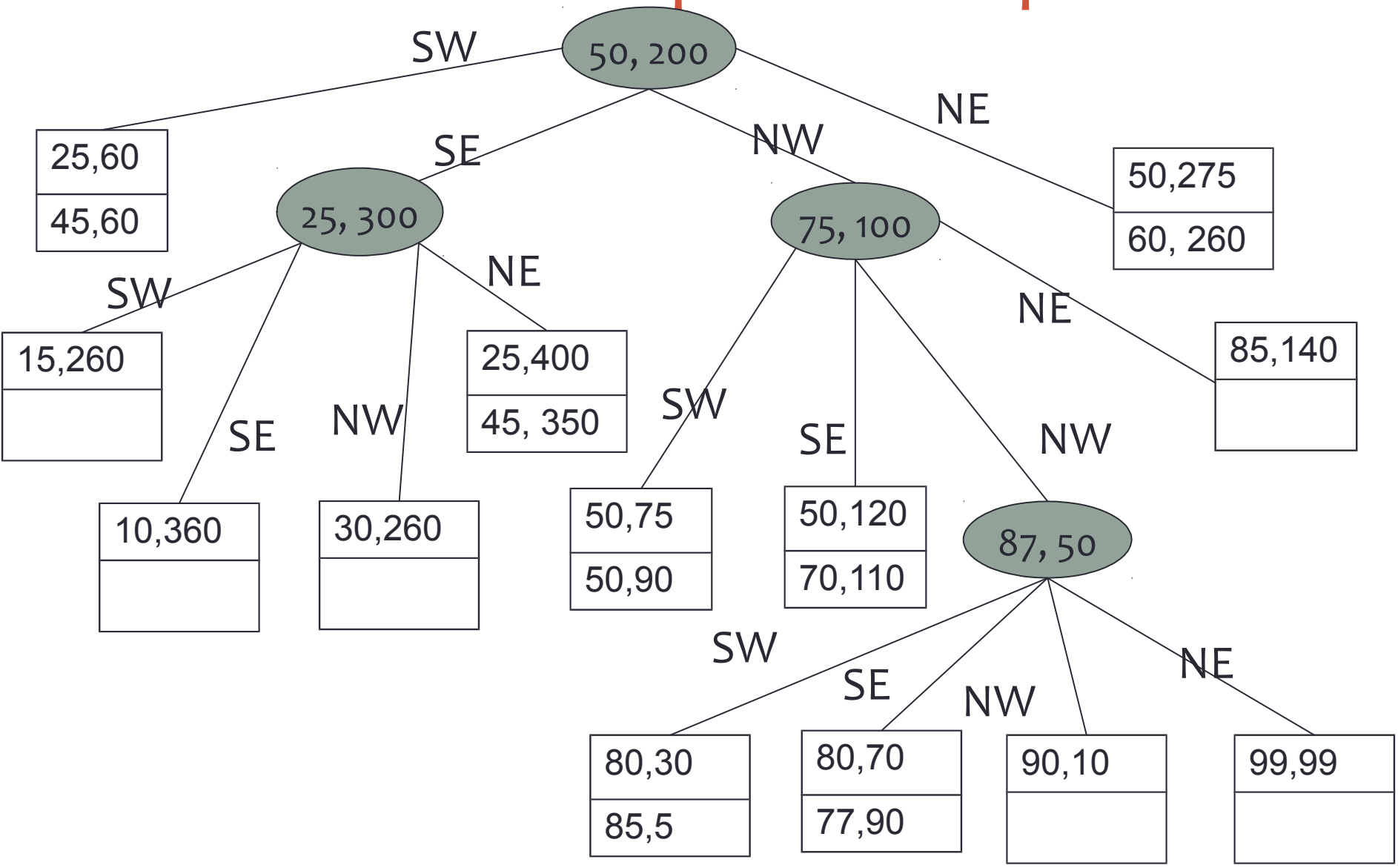
Operações sobre árvores quadrantes

- Pesquisa
  - como na árvore binária, até atingir uma única folha
- Inserção
  - como na pesquisa, procura-se até atingir uma única folha.
  - Se houver espaço
    - adiciona-se o registro.
  - Caso contrário
    - o registro existente terá que ser dividido em quatro quadrantes.

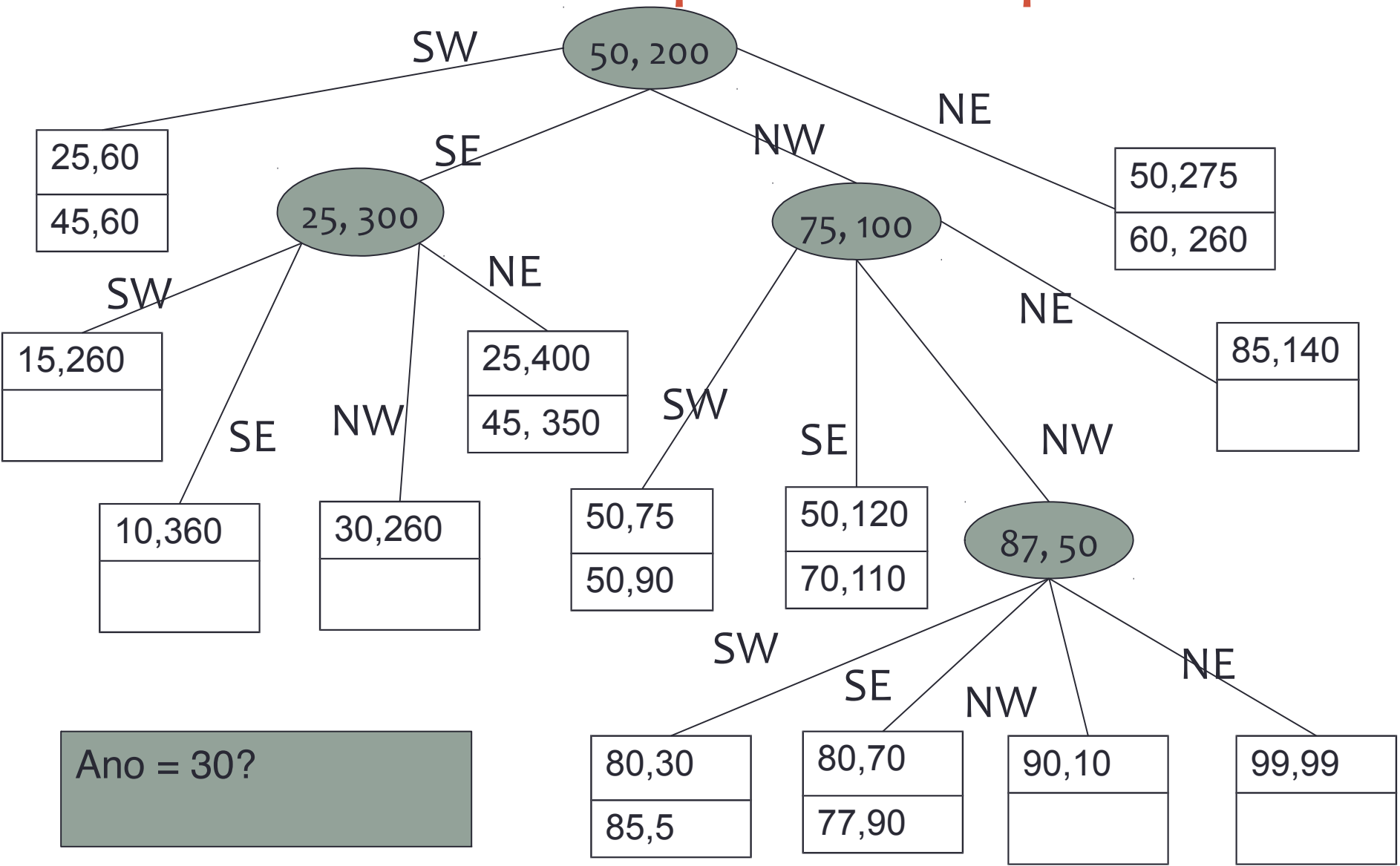
Operações sobre árvores quadrantes

- Consultas:
  - De correspondência parcial:
    - Com n dimensões, pode precisar seguir subárvores de um nó
  - De intervalo:
    - A divisão pode não satisfazer inteiramente o intervalo da consulta
  - De vizinho mais próximo:
    - Usar uma consulta de intervalo com um parâmetro d

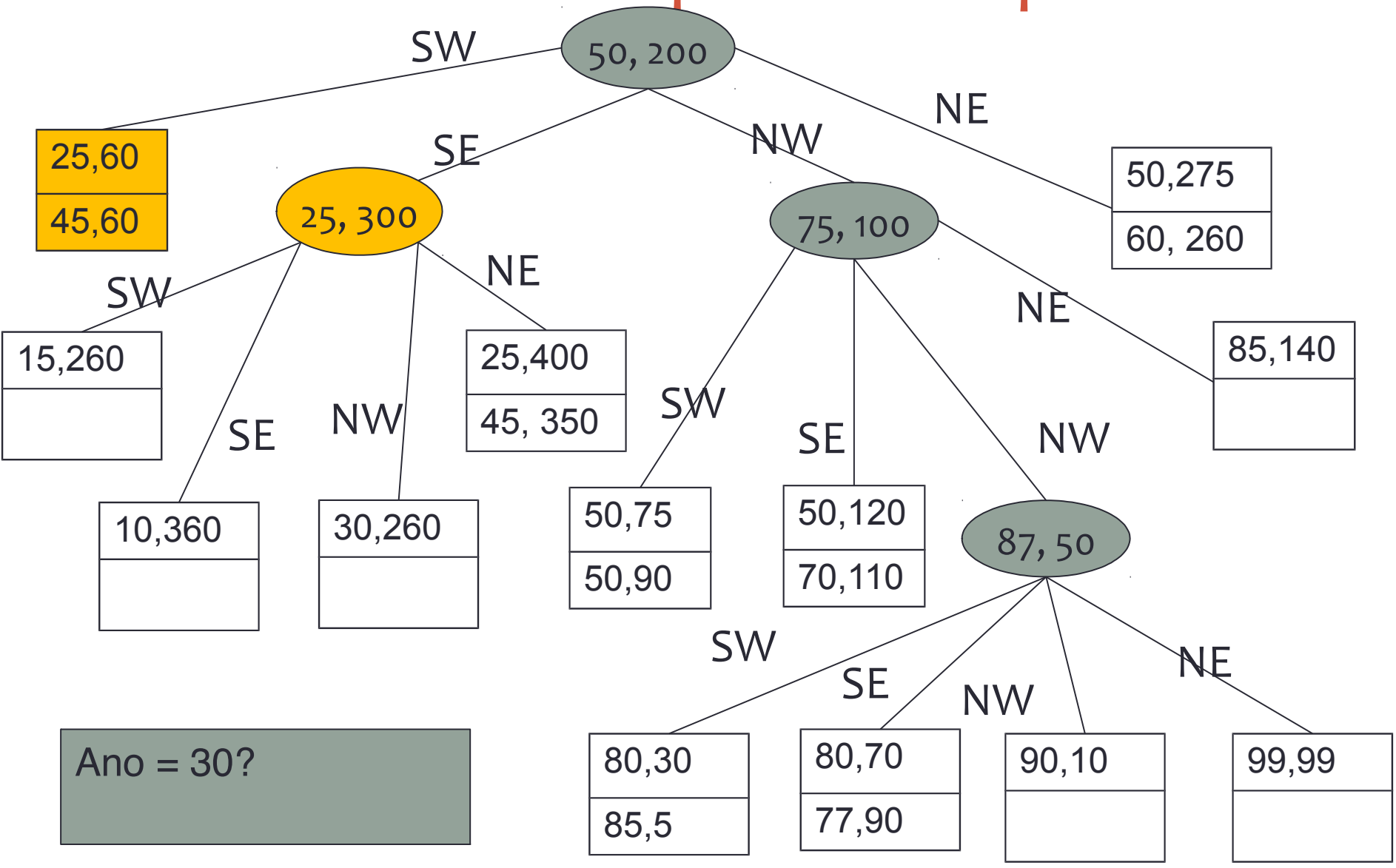
Consulta de correspondência parcial



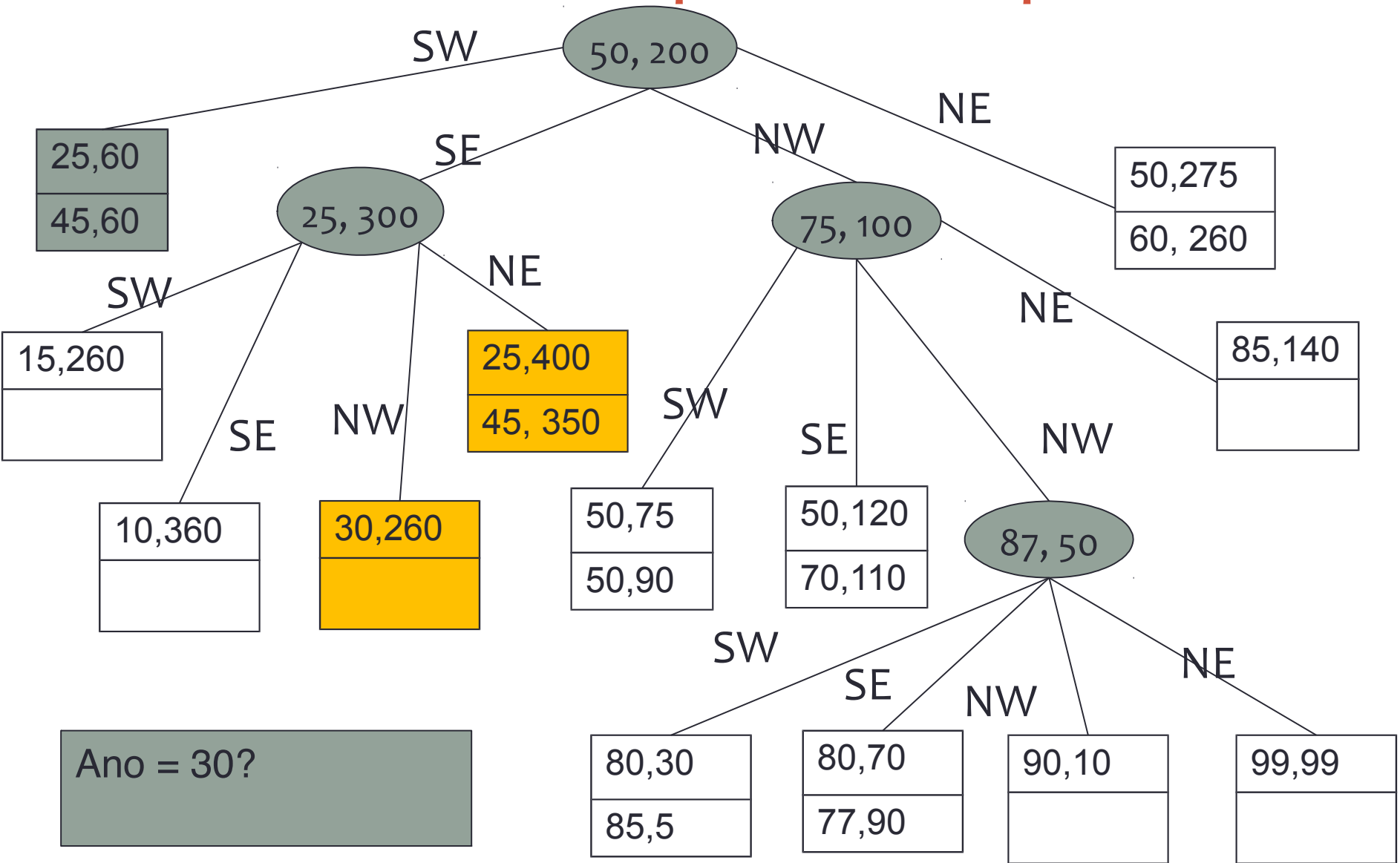
Consulta de correspondência parcial



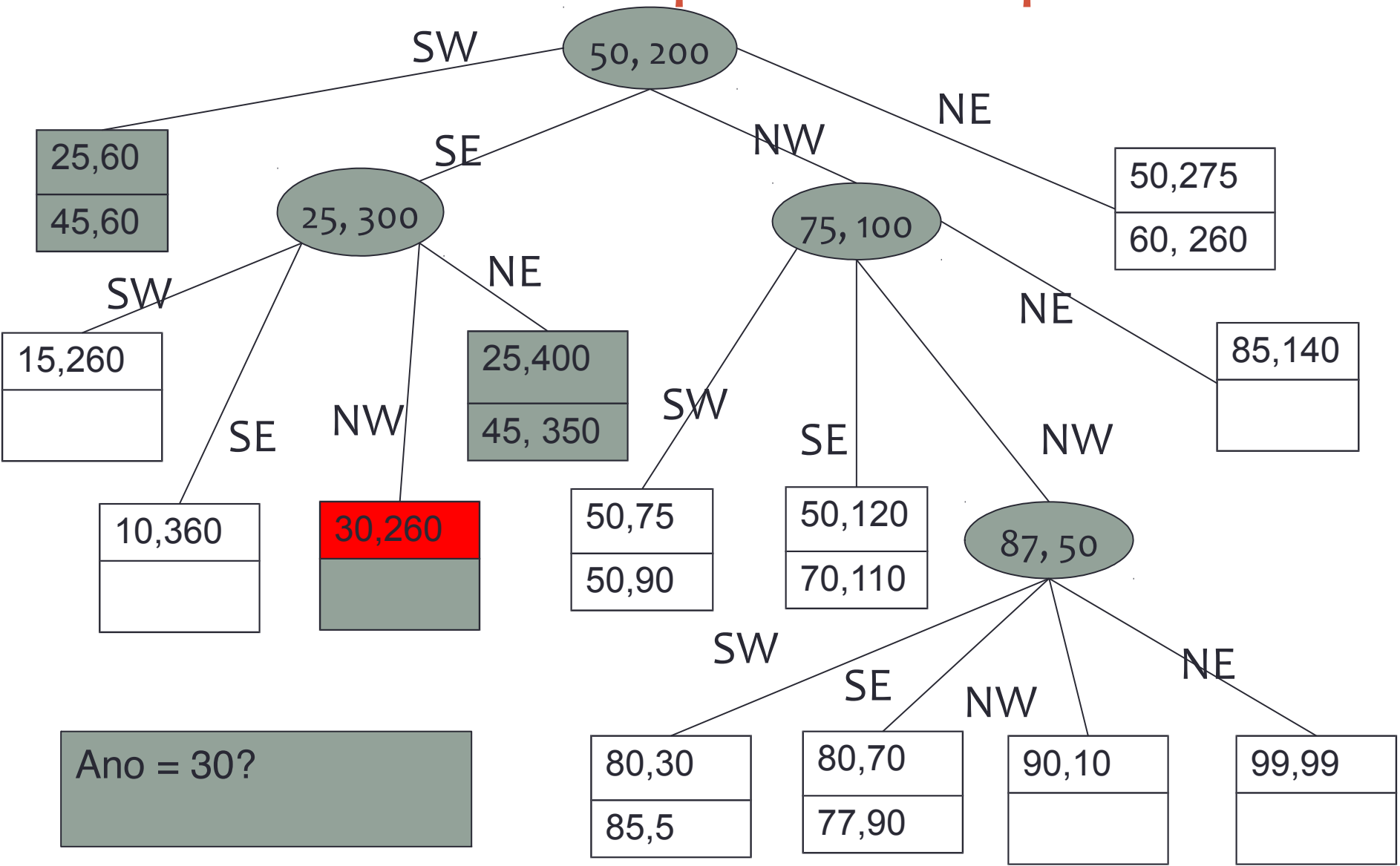
Consulta de correspondência parcial



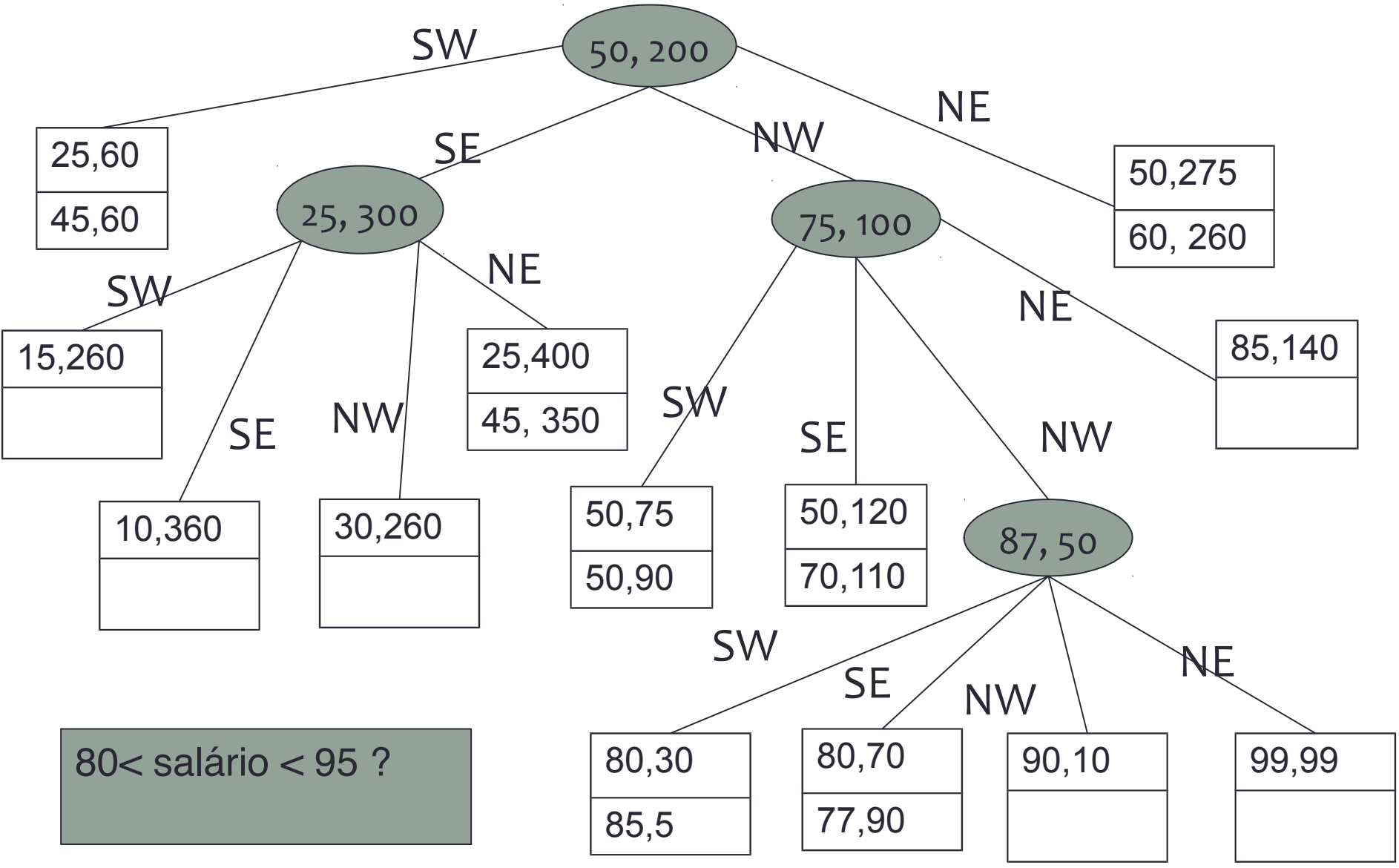
Consulta de correspondência parcial



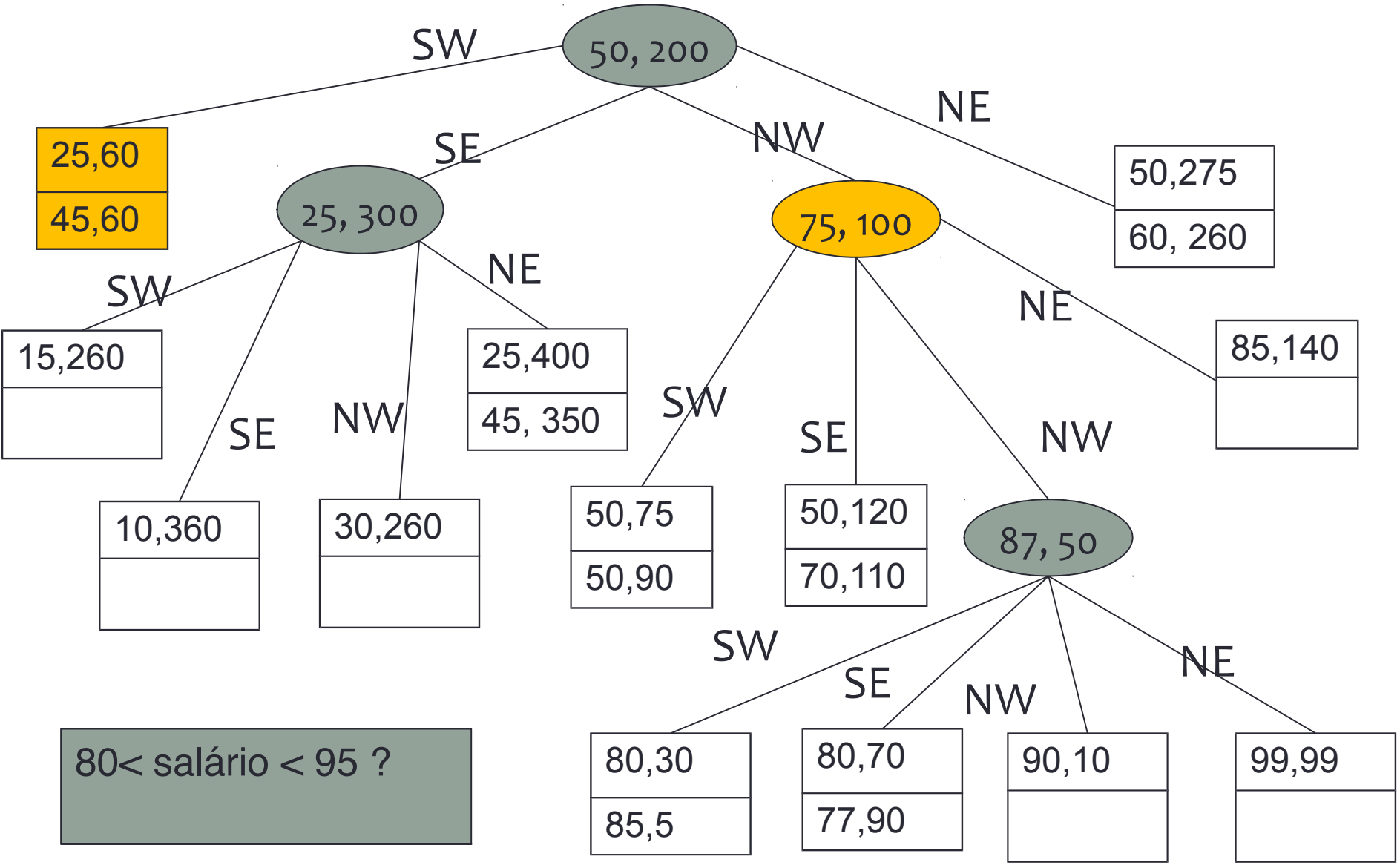
Consulta de correspondência parcial



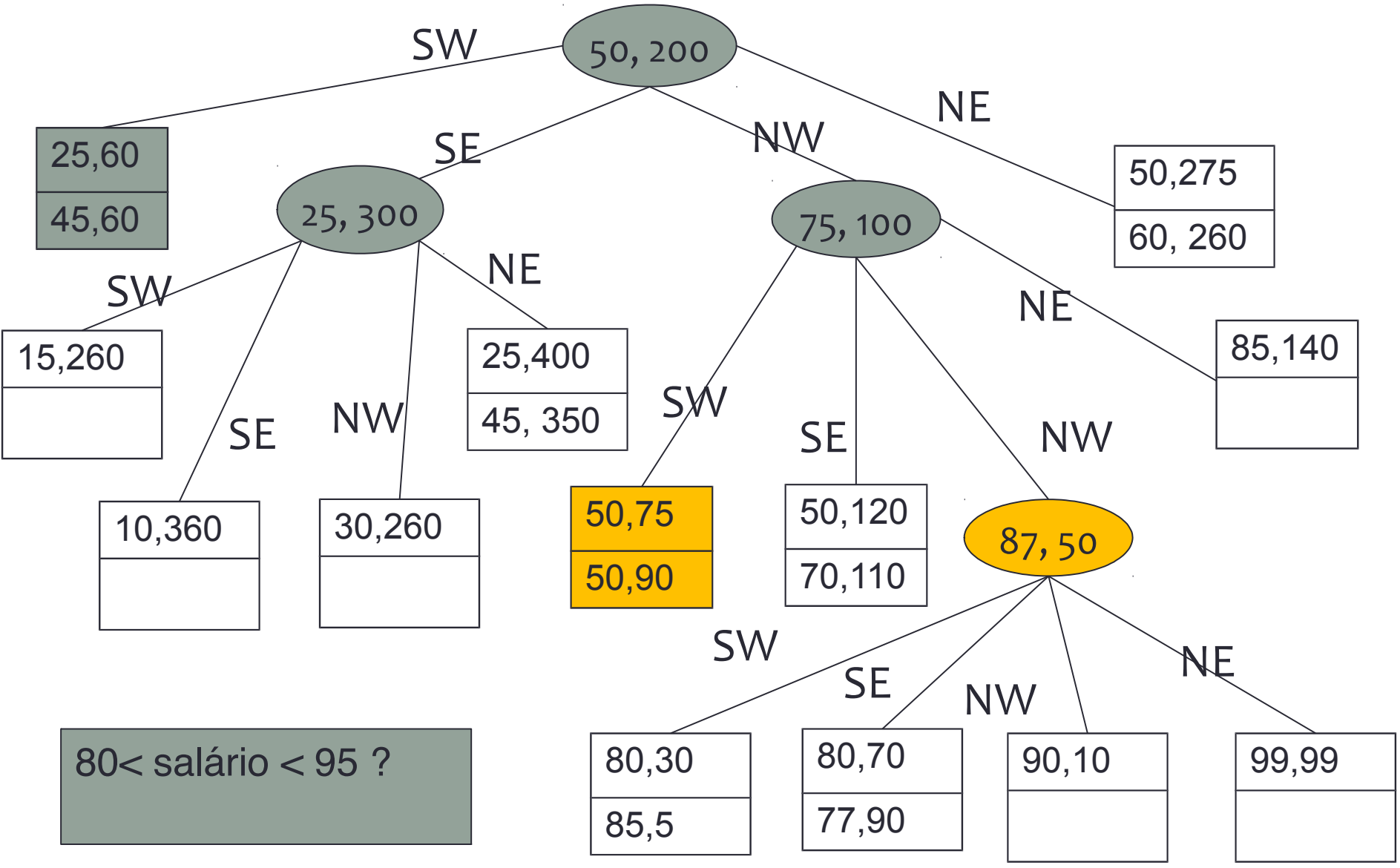
Consulta de intervalo



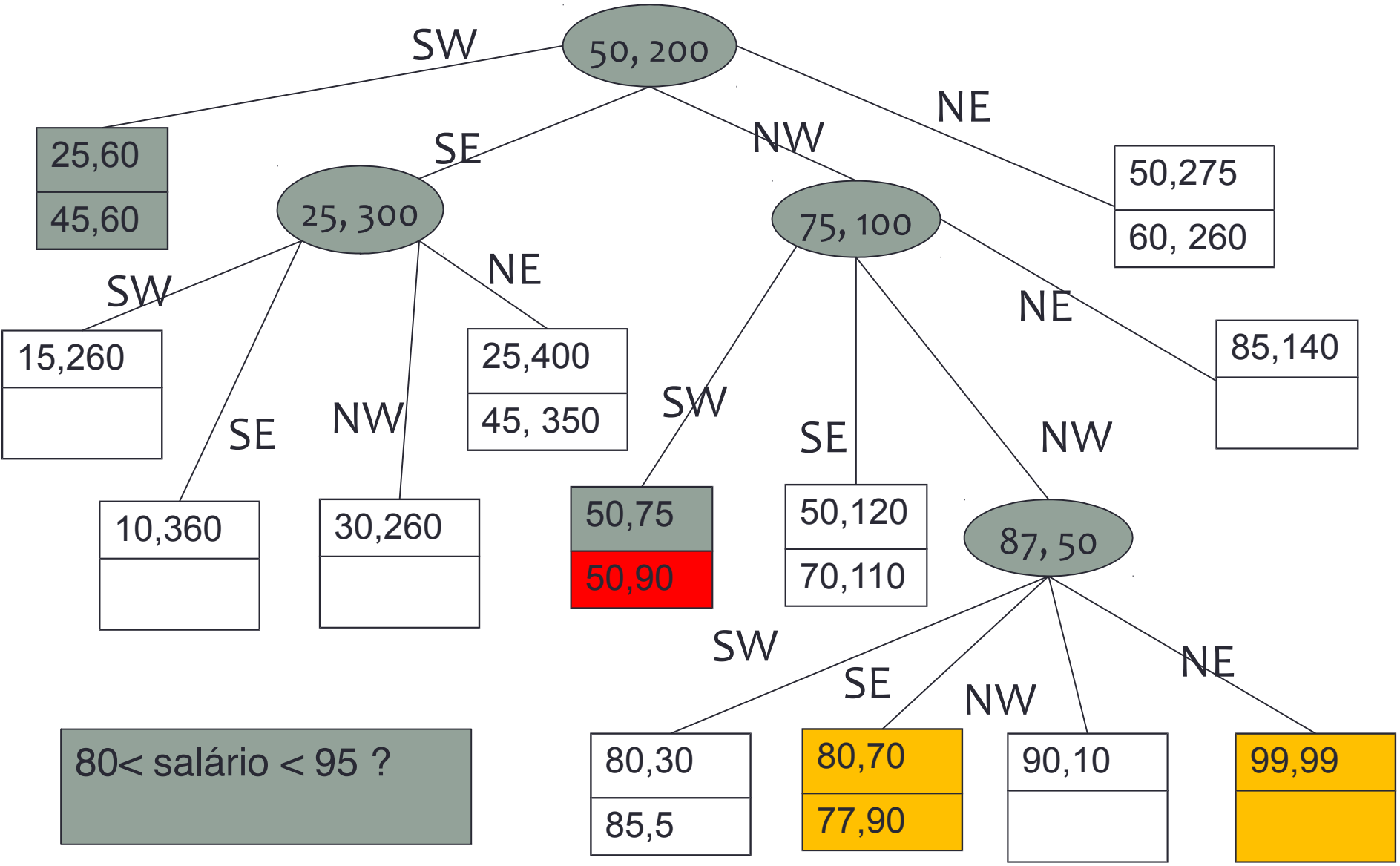
Consulta de intervalo



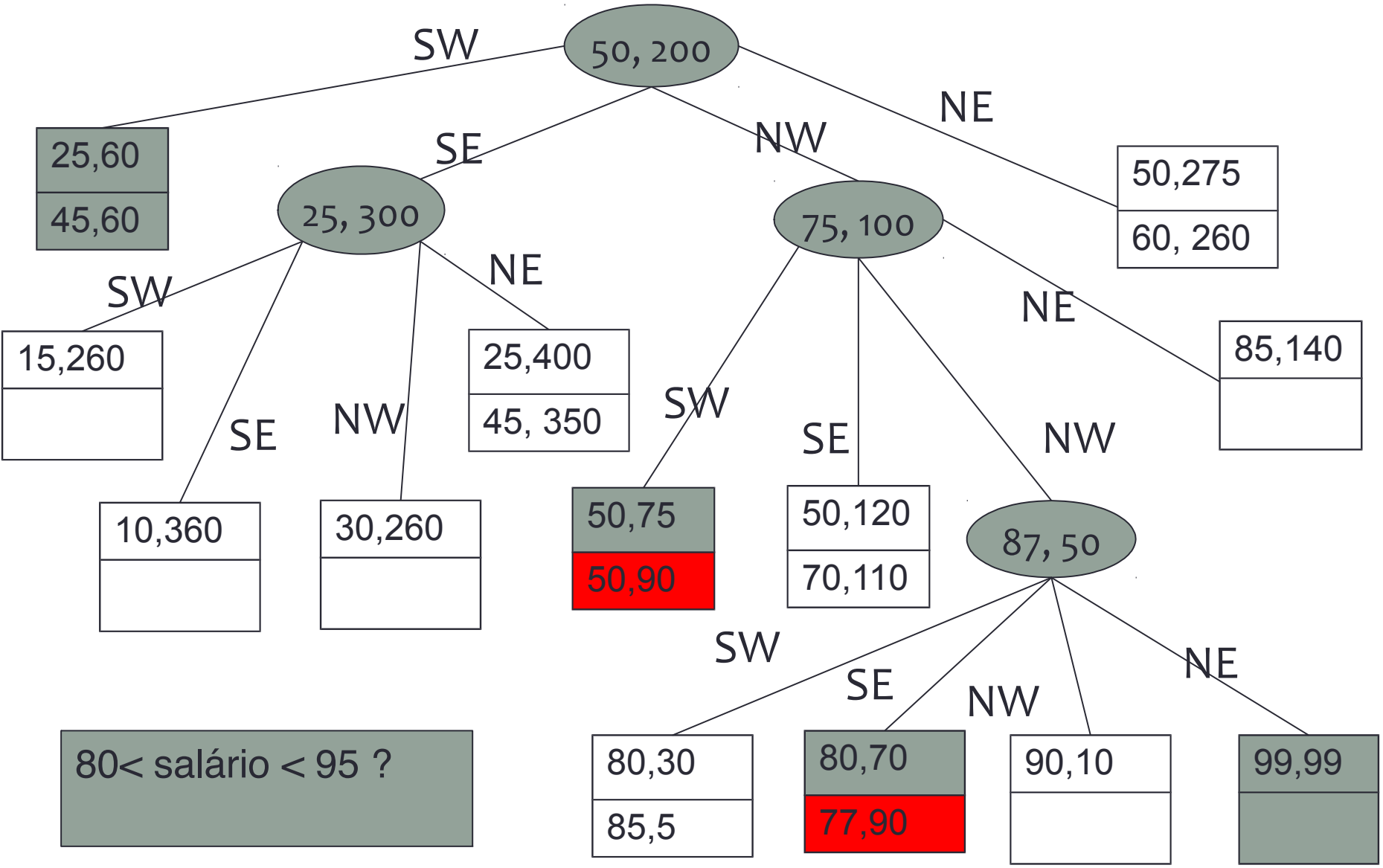
Consulta de intervalo



Consulta de intervalo



Consulta de intervalo

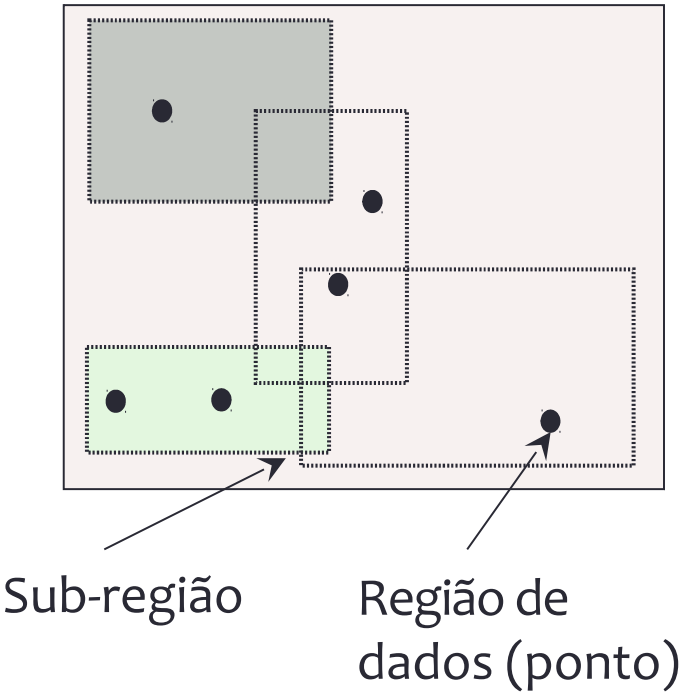


Análise

- Comparação com árvores kd
  - Semelhança
    - Pensados para uso em memória
    - Responde o mesmo tipo de consulta
  - Diferença
    - Árvore mais larga e menos profunda
    - Forma de divisão de um nó é pior
      - pode levar a muitos blocos vazios
      - O ponto de divisão pode não distribuir os elementos de forma regular.

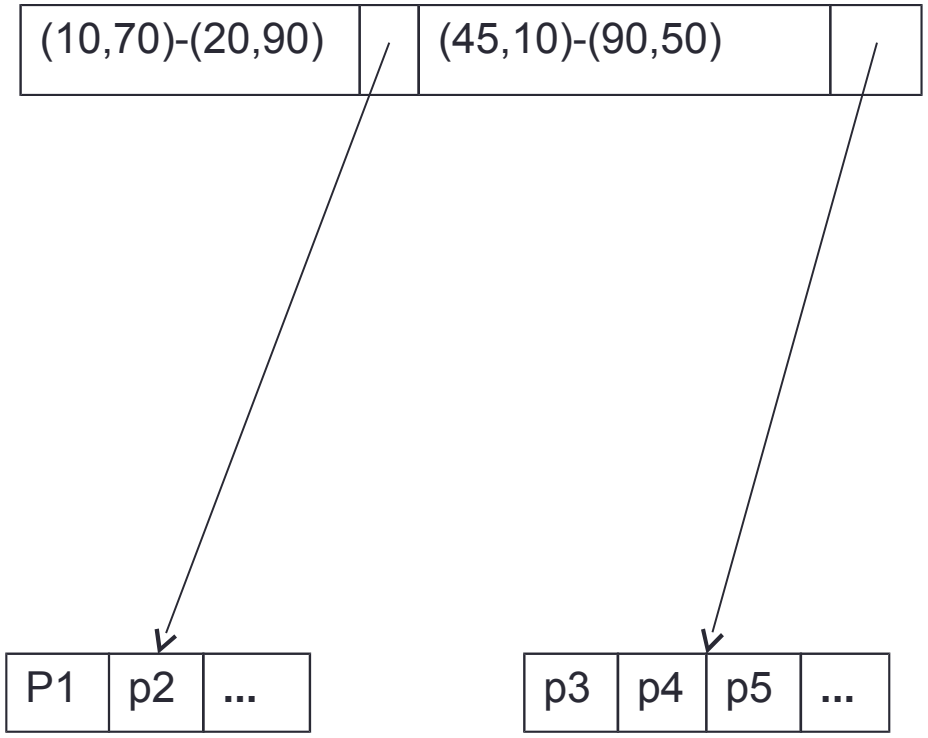
Árvores R

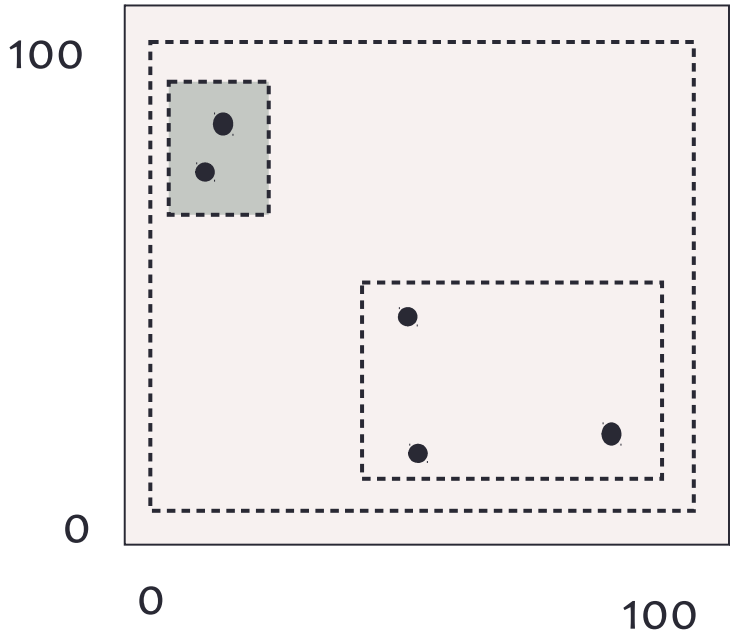
- Representa dados em regiões n-dimensionais
- Uma região pode ser composta por
  - Um conjunto de sub-regiões
  - Um conjunto de regiões de dados (pontos)
- Sub-regiões de uma região
  - podem se sobrepor
    - Porém, deve-se evitar (ou limitar) isso
  - Devem conter todos pontos da região pai
- Um ponto pode pertencer a mais de uma sub-região



Árvores R

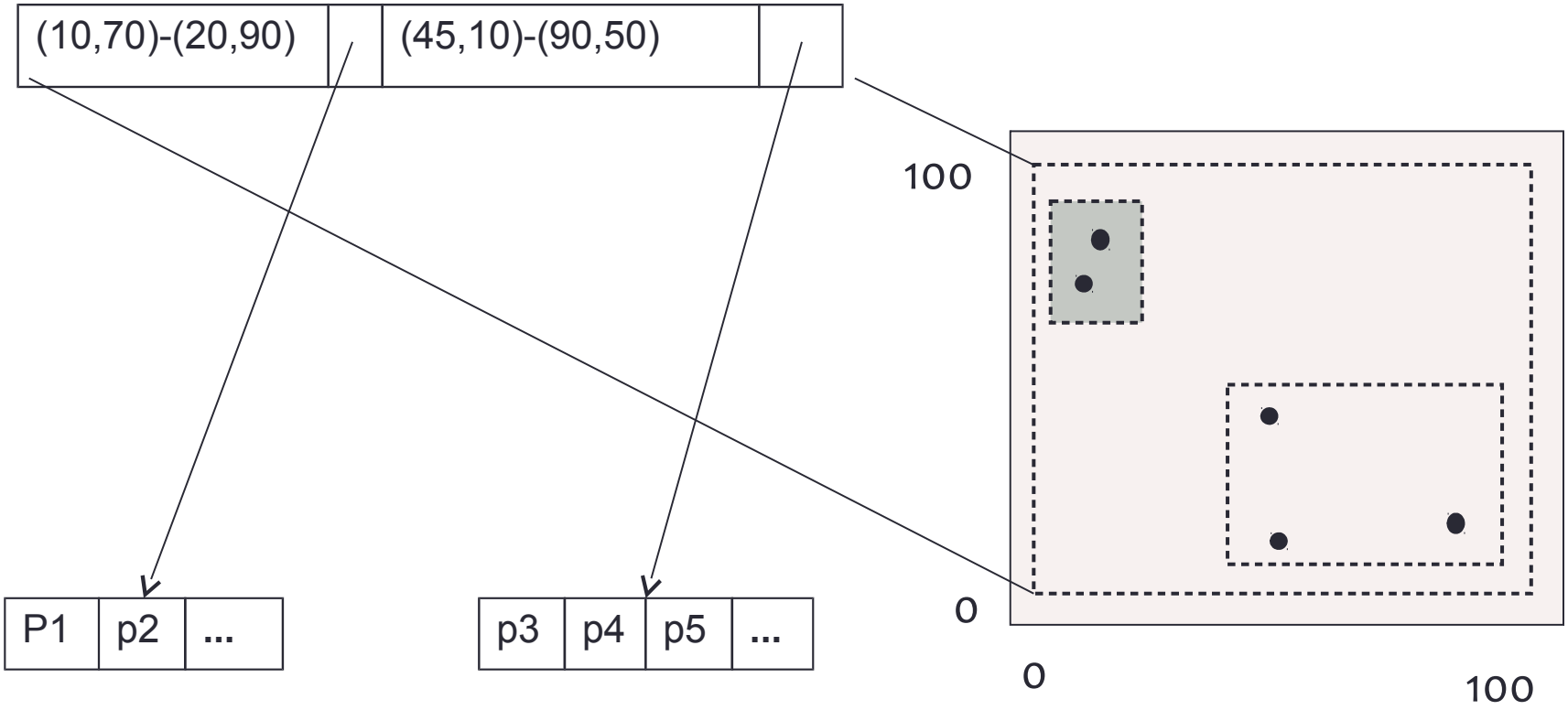
- Regiões são mapeadas como nós da árvore R
- A região mais externa representa o nó raiz
- Operações são semelhantes às operações em árvores B+





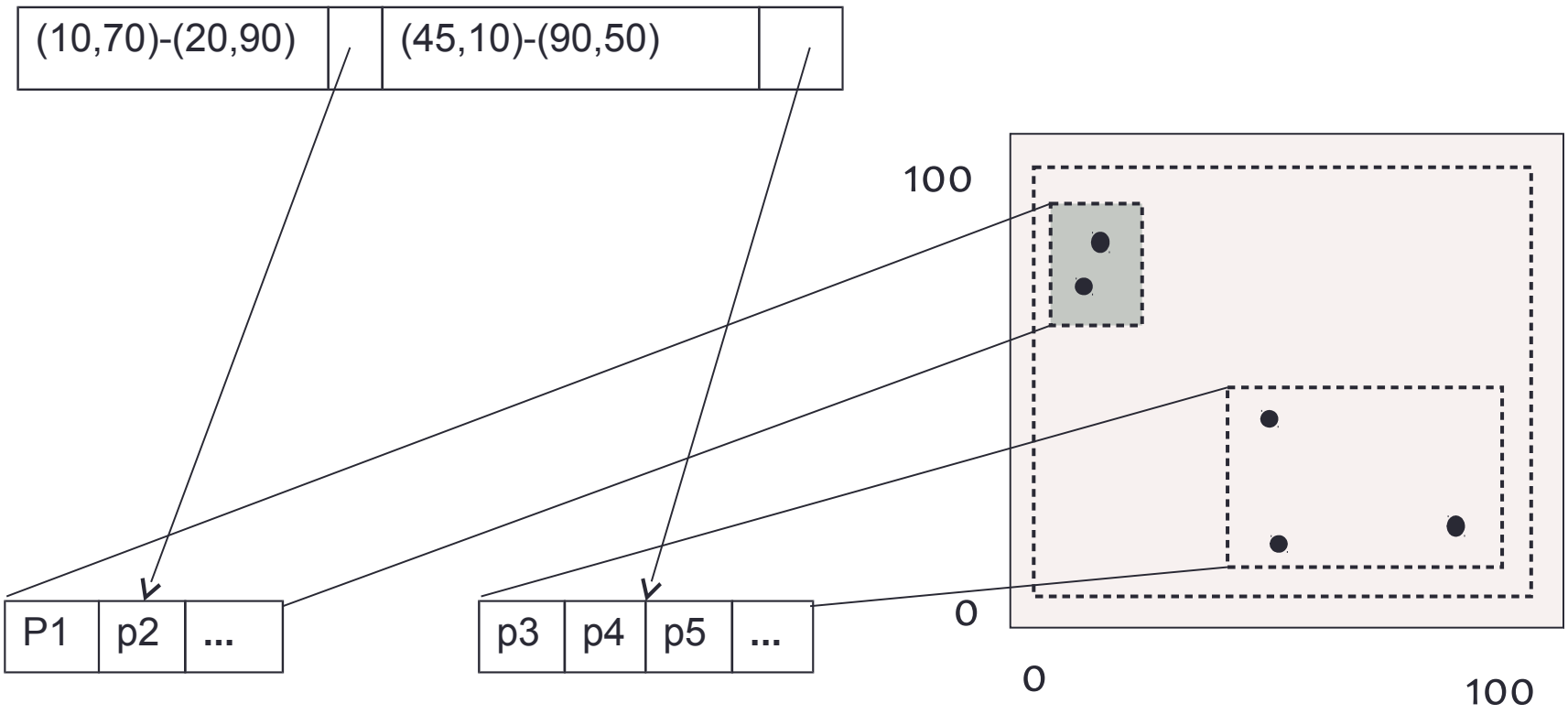
Árvores R

- Regiões são mapeadas como nós da árvore R
- A região mais externa representa o nó raiz
- Operações são semelhantes às operações em árvores B+



Árvores R

- Regiões são mapeadas como nós da árvore R
- A região mais externa representa o nó raiz
- Operações são semelhantes às operações em árvores B+



Inserção em árvores R

- O processamento começa pelo nó raiz
- Caminhamento
  - Procurar pela sub-região (sub-regiões) onde o ponto se encaixe
  - Escolher uma dessas sub-regiões
  - Visitar o nó filho que corresponda a essa sub-região
  - Repetir do início, até encontrar um nó folha

Inserção em árvores R

- Nó não folha
  - Se o ponto não encaixar em nenhuma das sub-regiões
    - Deve-se aumentar uma das sub-regiões

- Escolher uma política para decidir sobre qual sub-região aumentar
  - Ex. A sub-região que representar o menor aumento na área

Inserção em árvores R

- Nó folha
  - Se existe espaço no nó folha
    - Adicionar o ponto
  - Caso contrário
    - Dividir a folha em duas sub-regiões
  - Escolher uma política para a divisão das sub-regiões
  - Ex.
    - Procurar evitar sobreposições
    - Procurar balancear os pontos

Inserção em árvores R

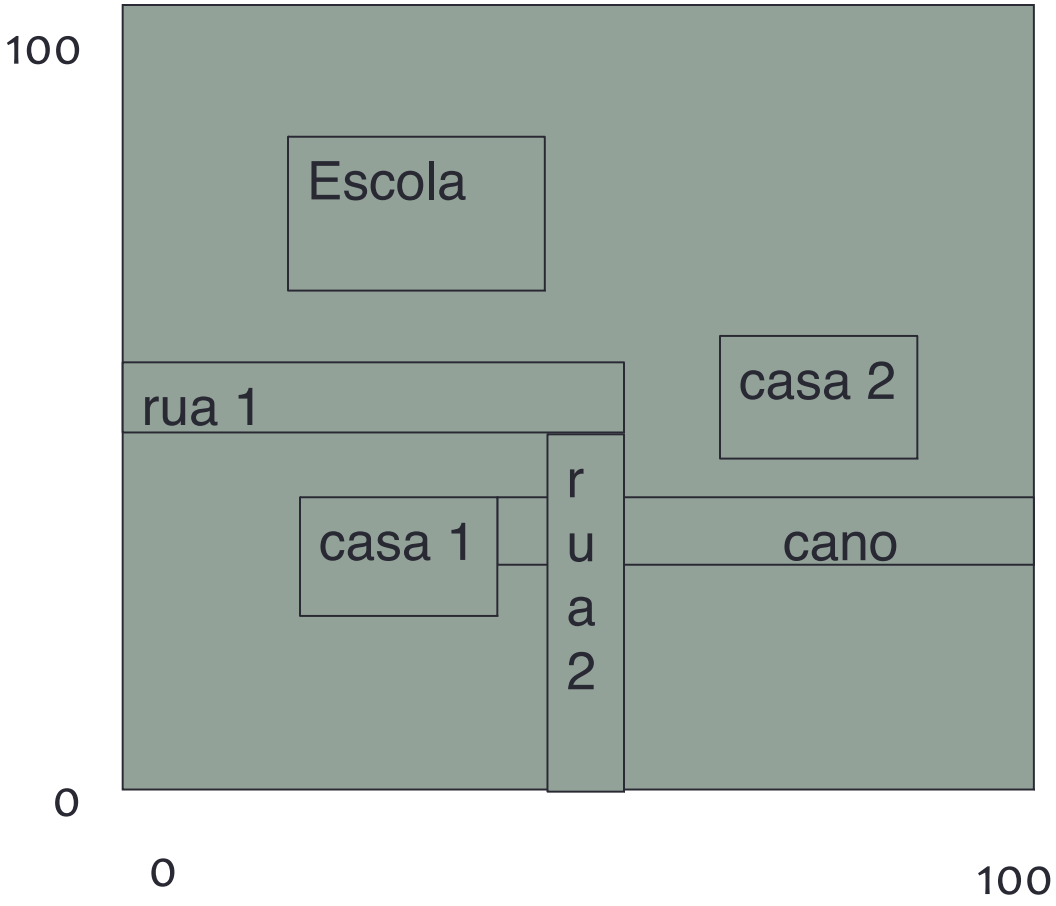
- Com a divisão, a árvore precisa ser atualizada (como nas árvores B+):

- Atualização do nó pai
  - Criação de um novo ponteiro para um dos nós divididos
  - Atualização dos intervalos das sub-regiões
  - Se o pai estiver cheio, dividi-lo também
    - Atualizar seu nó pai (e assim por diante)

- Obs. Em alguns casos, é necessário criar um novo nó pai
  - Caso o nó a ser dividido não possua pai
  - Esse nó se tornará o novo raiz

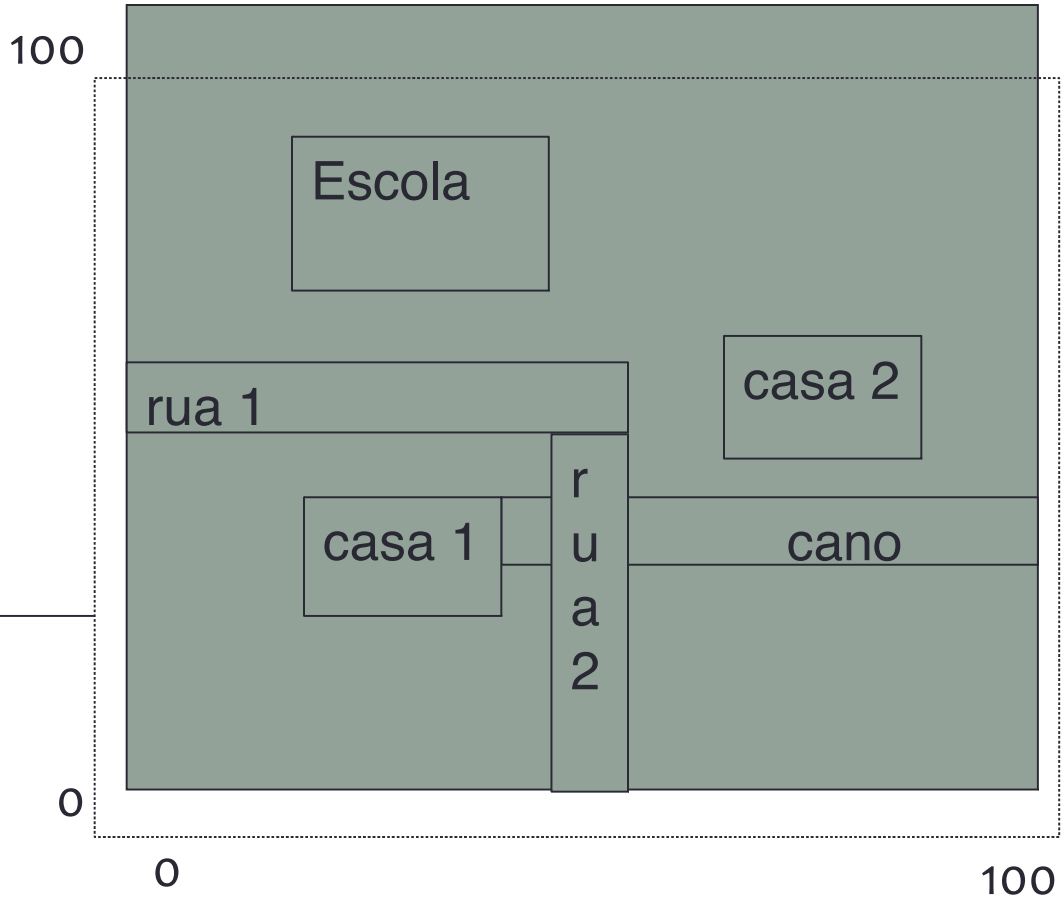
Exemplo

- Considere que um nó tenha espaço para 6 pontos apenas



Exemplo

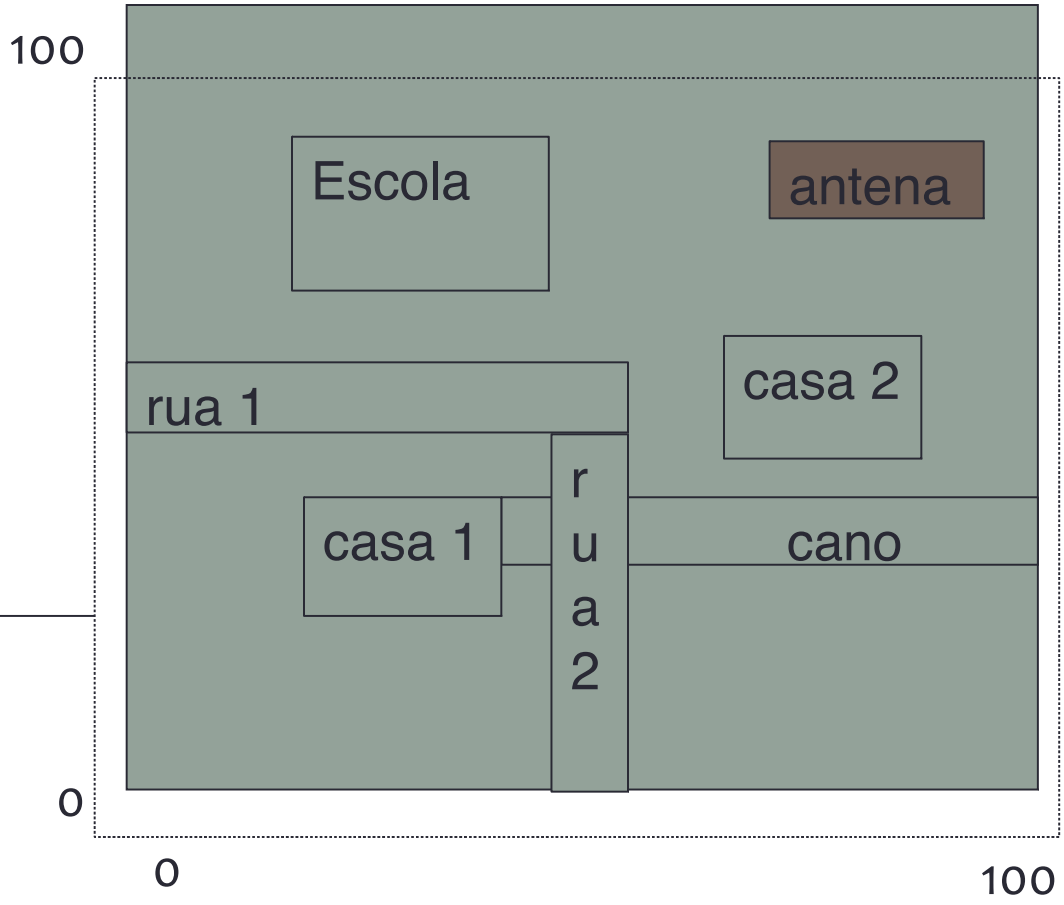
- Considere que um nó tenha espaço para 6 pontos apenas
- Nesse caso
  - a árvore tem apenas um nó (raiz e folha)



Cano	Casa 1	Casa2	Escola	Rua1	Rua2

Exemplo

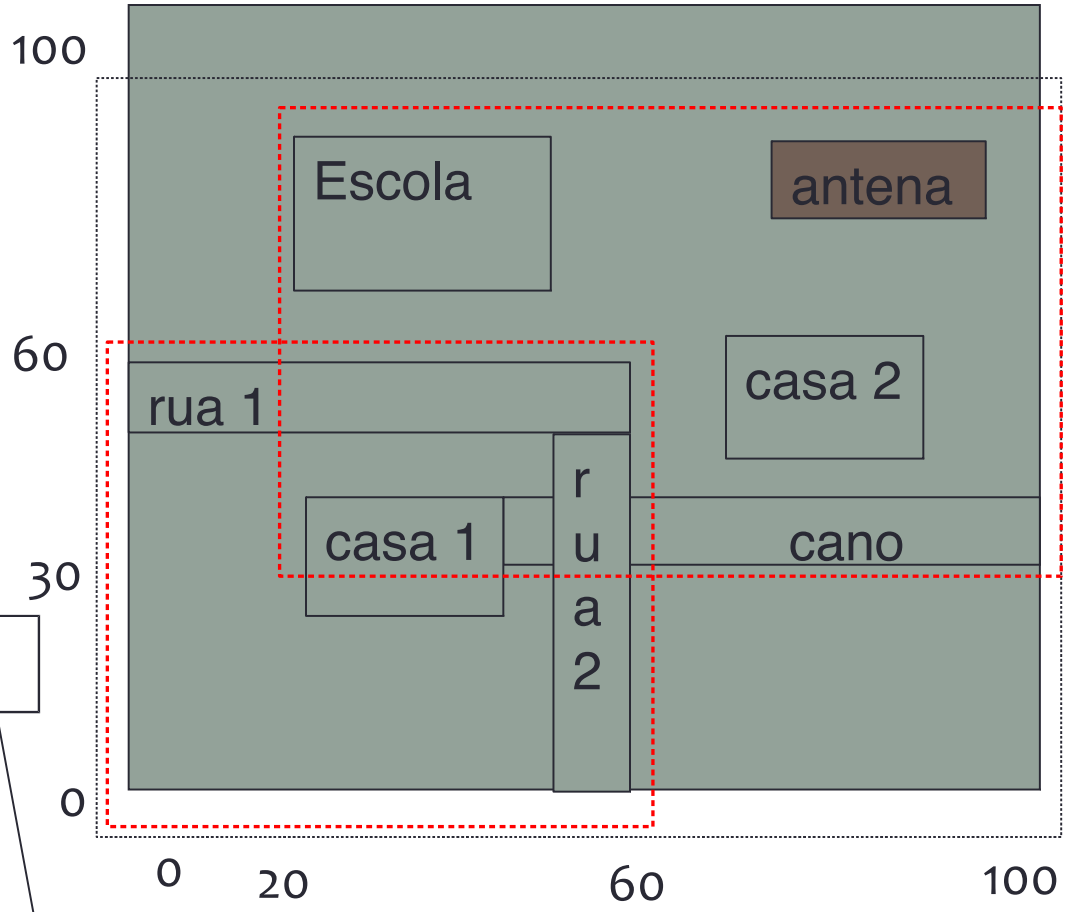
- E se fosse adicionado mais um ponto?
  - Uma antena



Cano	Casa 1	Casa2	Escola	Rua1	Rua2

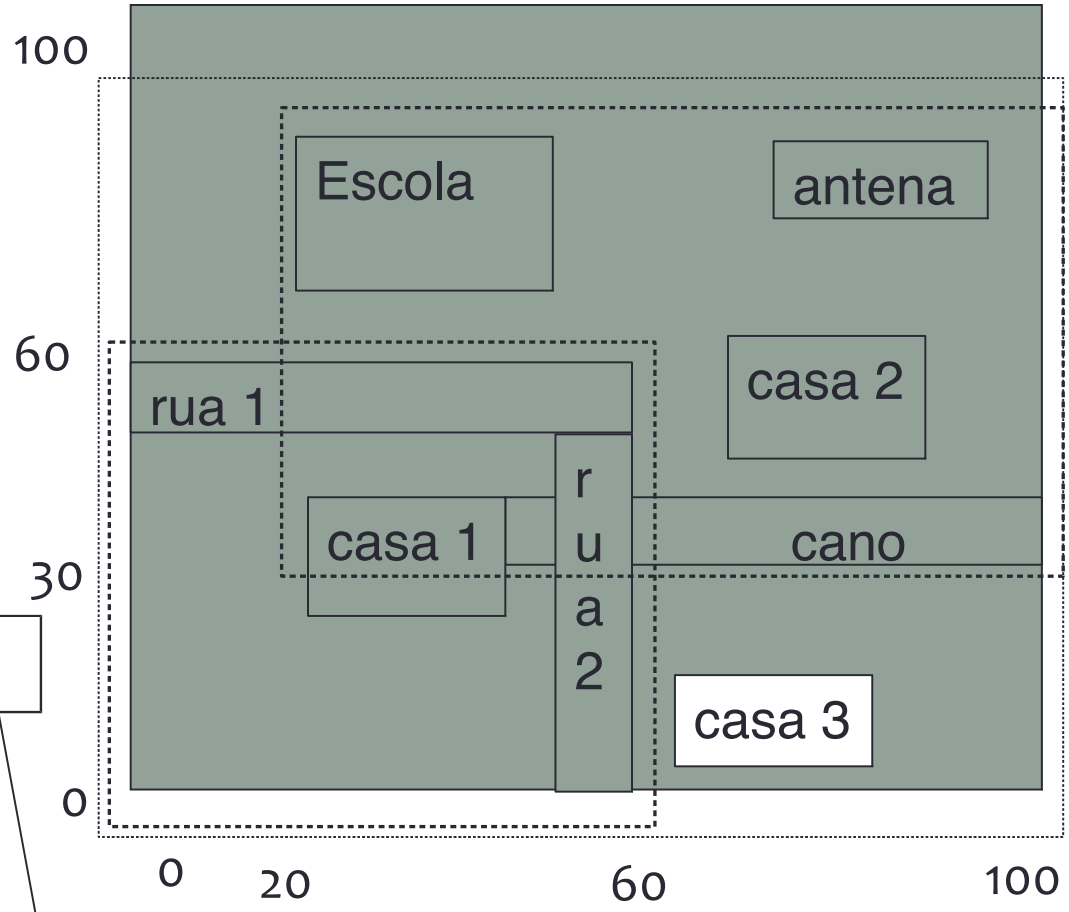
Exemplo

- Solução
  - Dividir a região em duas sub-regiões
    - evitar sobreposição
    - dividir de forma balanceada
    - Atualizar a árvore



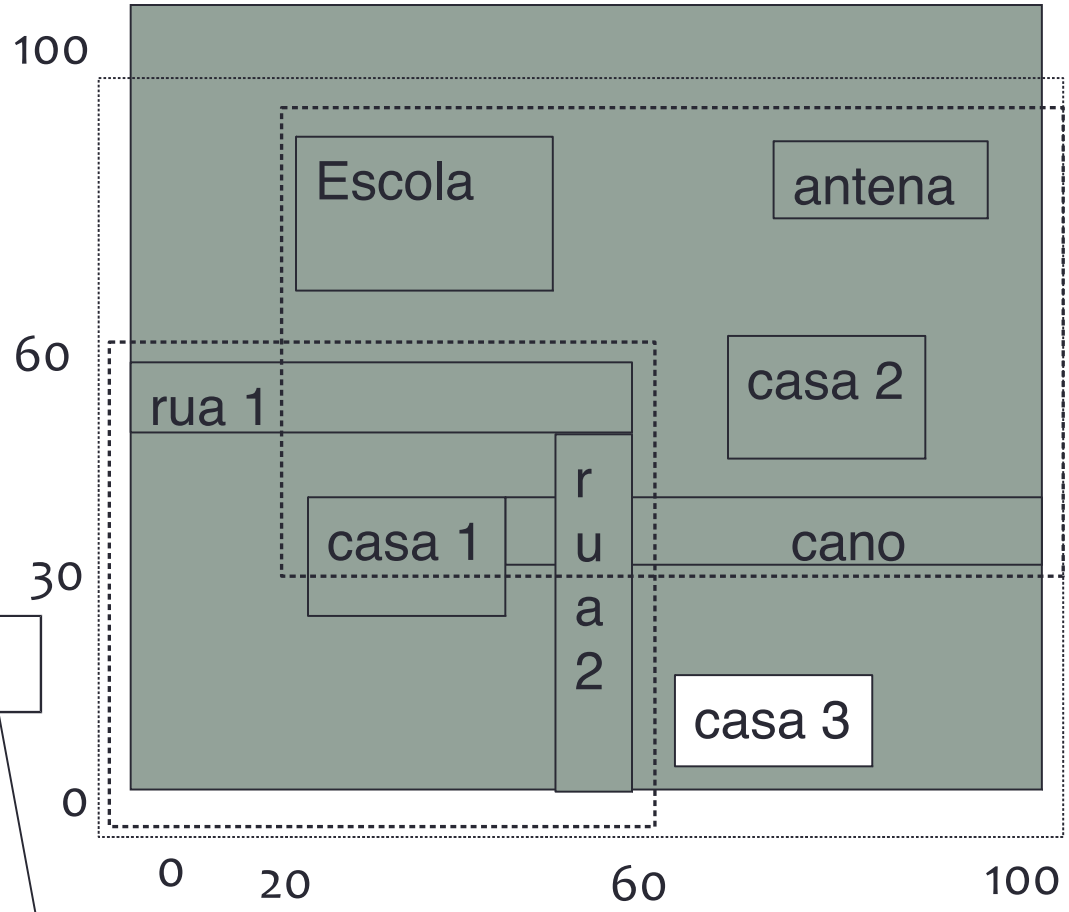
Exemplo

- Como indexar a casa 3?
  - Obs. Nenhumas das sub-regiões mapeia a área ocupada pela casa



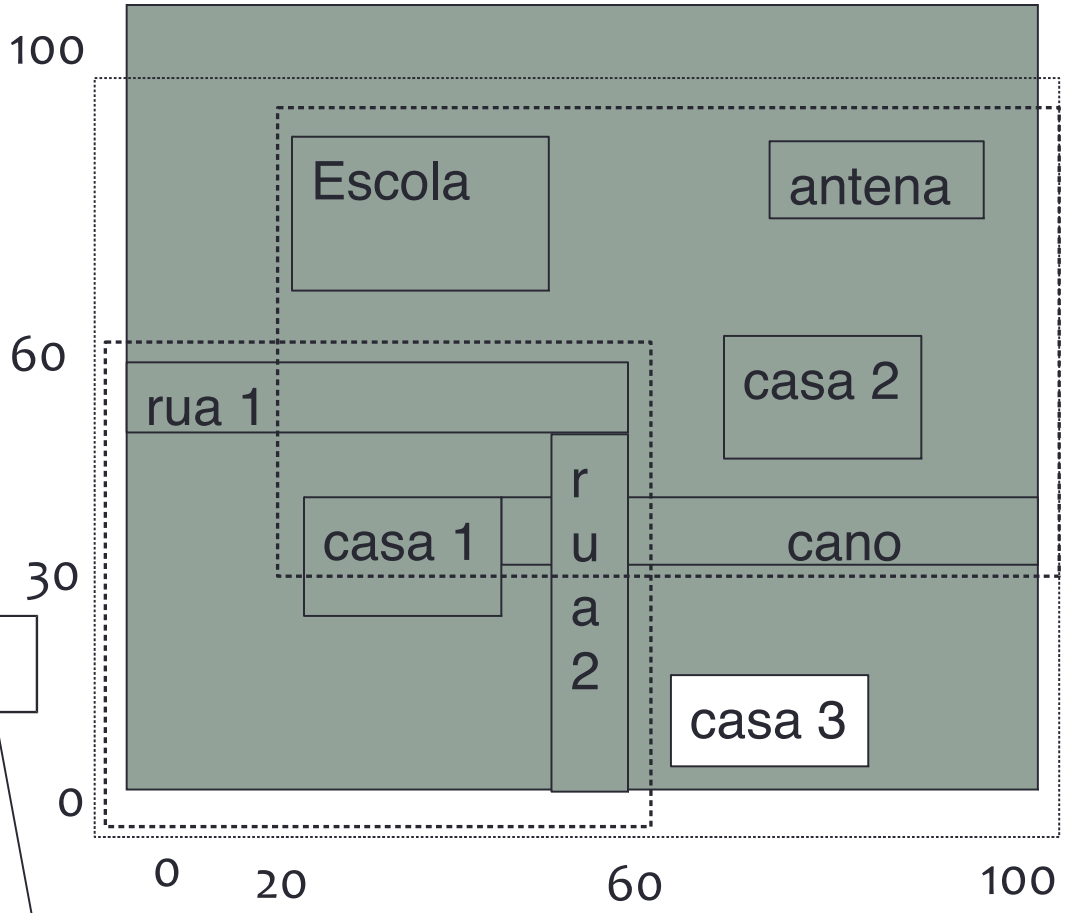
Exemplo

- Solução
  - Deve-se aumentar a área de cobertura de alguma das sub-regiões
  - Inserir o ponto nessa região



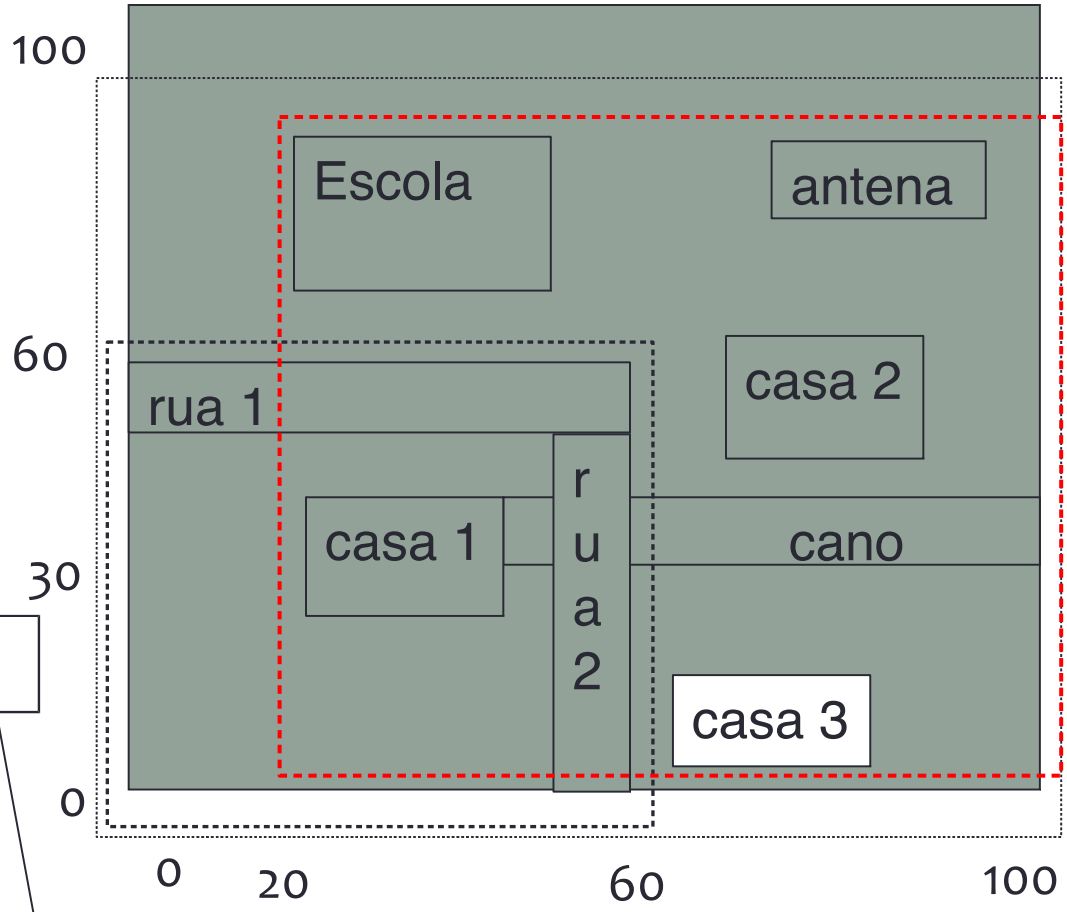
Exemplo

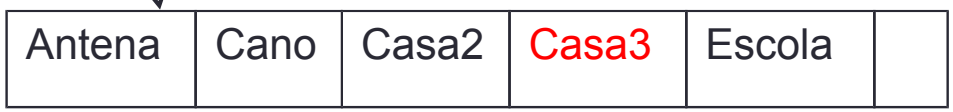
- Qual região aumentar?
  - A que representar o menor aumento



Exemplo

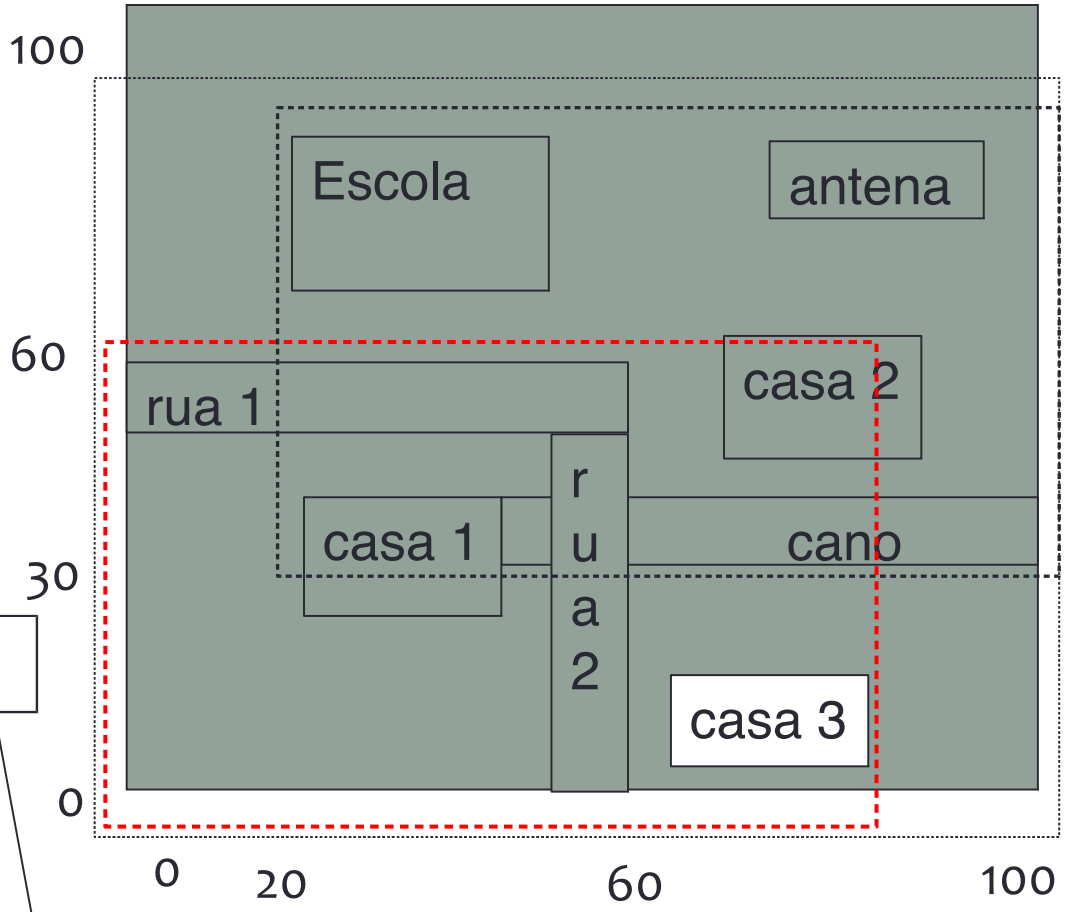
- Qual região aumentar?
  - Aumentando a região da direita
    - Aumento de área em 2.000 un.





Exemplo

- Qual região aumentar?
  - Aumentando a região da esquerda
    - Aumento de área em 1.200 un.



Buscas em árvores R

- Ideal para consultas do tipo “Onde eu estou?”
- O processamento começa pelo nó raiz
- Nó interno
  - Procurar pela sub-região (sub-regiões) onde o ponto se encaixe
  - Visitar cada uma dessas regiões
  - Repetir do início, até encontrar os nós folhas
- Nó folha
  - Contém
    - os registros mapeados naquela região ou
    - Ponteiros para os registros

Análise

- Custo de atualização
  - Maior
    - Se for importante evitar sobreposição de regiões
- Custo de acesso
  - Menor
    - Se regiões não forem sobrepostas

- Pontos importantes
  - Ideais para consultas do tipo “onde eu estou?”
  - Consultas de vizinho mais próximo podem ser efetuadas sem o parâmetro d

Análise

- Índices multidimensionais
  - Pensados para consultas envolvendo mais de uma dimensão
  - Custo de acesso
    - Aumenta
      - se poucas das dimensões mapeadas forem usadas em critérios de busca
  - Desempenho cai quando muitas dimensões são mapeadas
    - Tanto no acesso
    - Quanto na atualização
  - Por isso esses índices são mais indicados para indexar dados de localização (coordenadas x, y)

Análise

- A análise a seguir considera o desempenho dos índices estudados para os principais tipos de consulta
  - envolvendo duas dimensões (x,y)

- Índices kd e quadrante não serão comparados
  - Pois são mais utilizados em memória primária

Análise

- Consulta por correspondência exata
  - Objeto existente em (10,10)

Índice	Obs.
Simples	Bom
Composto	Bom
Múltiplas chaves	Bom
Grade	Ótimo. Acesso direto.
Hash	Ótimo. Acesso direto.
Árvore R	Regular. Árvore mais profunda que uma árvore B+. Mais de um filho pode ser acessado

Análise

- Consulta por correspondência parcial
  - Objetos existentes na coordenada X = 10
  - Objetos existentes na coordenada Y = 10

Índice	Obs.
Simples	Bom
Composto	Bom. Só funciona com X = valor
Múltiplas chaves	Bom. Só funciona com X = valor
Grade	Bom, desde que os buckets sejam bem preenchidos
Hash	Bom, desde que os buckets sejam bem preenchidos
Árvore R	Regular. Árvore mais profunda que uma árvore B+. Mais de um filho pode ser acessado

Análise

- Consulta por intervalo
  - Objetos localizados em X > 10 ?
  - Objetos localizados em Y > 10 ?
  - Objetos localizados depois de (10,10) ?

Índice	Obs.
Simples	Desempenho menor se fornecidos filtros para X e Y
Composto	Bom. Só funciona com X > valor
Múltiplas chaves	Bom. Não funciona com Y > valor
Grade	Bom, desde que os buckets sejam bem preenchidos
Hash	Inviável, pois só funciona com consultas por equivalência
Árvore R	Melhor para intervalos fechados. (ex. x>10 e x<20)

Análise

- Consulta pelo vizinho mais próximo
  - Objeto mais próximo de (10,10)?

Índice	Obs.
Simples	Precisa parâmetro d. Usa consulta por intervalo
Composto	Inviável, pois precisa do parâmetro d.
Múltiplas chaves	Precisa parâmetro d. Usa consulta por intervalo
Grade	Bom, desde que os buckets sejam bem preenchidos. Troca parâmetro d pela busca nos buckets vizinhos.
Hash	Inviável, pois precisa do parâmetro d (consulta por intervalo).
Árvore R	Bom. Não precisa de parâmetro d.

Análise

- Consulta do tipo onde eu estou?
  - Que objeto contém o ponto (10,10)?

Índice	Obs.
Simples	Ruim. Requer índice de quatro dimensões.
Composto	Ruim. Requer índice de quatro dimensões.
Múltiplas chaves	Ruim. Requer índice de quatro dimensões.
Grade	Ruim. Requer índice de quatro dimensões.
Hash	Inviável, pois depende de uma consulta por intervalo.
Árvore R	Bom.

Exercício

- Indexar os objetos da tabela em um índice quad-tree

- Suponha que tenha espaço para dois registros em cada bloco

- Considerar idade e salário para fins de indexação
  - Teto para idade: 100 anos
  - Teto para salário: 5.000

nome	idade	salário
João	30	3.000
Marcos	40	4.500
Janete	28	4.000
Zé	22	1.300
Souza	60	5.000
Pedro	44	1.500
Carla	40	2.300