Estrutura híbrida (ou Doidona)

Como tratar o elemento?

- Ao invez de criar uma variável int elem por exemplo para ser o elemento, cria-se um objeto para se referir ao objeto

A inserção é feita recursivamente.

```
class Doidona{
    Objeto T1[6];
    Objeto T3[5];
    Arvore arvoreBinaria;
    Lista lista;
    AVL arvoreAVL;
}
```

Exemplo:
Tenho uma estrutura “Doidona” de uma tabela hash T1, que se faltar espaço é dividido em 3 estruturas.
- A 1a é uma tabela hash com rehash, que se faltar espaço leva para uma árvore binaria;
- A 2a é uma lista flexível que adiciona os elementos sempre no final da lista;
- E a 3a é uma árvore AVL

```
class Objeto{
    int elem;
    public Objeto(int x){
        this.elem = x;
    }
}

class Arvore{
    Objeto elem;
    Arvore dir, esq;
}

class Celula{
    Objeto elem;
    Celula prox;
}

class Lista{
    Celula primeira;
    Celula ultima;
}

class AVL{
    Objeto elem;
    AVL dir, esq;
}
```

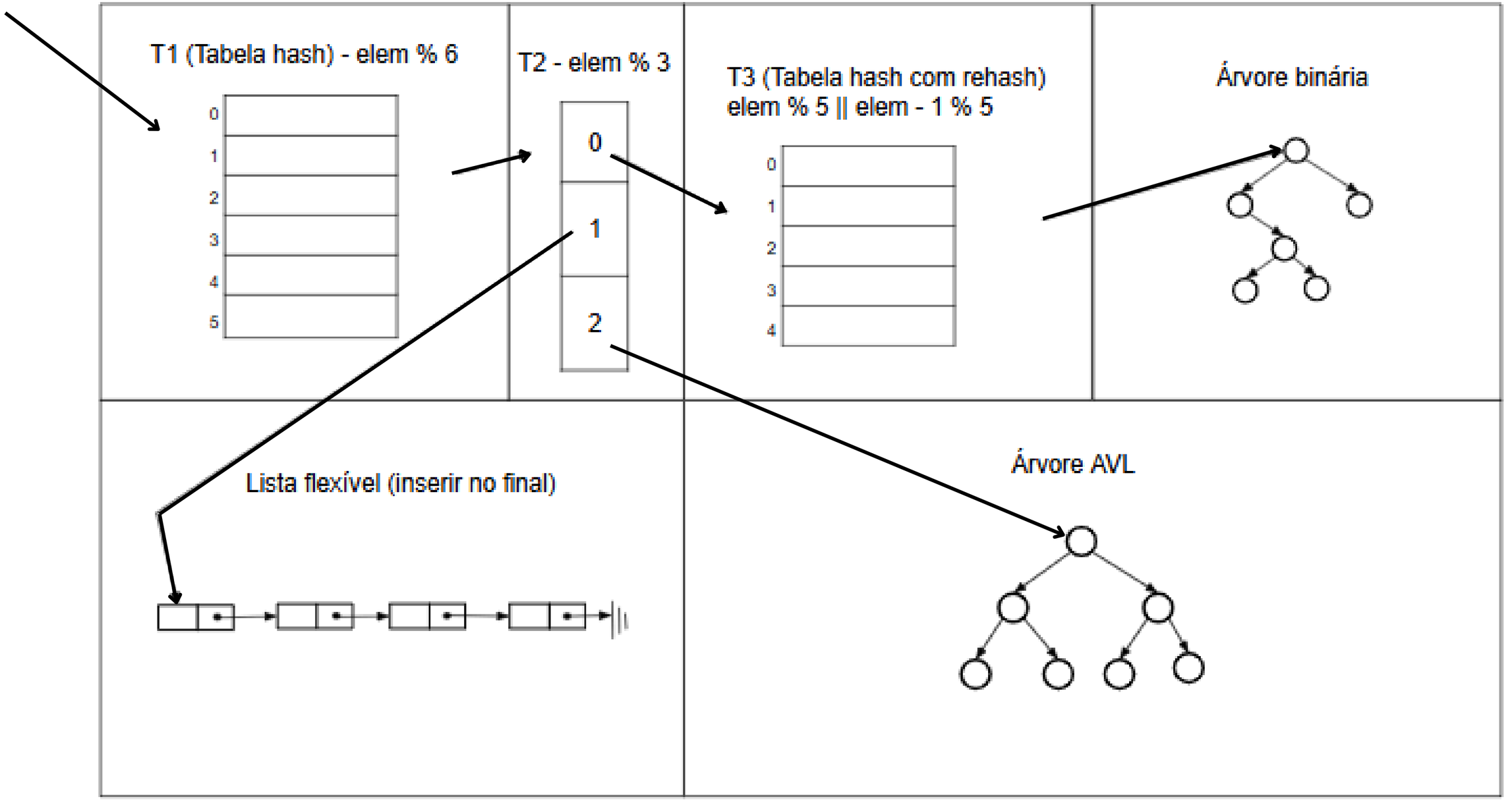
```java
public void inserir(Objeto objeto){
    int elem = objeto.elem;
    int i = elem % 6; //hash da T1
    if(T1[i] == null){
        T1[i] = elem;
    }else{
        i = elem % 3; //hash da T2
        if(i == 0){
            i = elem % 5; //hash da T3
            if(T3[i] == null){
                T3[i] = elem;
            } else {
                i = elem - 1 % 5; //rehash da T3
                if(t3[i] == null){
                    t3[i] = elem;
                } else {
                    arvoreBinaria.inserir(objeto);
                }
            }
        }else if (i == 1){
            lista.inserirFim(objeto);
        }else{ //if (i == 2)
            arvoreAVL.inserir(objeto);
        }
    }
}
```

```java
public boolean search(int elem){
    int i = elem % 6; //hash da T1
    if(T1[i] == null){
        return false;
    } else if(T1[i].elem == elem){
        return true;
    } else {
        i = elem % 3; //hash da T2
        if(i == 0){
            i = elem % 5; //hash da T3
            if(T3[i] == null){
                return false;
            } else if (T3[i].elem == elem){
                return true;
            } else {
                i = elem - 1 % 5; //rehash da T3
                if(T3[i] == null){
                    return false;
                } else if(T3[i].elem == elem){
                    return true;
                }else {
                    arvoreBinaria.search(elem);
                }
            }
        }else if (i == 1){
            lista.search(elem);
        }else{ //if (i == 2)
            arvoreAVL.search(elem);
        }
    }
}
```

```java
public void inserir(Objeto objeto){
    int elem = objeto.elem;
    int i = elem % 6; //hash da T1
    if(T1[i] == null){
      T1[i] = elem;
    }else{
      i = elem % 3; //hash da T2
      if(i == 0){
        i = elem % 5; //hash da T3
        if(T3[i] == null){
          T3[i] = elem;
        } else {
          i = elem - 1 % 5; //rehash da T3
          if(t3[i] == null){
            t3[i] = elem;
          } else {
            arvoreBinaria.inserir(objeto);
          }
        }
      }else if (i == 1){
        lista.inserirFim(objeto);
      }else{ //if (i == 2)
        arvoreAVL.inserir(objeto);
      }
    }
  }
```

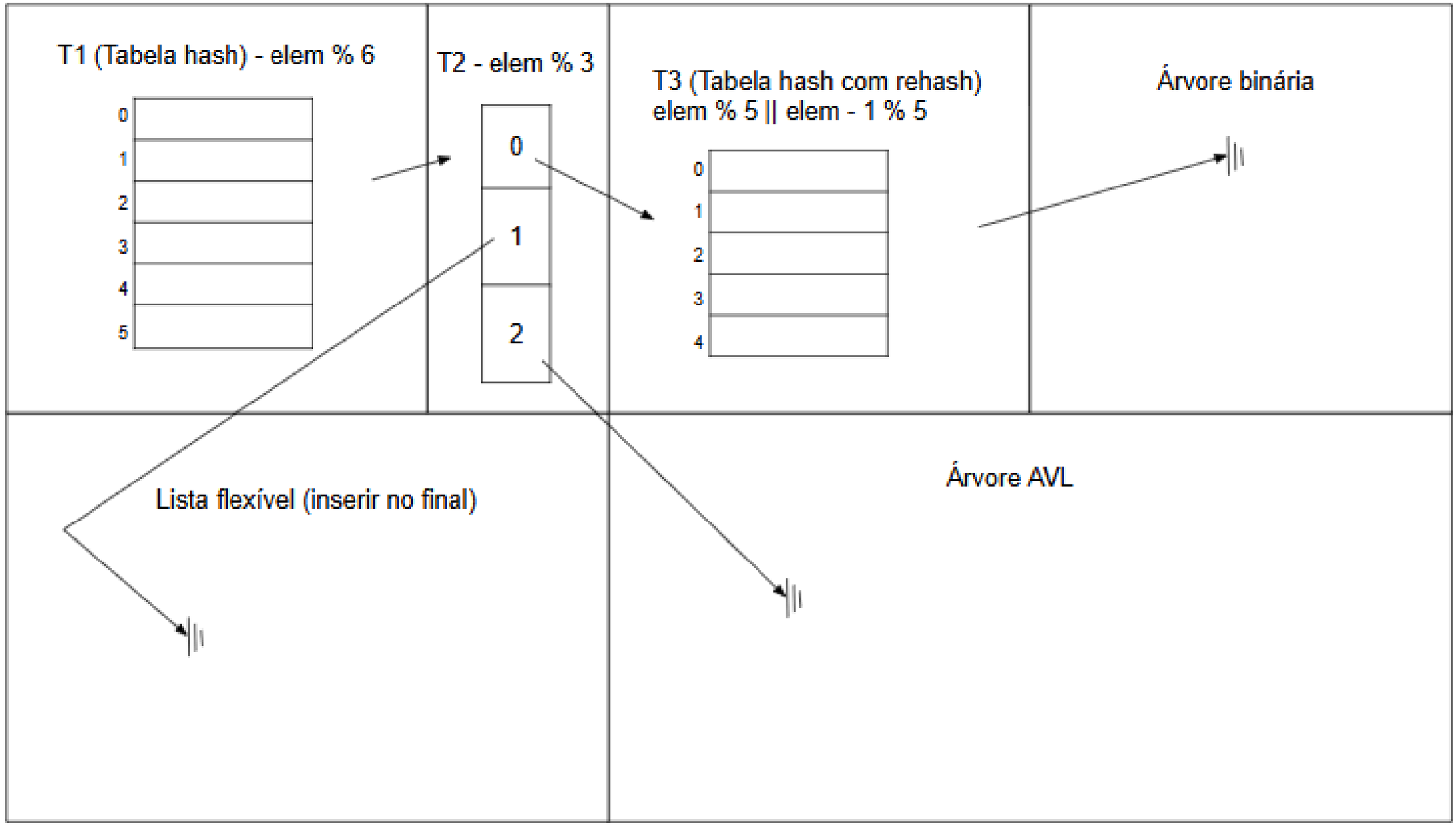
```java
public boolean search(int elem){
    int i = elem % 6; //hash da T1
    if(T1[i] == null){
      return false;
    } else if(T1[i].elem == elem){
      return true;
    } else {
      i = elem % 3; //hash da T2
      if(i == 0){
        i = elem % 5; //hash da T3
        if(T3[i] == null){
          return false;
        } else if (T3[i].elem == elem){
          return true;
        } else {
          i = elem - 1 % 5; //rehash da T3
          if(T3[i] == null){
            return false;
          } else if(T3[i].elem == elem){
            return true;
          }else {
            arvoreBinaria.search(elem);
          }
        }
      }else if (i == 1){
        lista.search(elem);
      }else{ //if (i == 2)
        arvoreAVL.search(elem);
      }
    }
  }
```

Exemplo:

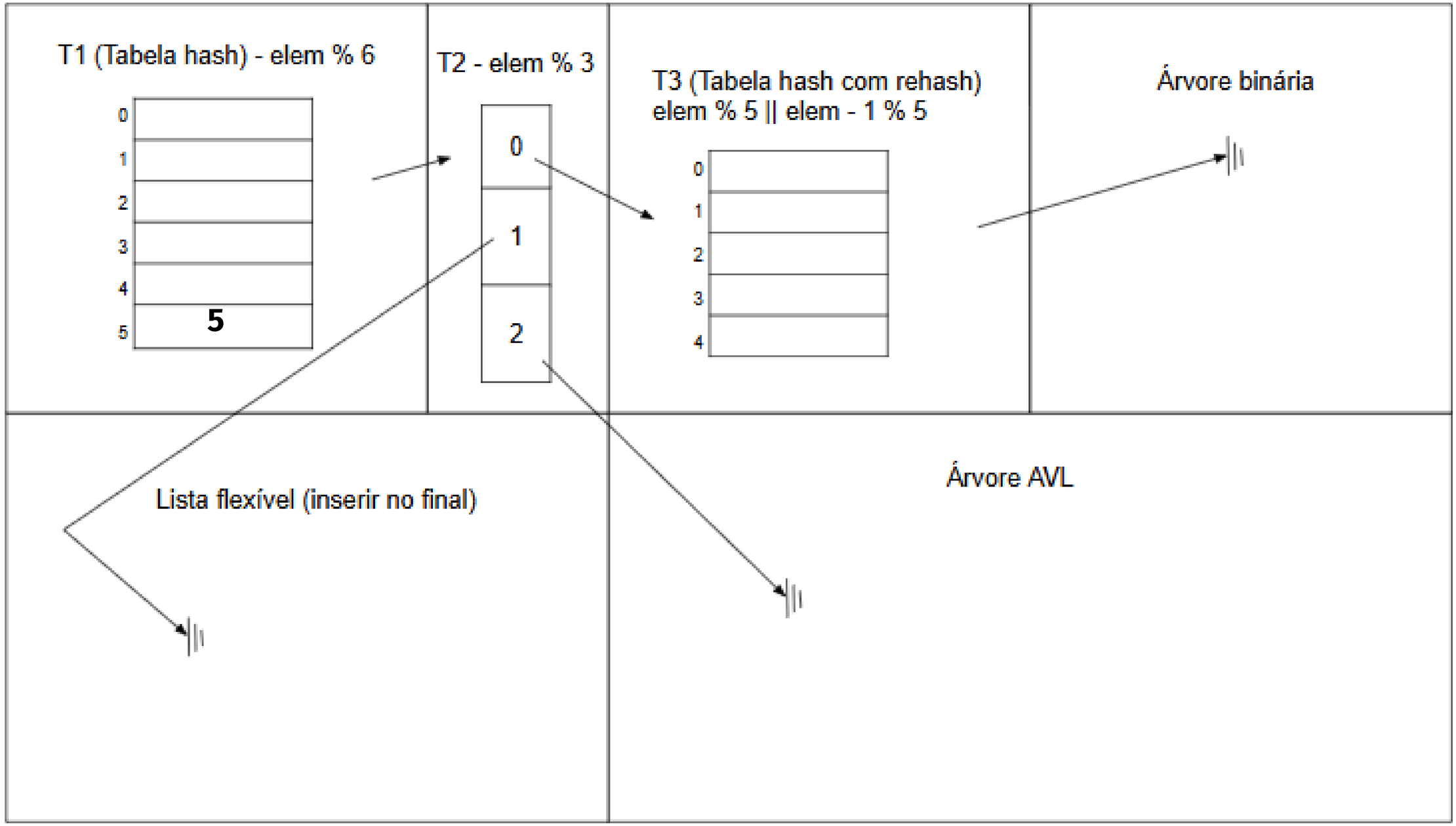

T1 (Tabela hash) - elem % 6
0
1
2
3
4
5
T2 - elem % 3
0
1
2
T3 (Tabela hash com rehash)
elem % 5 || elem - 1 % 5
0
1
2
3
4
Árvore binária
Lista flexível (inserir no final)
Árvore AVL

Tente inserir os seguintes elementos:

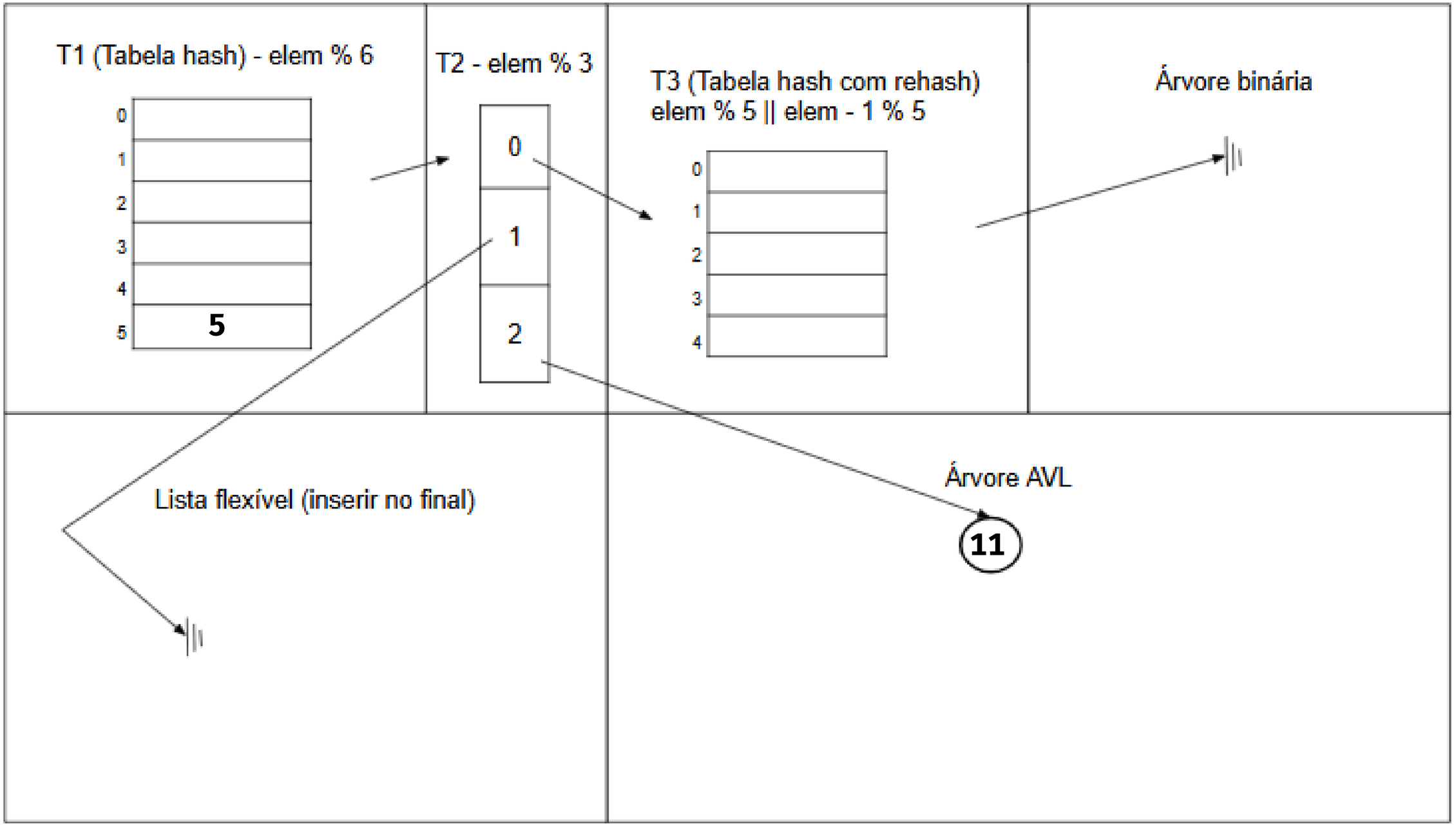
5, 11, 3, 9, 18, 21, 33, 27, 35, 41, 36, 4, 8, 10, 20, 24, 48, 50, 25, 72, 19



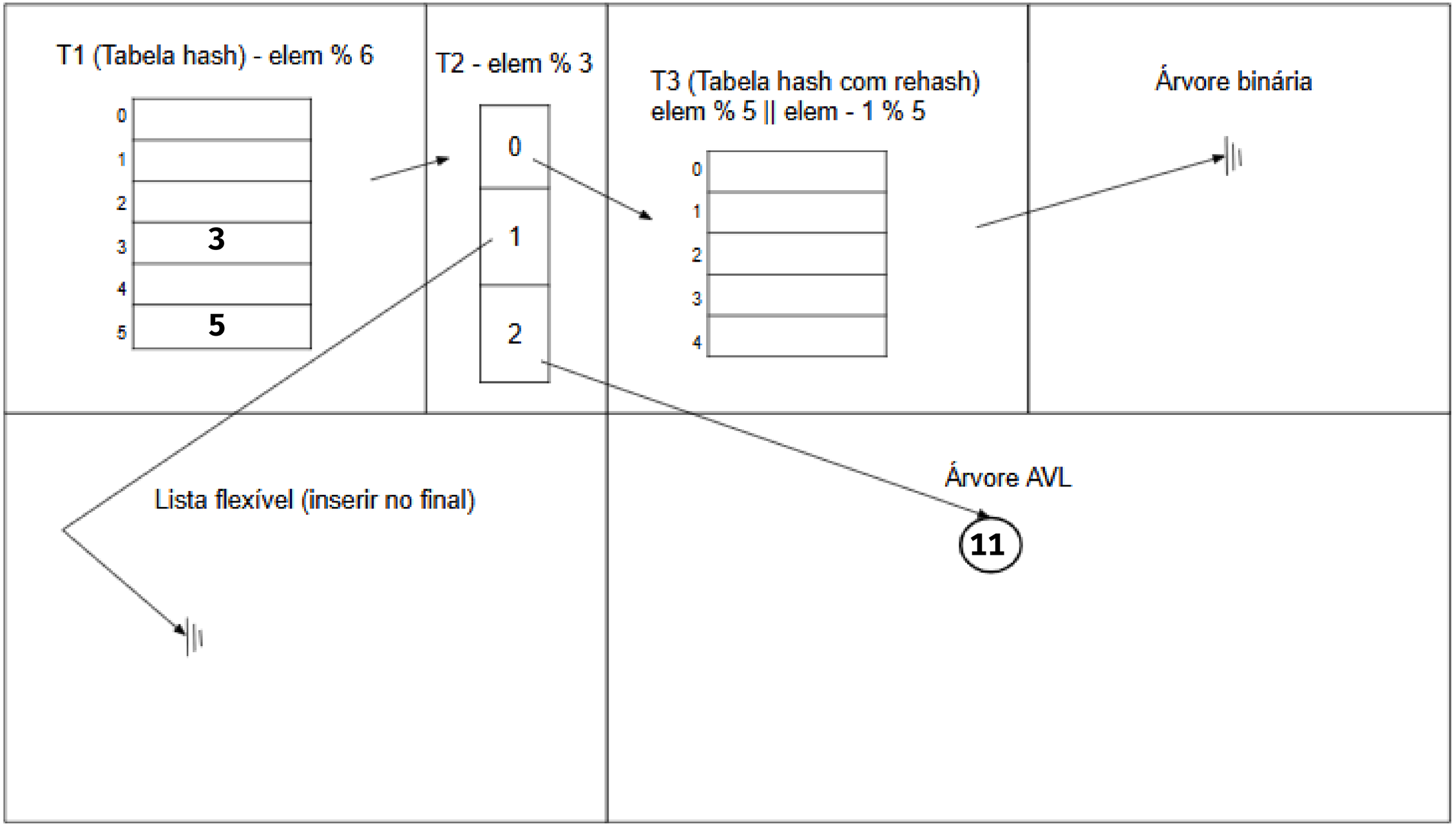
5, 11, 3, 9, 18, 21, 33, 27, 35, 41, 36, 4, 8, 10, 20, 24, 48, 50, 25, 72, 19



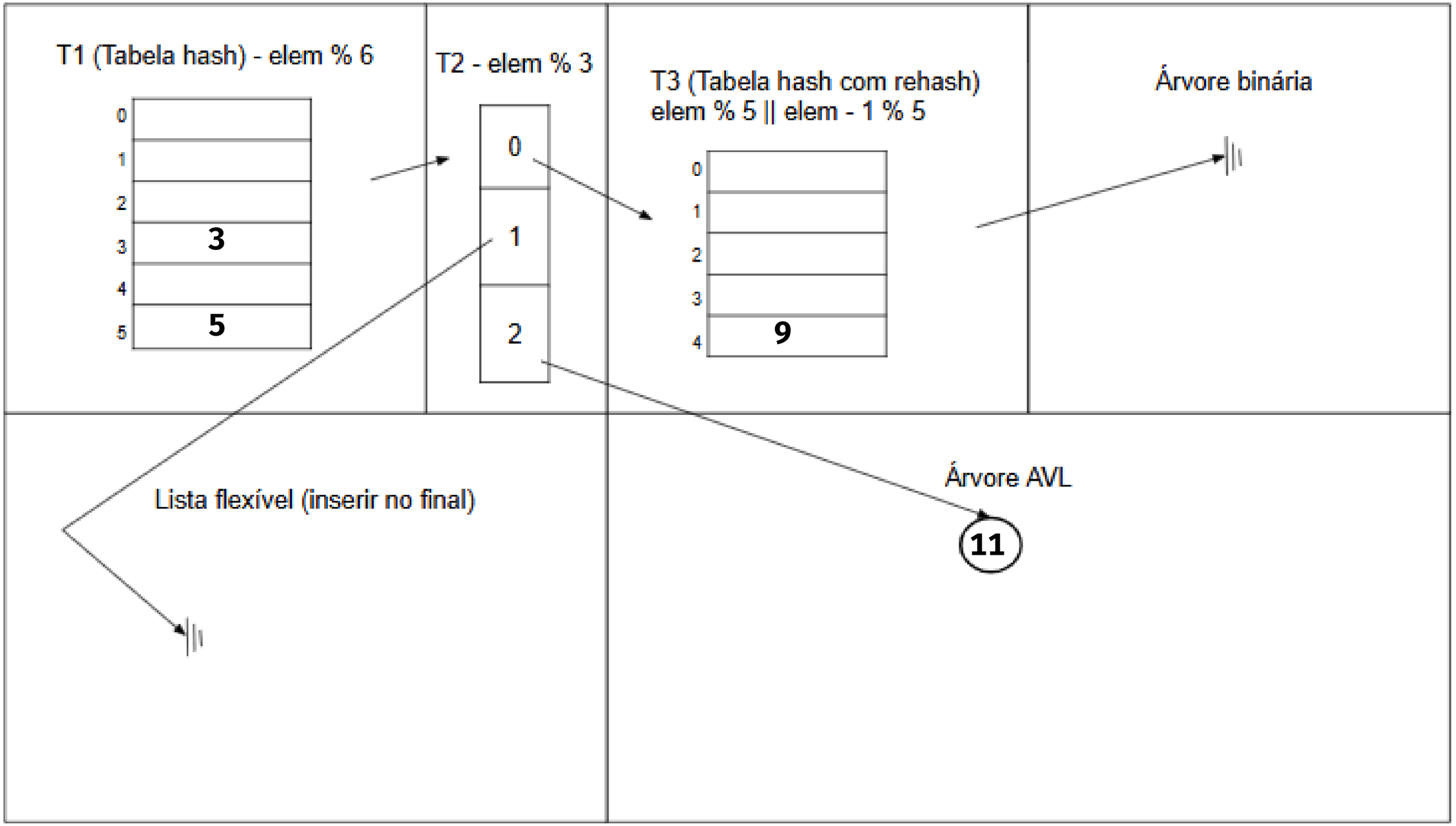
5, 11, 3, 9, 18, 21, 33, 27, 35, 41, 36, 4, 8, 10, 20, 24, 48, 50, 25, 72, 19



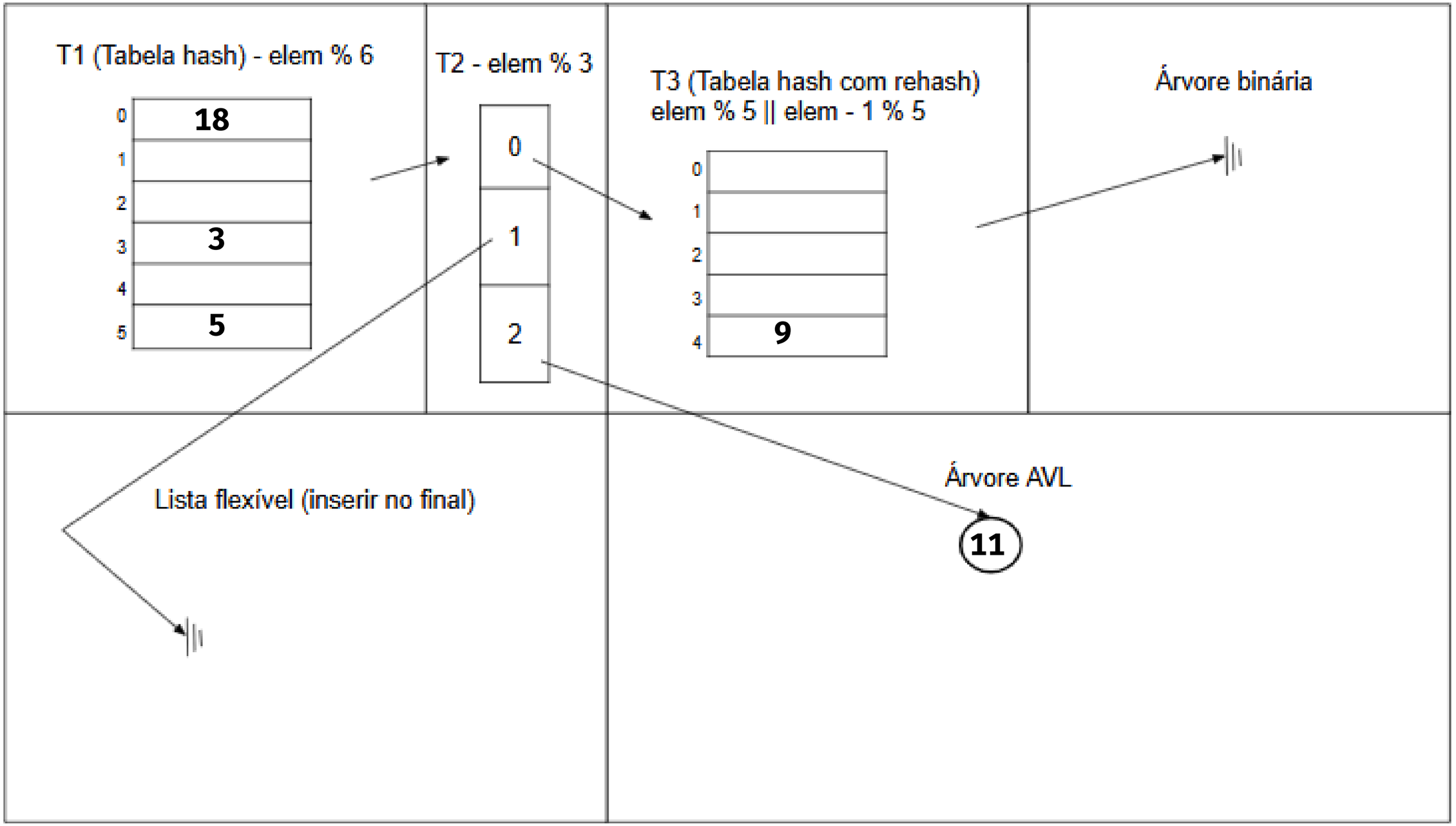
~~3~~, ~~11~~, ~~3~~, 9, 18, 21, 33, 27, 35, 41, 36, 4, 8, 10, 20, 24, 48, 50, 25, 72, 19



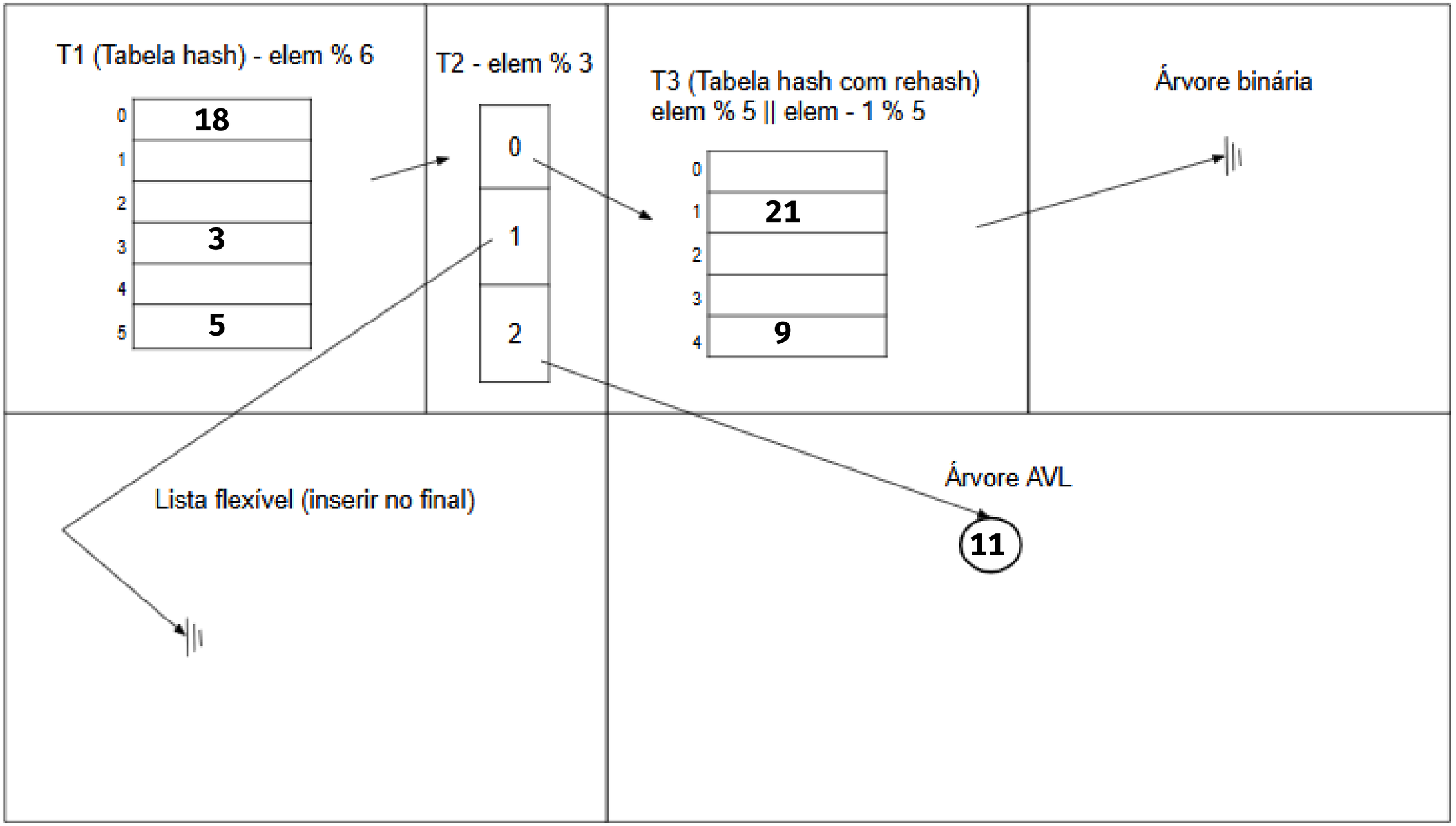
~~3~~, ~~11~~, ~~3~~, ~~9~~, 18, 21, 33, 27, 35, 41, 36, 4, 8, 10, 20, 24, 48, 50, 25, 72, 19



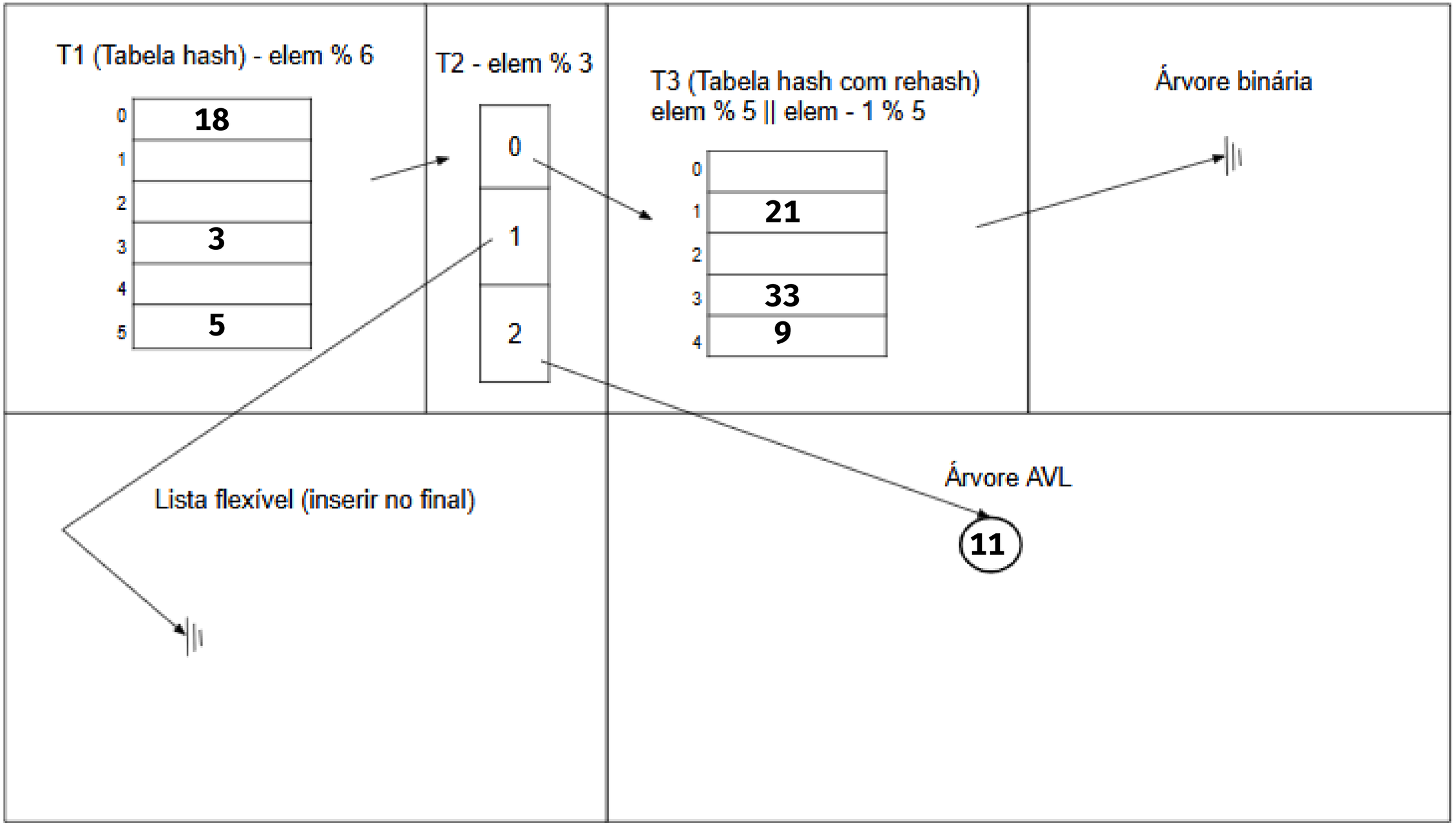
~~3~~, ~~11~~, ~~3~~, ~~9~~, ~~18~~, 21, 33, 27, 35, 41, 36, 4, 8, 10, 20, 24, 48, 50, 25, 72, 19



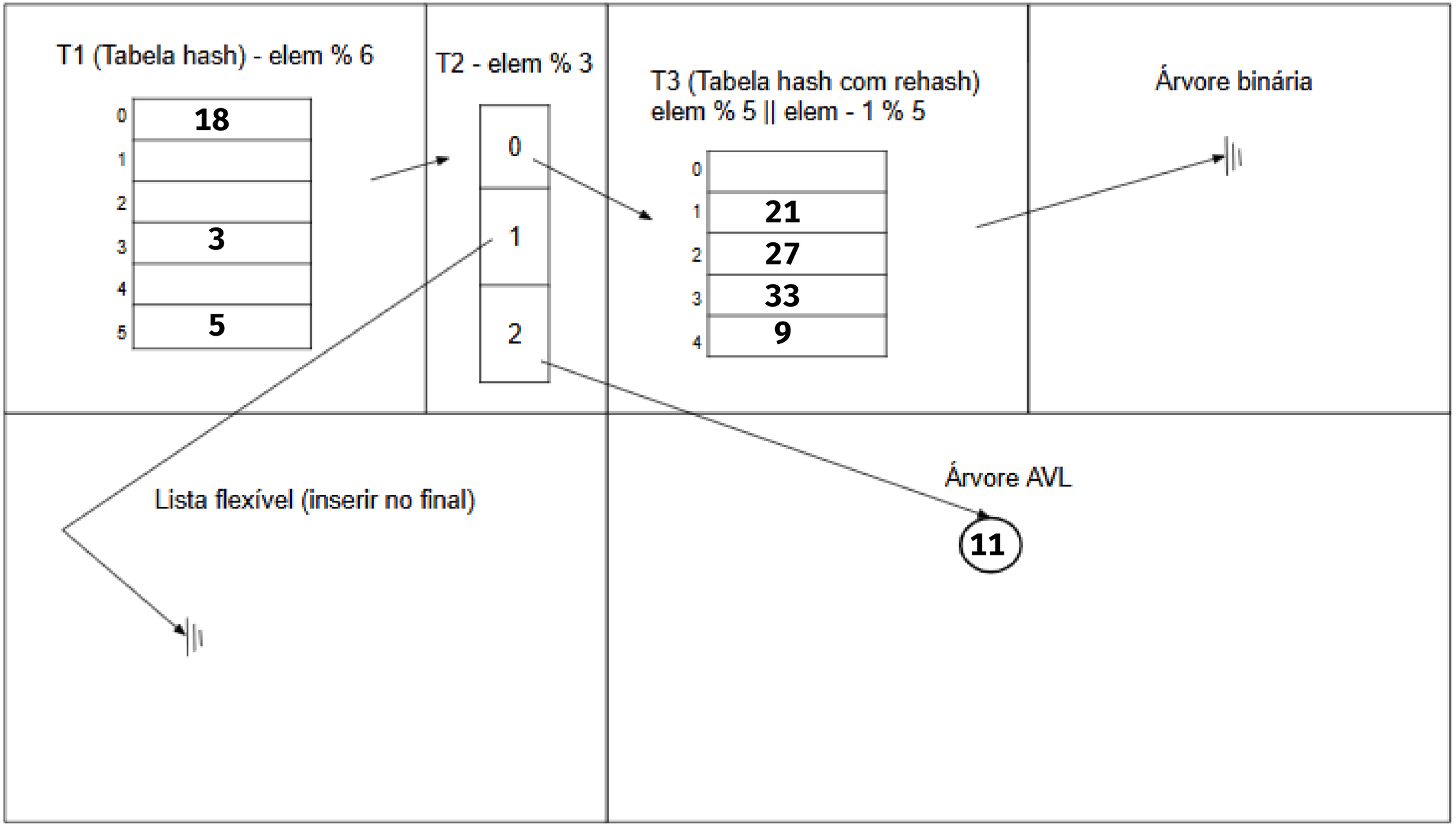
~~5~~, ~~11~~, ~~3~~, ~~9~~, ~~18~~, ~~21~~, 33, 27, 35, 41, 36, 4, 8, 10, 20, 24, 48, 50, 25, 72, 19



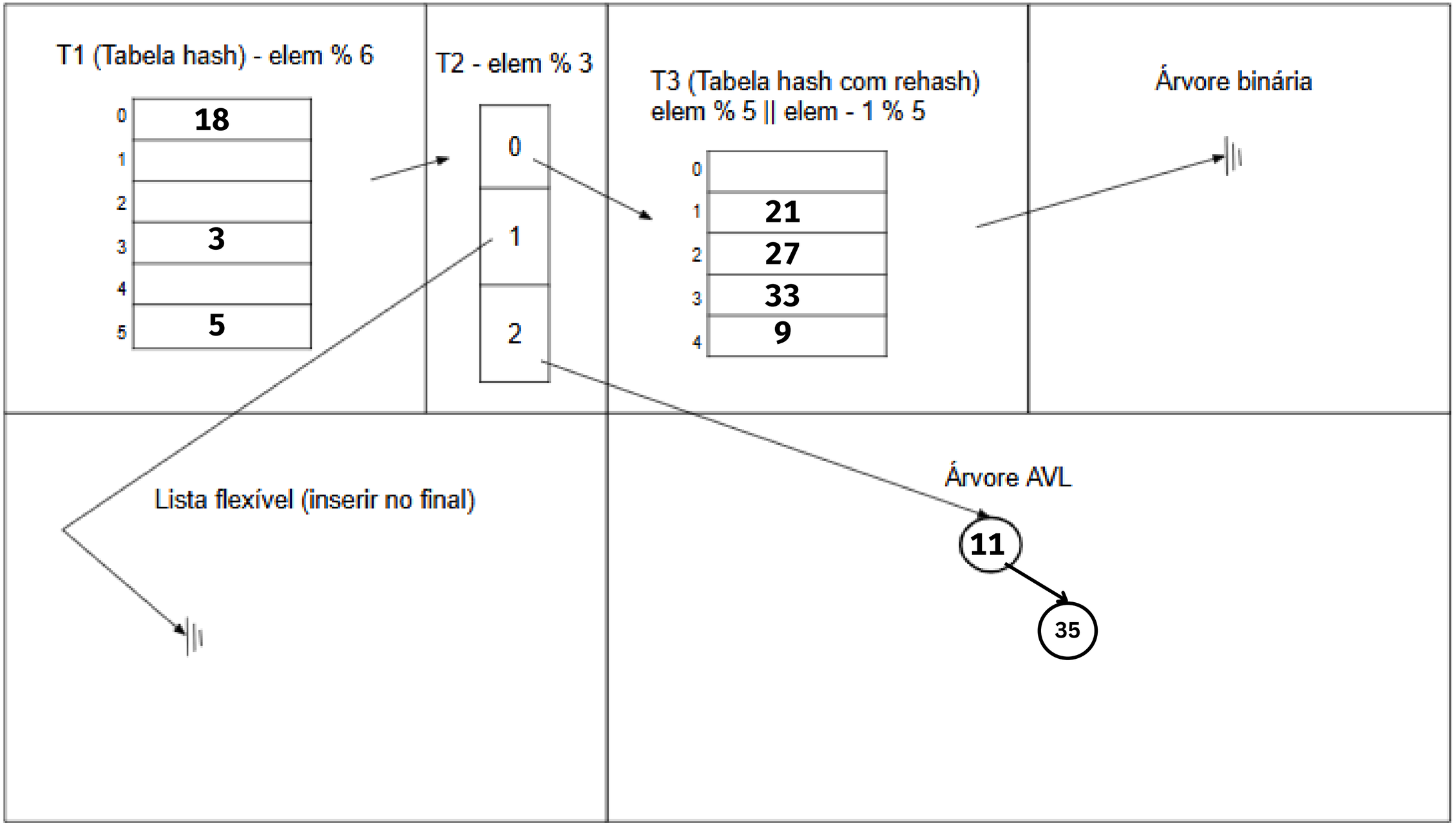
~~5~~, ~~11~~, ~~3~~, ~~9~~, ~~18~~, ~~21~~, ~~33~~, 27, 35, 41, 36, 4, 8, 10, 20, 24, 48, 50, 25, 72, 19



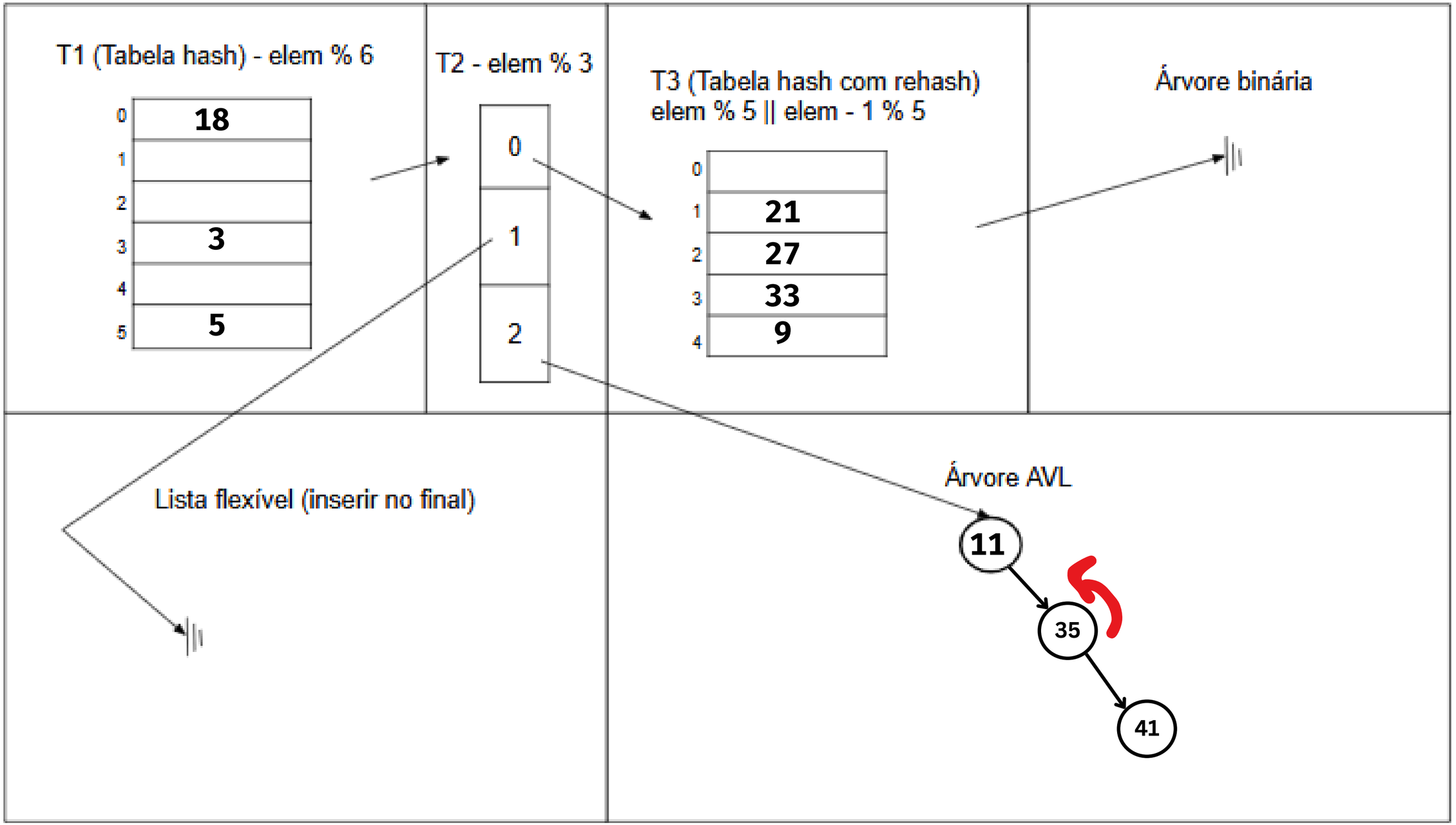
~~5~~, ~~11~~, ~~3~~, ~~9~~, ~~18~~, ~~21~~, ~~33~~, ~~27~~, 35, 41, 36, 4, 8, 10, 20, 24, 48, 50, 25, 72, 19



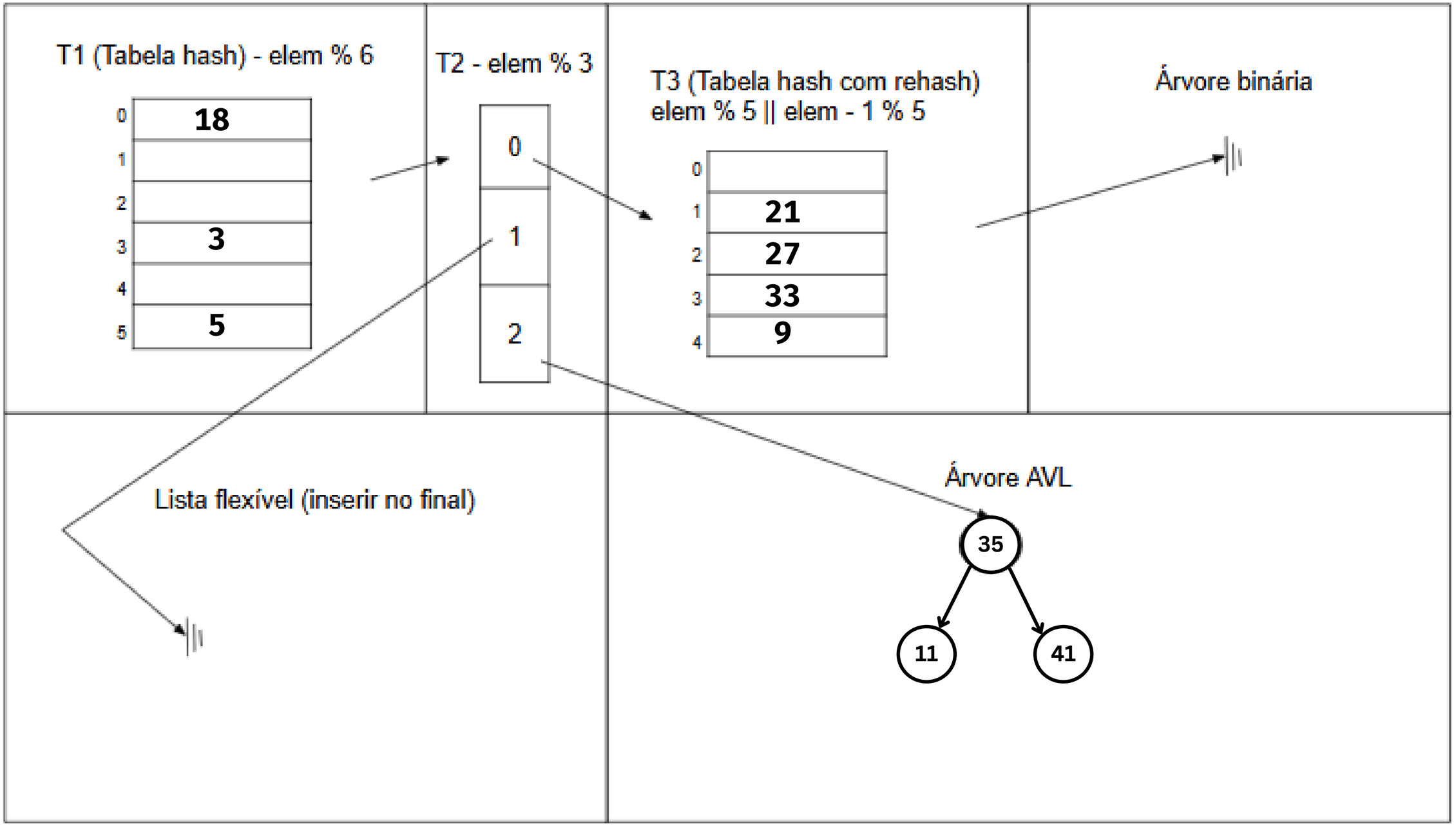
~~5~~, ~~11~~, ~~3~~, ~~9~~, ~~18~~, ~~21~~, ~~33~~, ~~27~~, 35, 41, 36, 4, 8, 10, 20, 24, 48, 50, 25, 72, 19



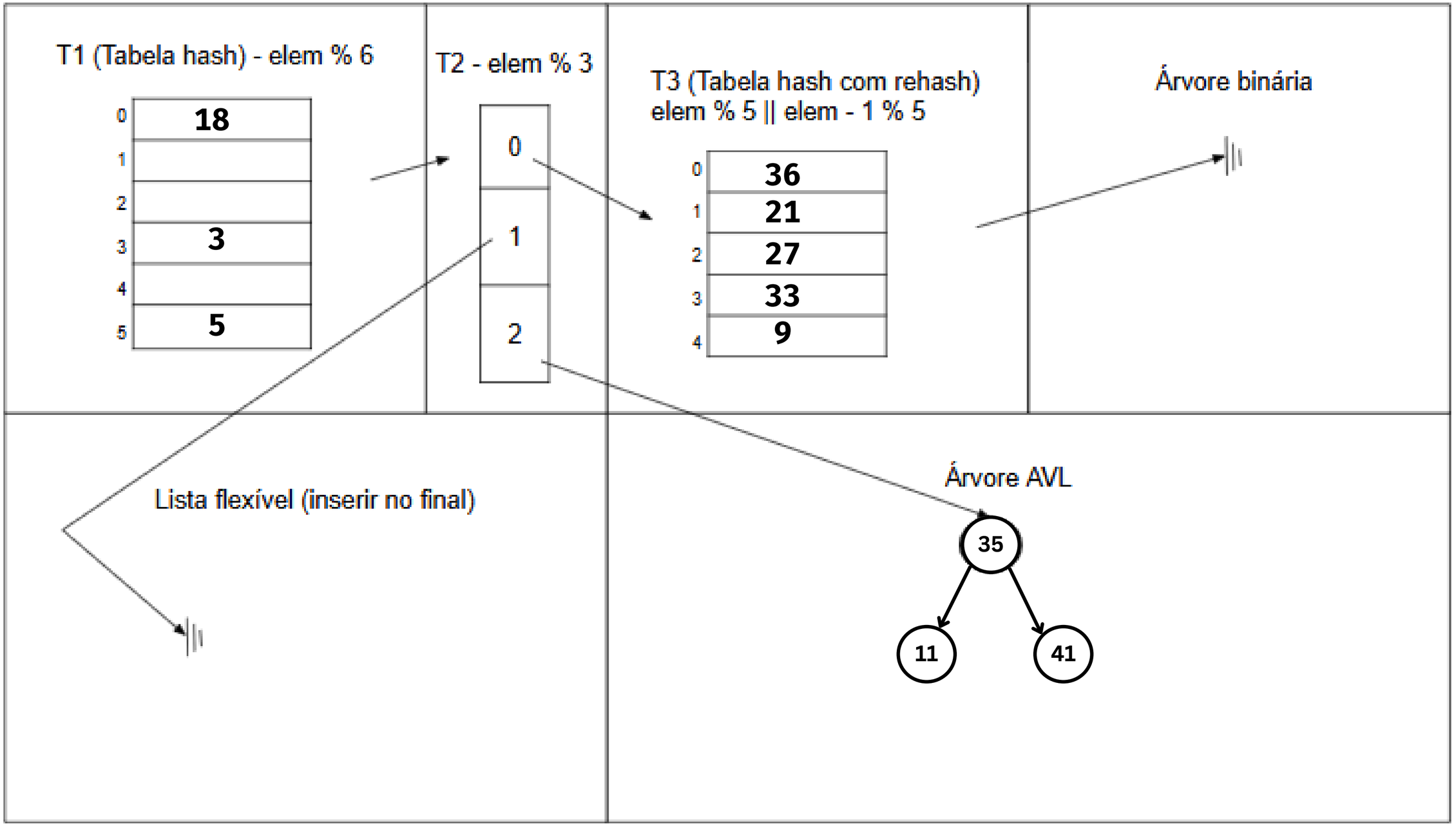
~~5~~, ~~11~~, ~~3~~, ~~9~~, ~~18~~, ~~21~~, ~~33~~, ~~27~~, ~~35~~, 41, 36, 4, 8, 10, 20, 24, 48, 50, 25, 72, 19



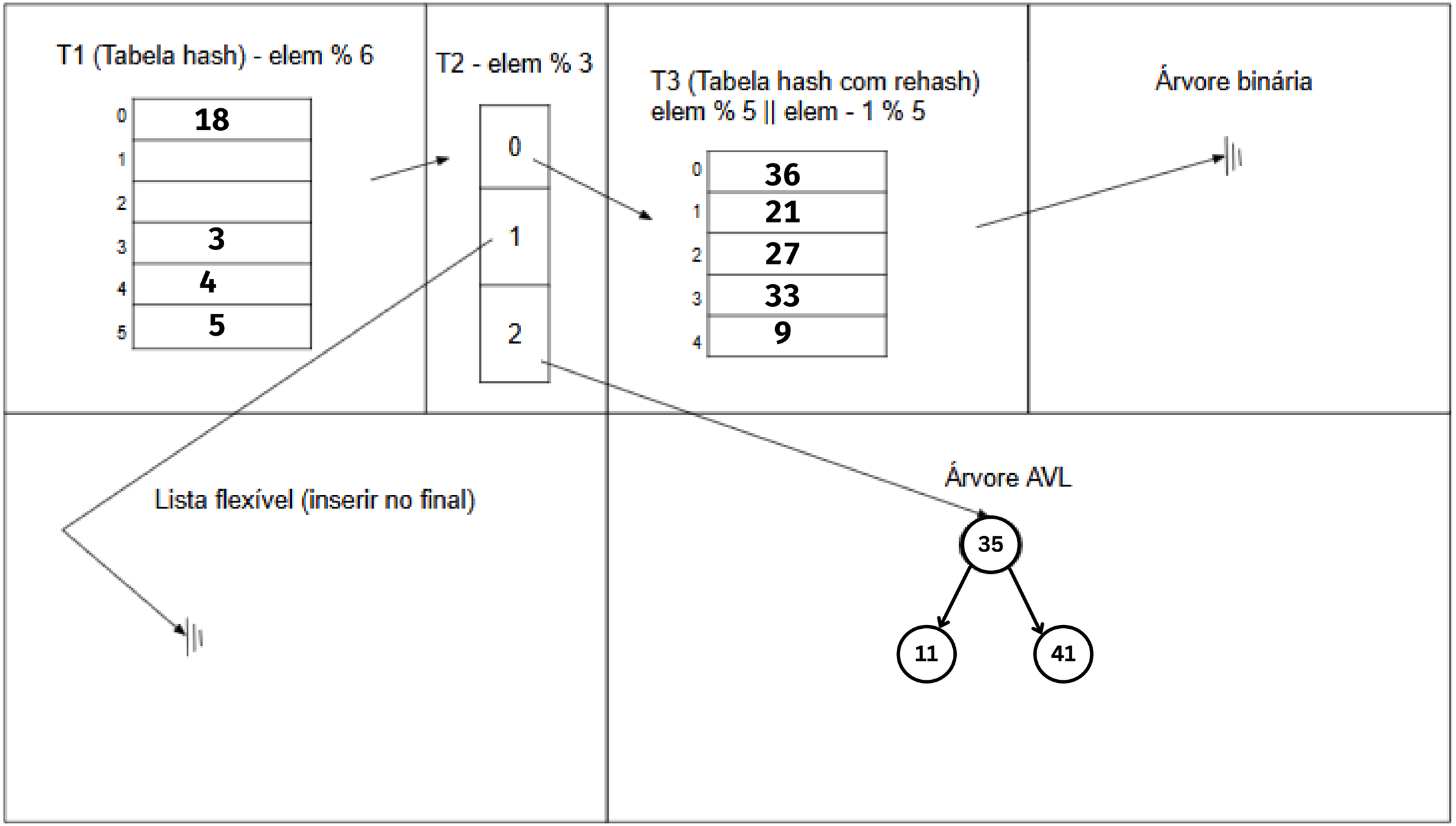
~~5~~, ~~11~~, ~~3~~, ~~9~~, ~~18~~, ~~21~~, ~~33~~, ~~27~~, ~~35~~, ~~41~~, 36, 4, 8, 10, 20, 24, 48, 50, 25, 72, 19



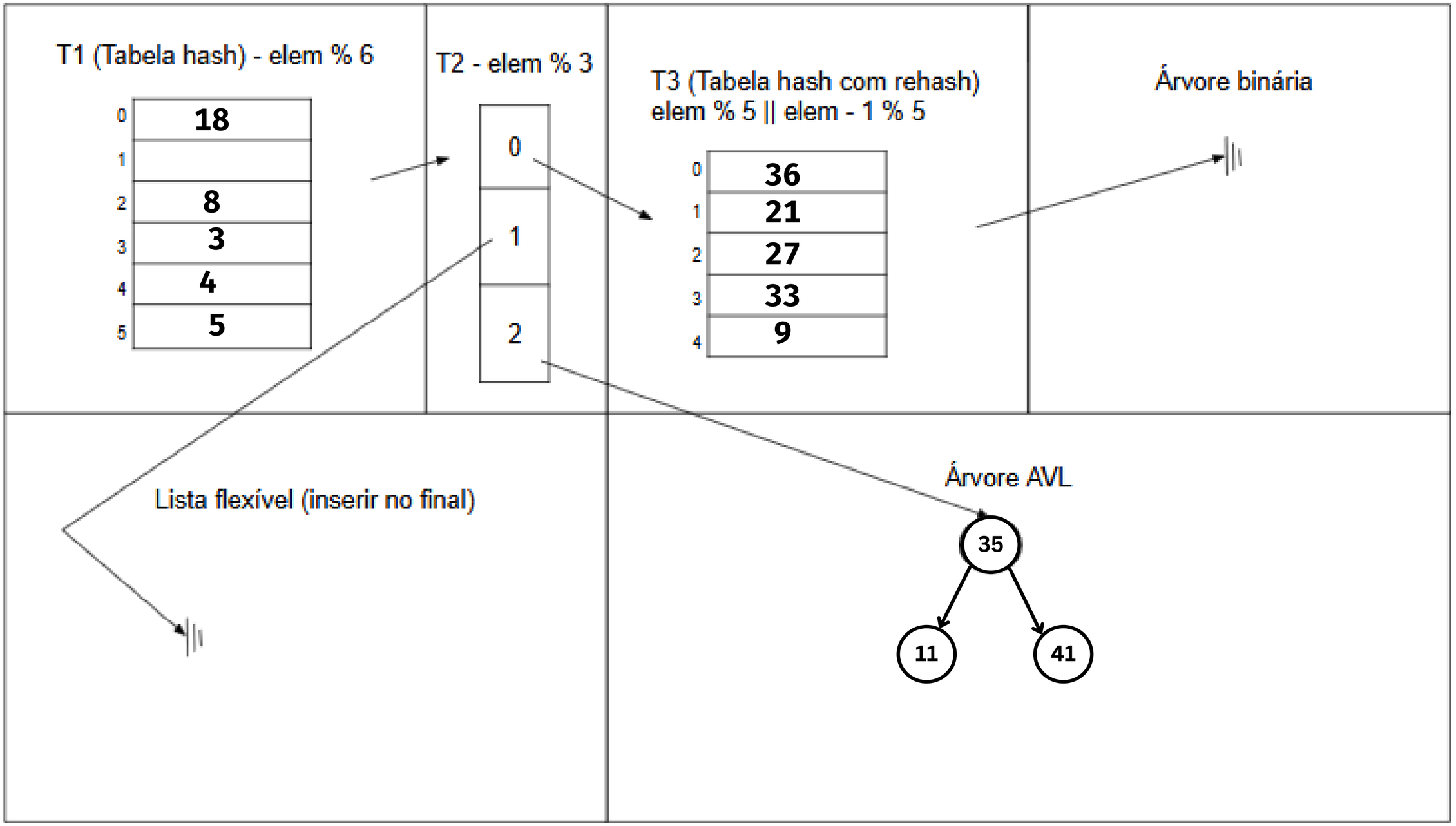
~~5~~, ~~11~~, ~~3~~, ~~9~~, ~~18~~, ~~21~~, ~~33~~, ~~27~~, ~~35~~, ~~41~~, ~~36~~, 4, 8, 10, 20, 24, 48, 50, 25, 72, 19



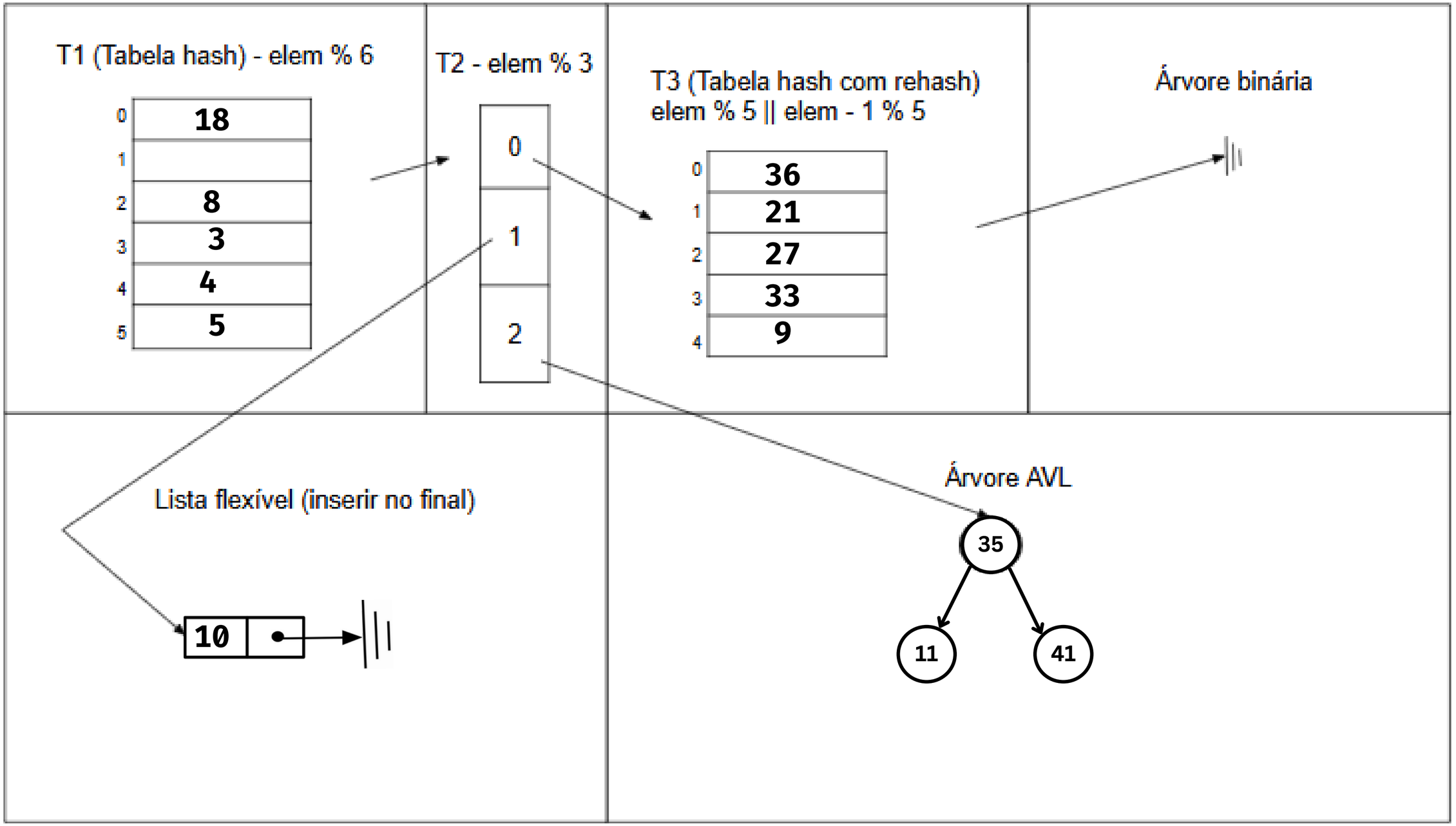
~~3~~, ~~11~~, ~~3~~, ~~9~~, ~~18~~, ~~21~~, ~~33~~, ~~27~~, ~~35~~, ~~41~~, ~~36~~, ~~4~~, 8, 10, 20, 24, 48, 50, 25, 72, 19



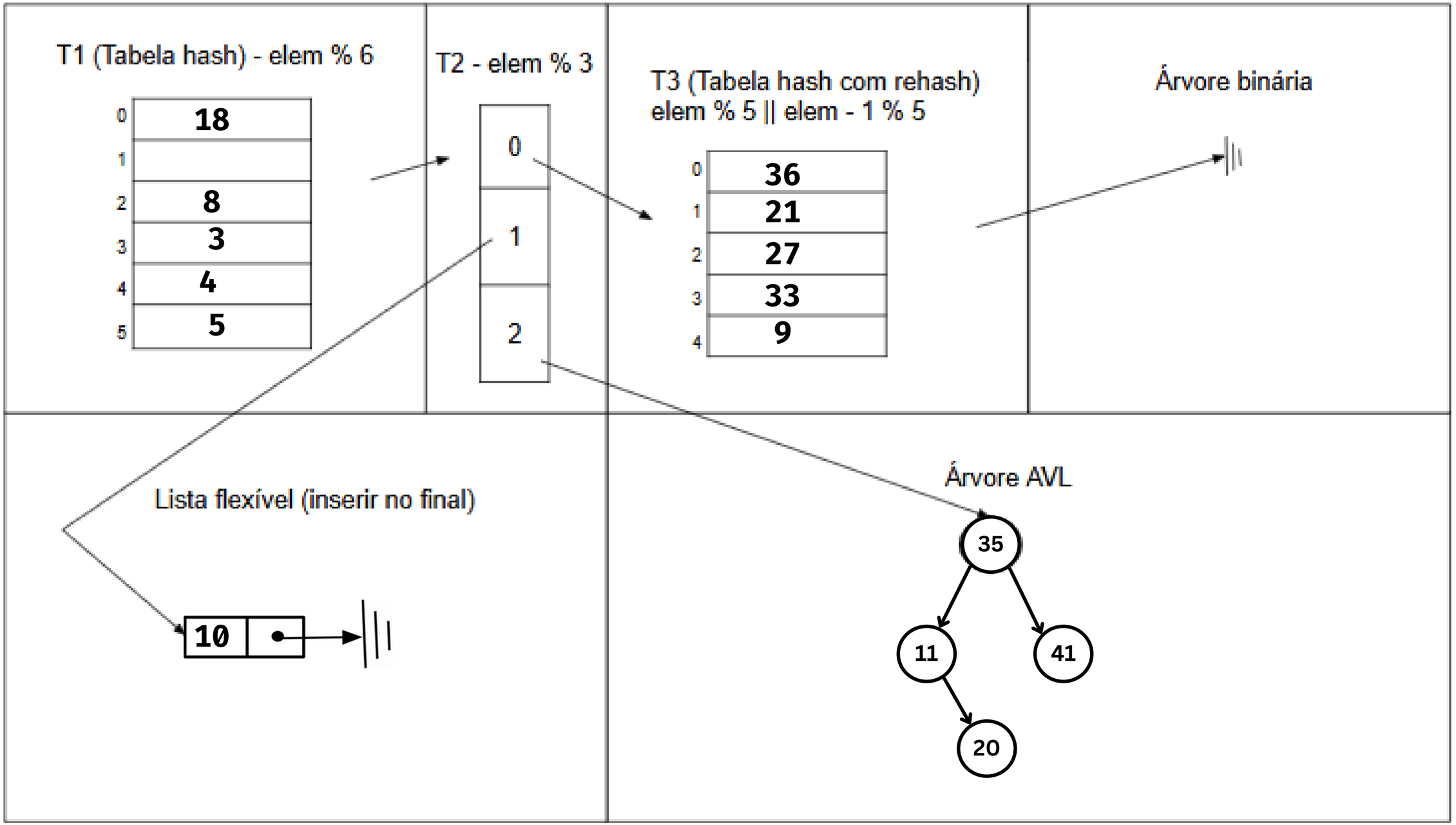
~~5~~, ~~11~~, ~~3~~, ~~9~~, ~~18~~, ~~21~~, ~~33~~, ~~27~~, ~~35~~, ~~41~~, ~~36~~, ~~4~~, ~~8~~, 10, 20, 24, 48, 50, 25, 72, 19



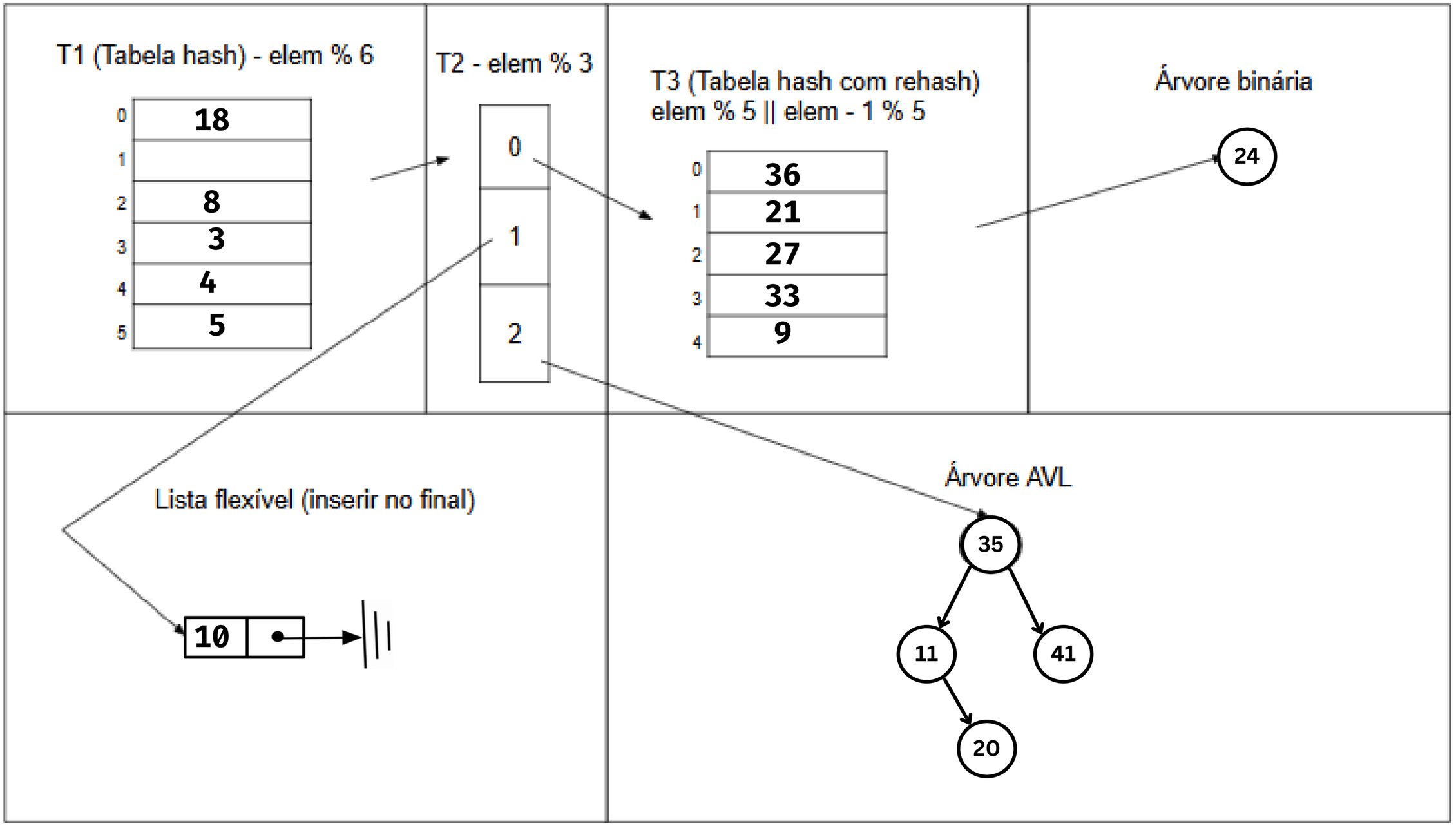
~~5~~, ~~11~~, ~~3~~, ~~9~~, ~~18~~, ~~21~~, ~~33~~, ~~27~~, ~~35~~, ~~41~~, ~~36~~, ~~4~~, ~~8~~, ~~10~~, 20, 24, 48, 50, 25, 72, 19



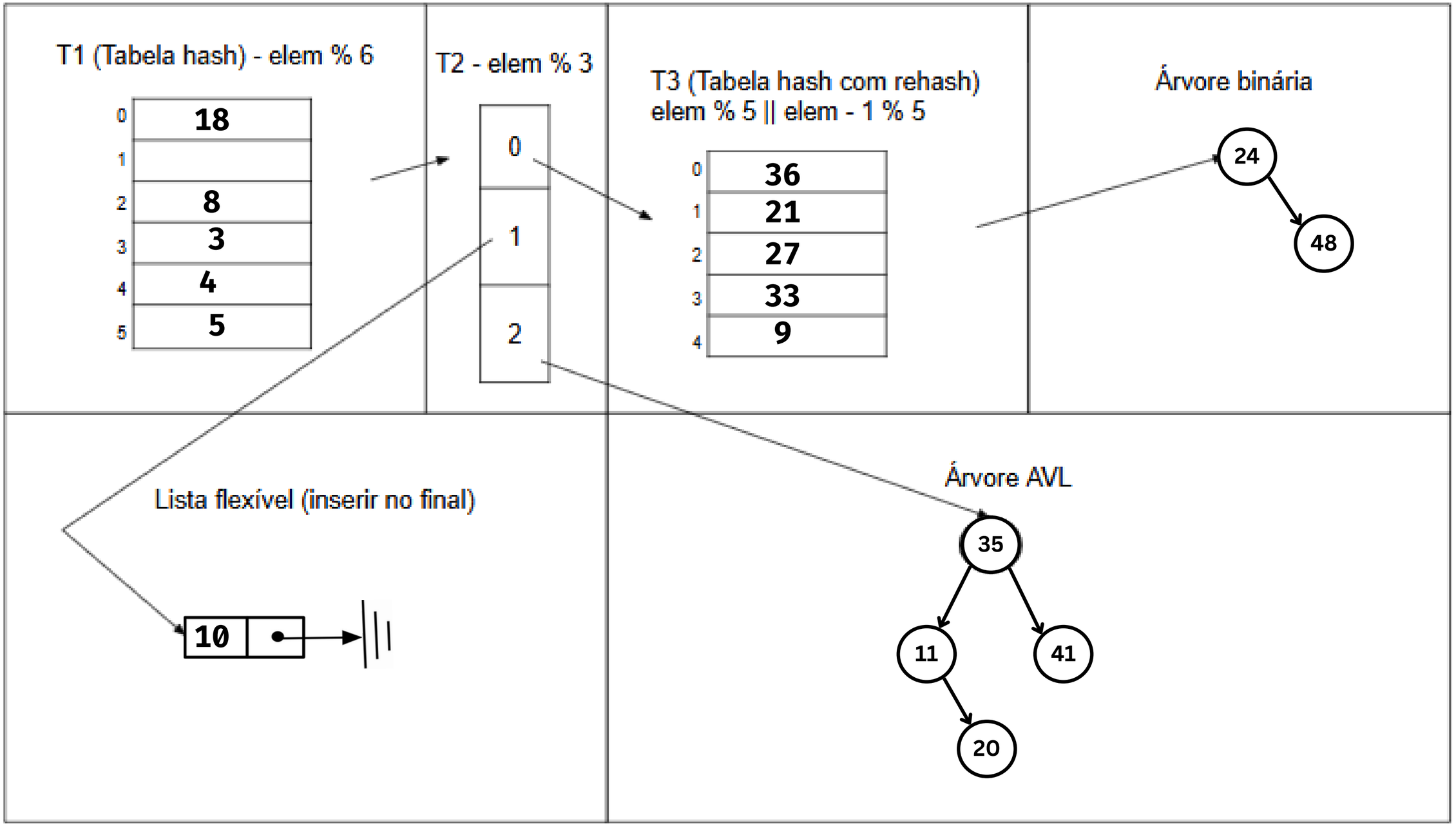
~~5~~, ~~11~~, ~~3~~, ~~9~~, ~~18~~, ~~21~~, ~~33~~, ~~27~~, ~~35~~, ~~41~~, ~~36~~, ~~4~~, ~~8~~, ~~10~~, ~~20~~, 24, 48, 50, 25, 72, 19



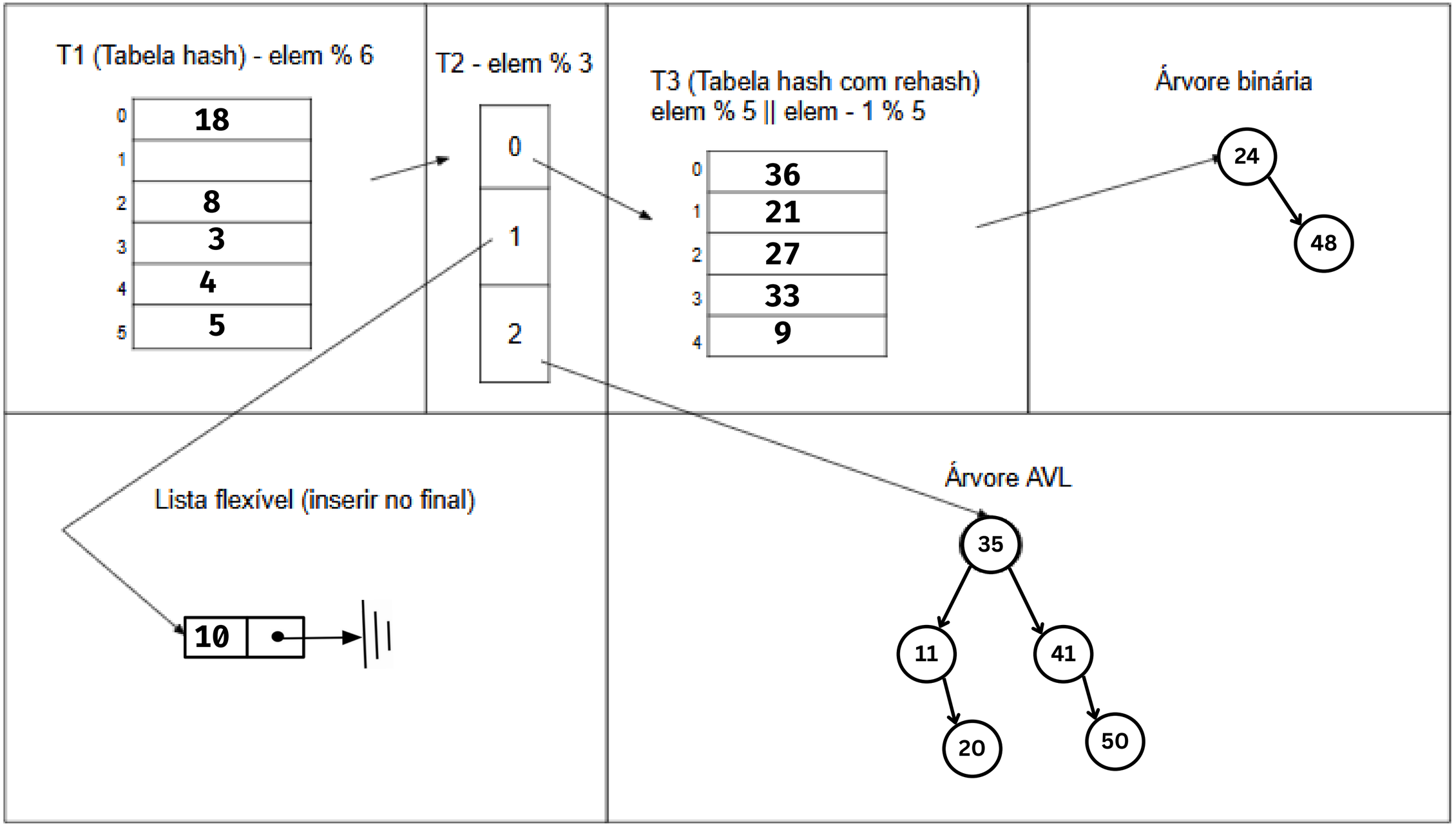
~~5~~, ~~11~~, ~~3~~, ~~9~~, ~~18~~, ~~21~~, ~~33~~, ~~27~~, ~~35~~, ~~41~~, ~~36~~, ~~4~~, ~~8~~, ~~10~~, ~~20~~, ~~24~~, 48, 50, 25, 72, 19



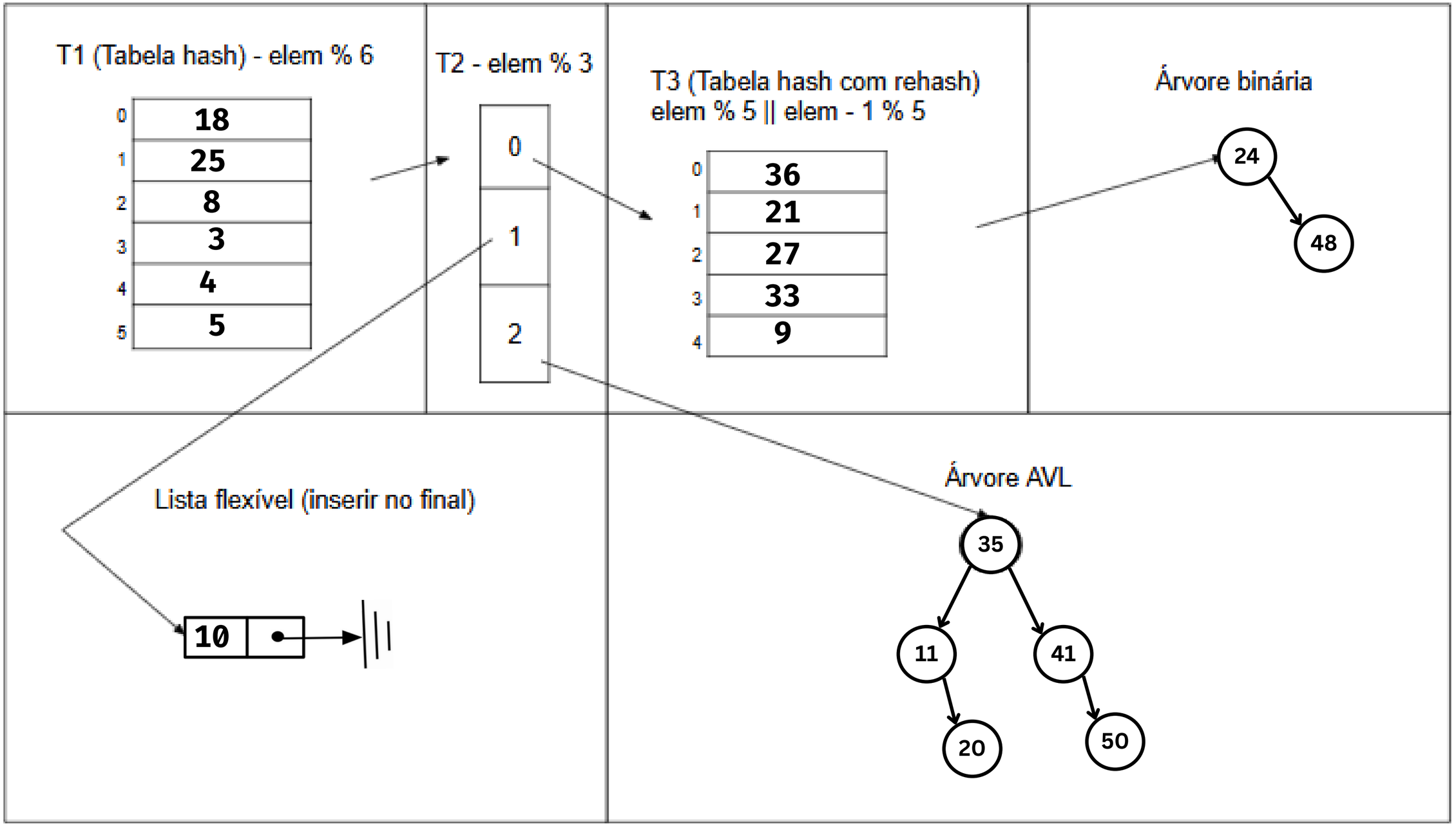
5, 11, 3, 9, 18, 21, 33, 27, 35, 41, 36, 4, 8, 10, 20, 24, 48, 50, 25, 72, 19



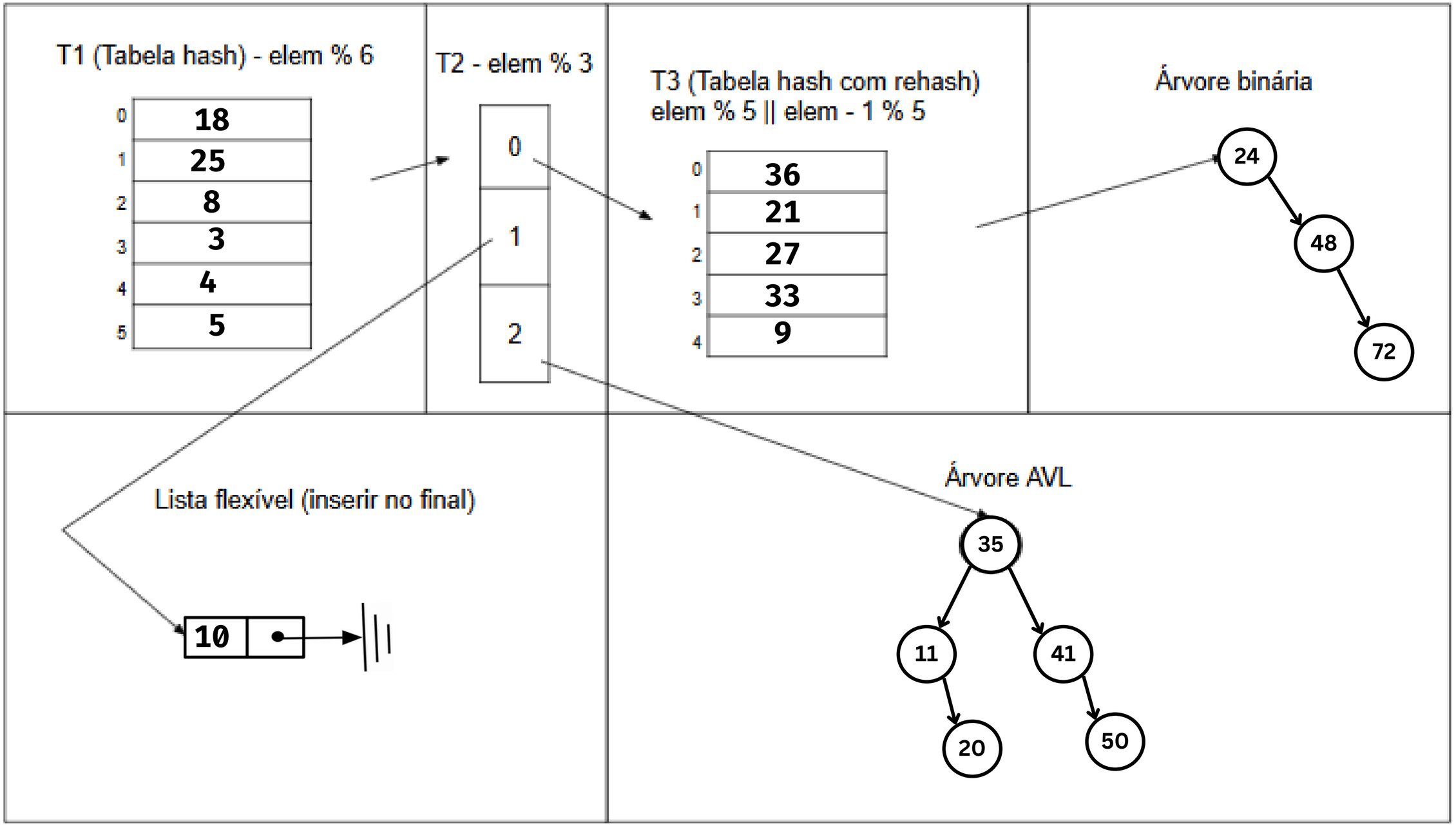
5, 11, 3, 9, 18, 21, 33, 27, 35, 41, 36, 4, 8, 10, 20, 24, 48, 50, 25, 72, 19



~~5~~, ~~11~~, ~~3~~, ~~9~~, ~~18~~, ~~21~~, ~~33~~, ~~27~~, ~~35~~, ~~41~~, ~~36~~, ~~4~~, ~~8~~, ~~10~~, ~~20~~, ~~24~~, ~~48~~, ~~50~~, ~~25~~, 72, 19



5, 11, 3, 9, 18, 21, 33, 27, 35, 41, 36, 4, 8, 10, 20, 24, 48, 50, 25, 72, 19



~~5, 11, 3, 9, 18, 21, 33, 27, 35, 41, 36, 4, 8, 10, 20, 24, 48, 50, 25, 72, 19~~