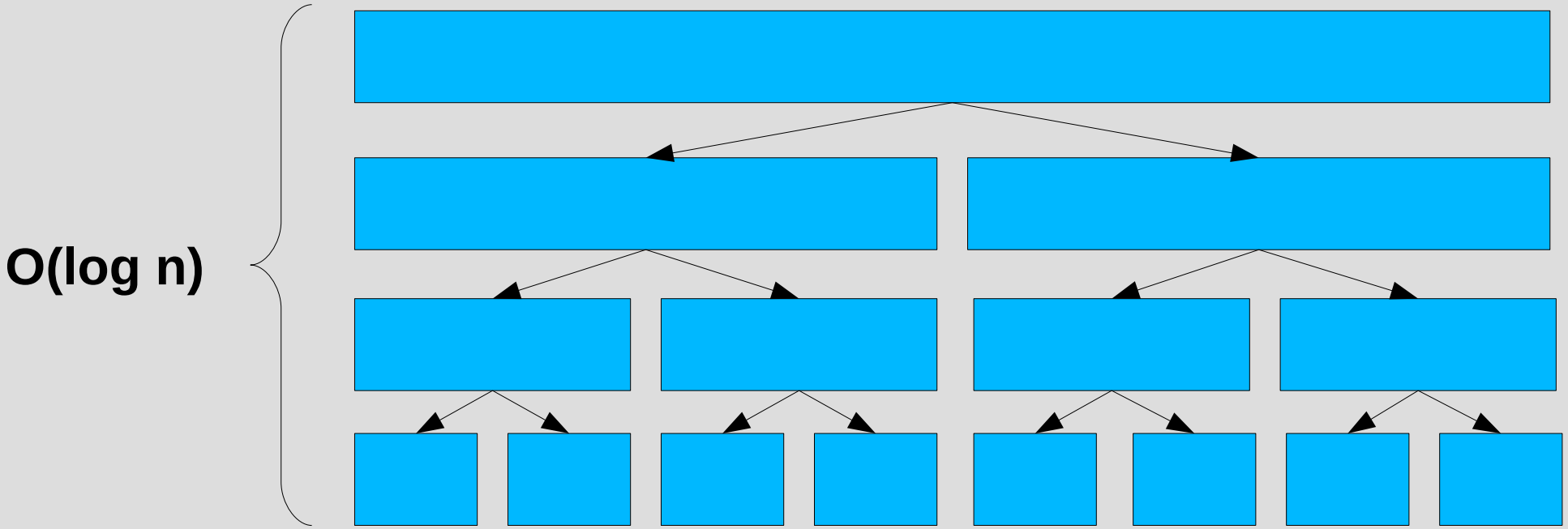
Estruturas de Dados

Mergesort

Mergesort

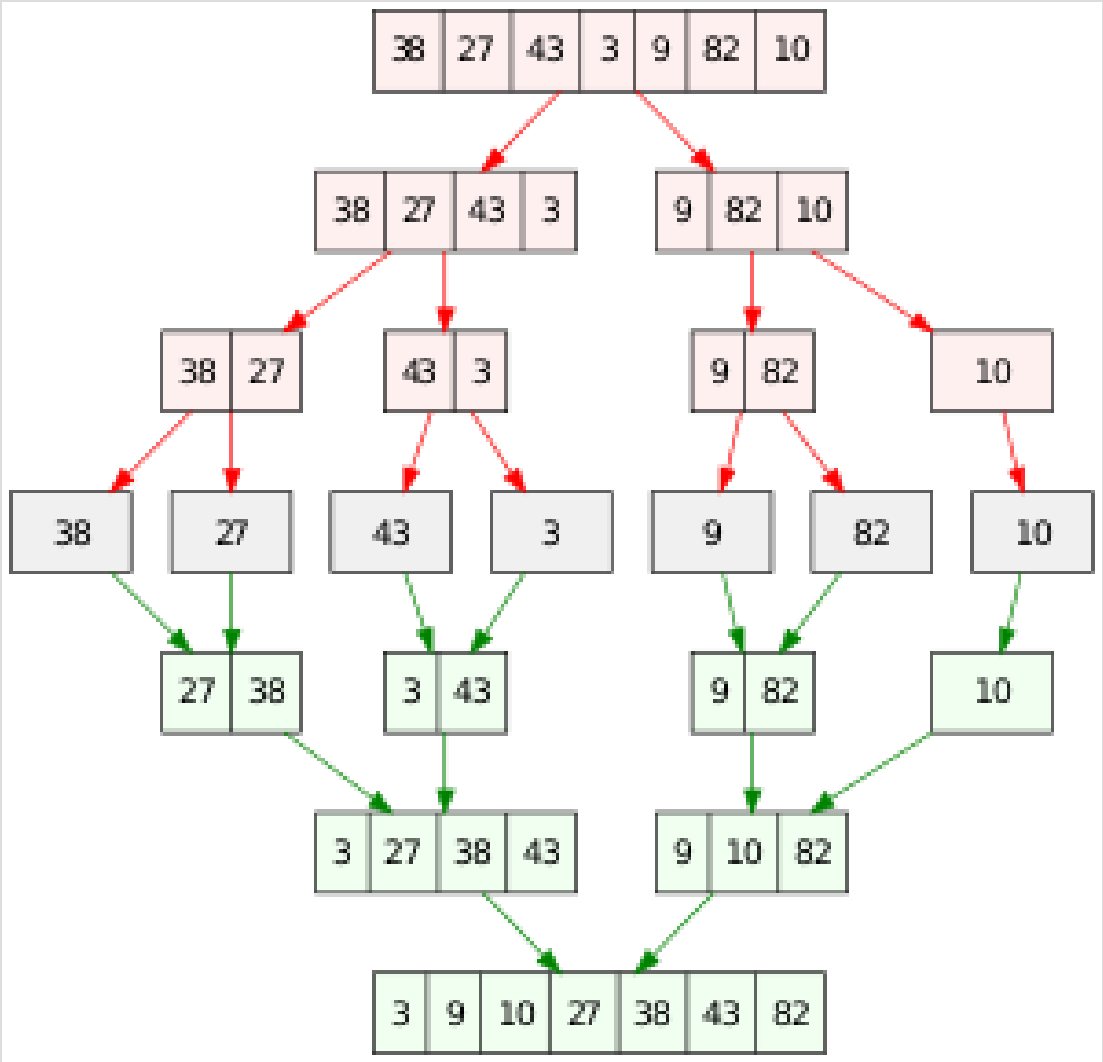
- Algoritmo baseado na estratégia dividir para conquistar, assim como o quicksort
- Complexidade: **O(n log n)**
- Requer memória adicional - **O(n)**

Mergesort



Como o número de operações em cada nível da árvore é **O(n)**, o método Mergesort tem complexidade **O(n log n)**

Mergesort



- Fonte: https://en.wikipedia.org/wiki/Merge_sort

Mergesort

```
void mergeSort(int *v, int n) {
   mergeSort_ordena(v, 0, n-1);
}

void mergeSort_ordena(int *v, int esq, int dir) {
  if (esq == dir)
     return;
  int meio = (esq + dir) / 2;
  mergeSort_ordena(v, esq, meio);
  mergeSort_ordena(v, meio+1, dir);
  mergeSort_intercala(v, esq, meio, dir);
  return;
}
```

- Fonte: slides do Prof. Túlio Toffolo: http://www3.decom.ufop.br/toffolo/site_media/uploads/2013-1/bcc202/slides/14._mergesort.pdf

Mergesort

```
void mergeSort_intercala(int *v, int esq, int meio, int dir){
  int i, j, k;
  int a_tam = meio-esq+1;
  int b_tam = dir-meio;
  int *a = (int*) malloc(sizeof(int) * a_tam);
  int *b = (int*) malloc(sizeof(int) * b_tam);

  for (i = 0; i < a_tam; i++) a[i] = v[i+esq];
  for (i = 0; i < b_tam; i++) b[i] = v[i+meio+1];

  for (i = 0, j = 0, k = esq; k <= dir; k++) {
    if (i == a_tam) v[k] = b[j++];
    else if (j == b_tam) v[k] = a[i++];
    else if (a[i].chave < b[j].chave) v[k] = a[i++];
    else v[k] = b[j++];
  }
  free(a); free(b);
}
```

- Fonte: slides do Prof. Túlio Toffolo: http://www3.decom.ufop.br/toffolo/site_media/uploads/2013-1/bcc202/slides/14._mergesort.pdf