Algoritmos de Ordenação

INSERTION SORT E MERGE SORT

Grupo

EDUARDO JORGE

GUSTAVO JORDAN

JANSEN JUNIOR

VINÍCIUS CAVALCANTE

Roteiro

INSERTION SORT

Como funciona

Características

Implementação

MERGE SORT

Como funciona

Características

Implementação

REFERÊNCIAS BIBLIOGRÁFICAS

Insertion Sort

Ordenação por inserção

Um algoritmo eficiente para ordenar um número pequeno de elementos.

Como funciona ?

- Um item chave é selecionado para ser comparado com os valores que se encontram a sua direita, a comparação ocorre até que apareça um valor maior ou o fim da lista.

- Para realizar a comparação o item chave começa sendo o segundo valor da lista, pois não haveria valores para comparar se ele fosse o primeiro elemento da lista.

Características

ONDE USAR?

Ordenação de listas com poucos elementos ou em listas quase ordenadas.

VANTAGENS

- Simples implementação, leitura e manutenção;
- Tabalha bem com listas pequenas;
- Estável.

DESVANTAGENS

- Alto custo de movimentação de elementos no vetor;
- Baixa eficiência em listas grandes.

7 |10 |6|8|3|5|1|4|2|9

7 |10 |6|8|3|5|1|4|2|9

7|10| |6 |8|3|5|1|4|2|9


7 |10 |6|8|3|5|1|4|2|9
7|10| |6 |8|3|5|1|4|2|9
6|7|10 |8 |3|5|1|4|2|9


7 |10 |6|8|3|5|1|4|2|9
7|10| |6 |8|3|5|1|4|2|9
6|7|10 |8 |3|5|1|4|2|9
6|7|8|10 |3 |5|1|4|2|9

7 |10 |6|8|3|5|1|4|2|9

7|10| |6 |8|3|5|1|4|2|9

6|7|10 |8 |3|5|1|4|2|9

6|7|8|10 |3 |5|1|4|2|9

3|6|7|8|10| 5 | 1|4|2|9

7 |10 |6|8|3|5|1|4|2|9

7|10| |6 |8|3|5|1|4|2|9

6|7|10 |8 |3|5|1|4|2|9

6|7|8|10 |3 |5|1|4|2|9

3|6|7|8|10| 5 | 1|4|2|9

3|4|5|6|7|8|10|1|4|2|9

7 |10 |6|8|3|5|1|4|2|9

7|10| |6 |8|3|5|1|4|2|9

6|7|10 |8 |3|5|1|4|2|9

6|7|8|10 |3 |5|1|4|2|9

3|6|7|8|10| 5 | 1|4|2|9

3|4|5|6|7|8|10|1|4|2|9

1|3|4|5|6|7|8|10|4 |2|9

7 |10 |6|8|3|5|1|4|2|9

7|10| |6 |8|3|5|1|4|2|9

6|7|10 |8 |3|5|1|4|2|9

6|7|8|10 |3 |5|1|4|2|9

3|6|7|8|10 | 5 | 1|4|2|9

3|4|5|6|7|8|10 |1| 4|2|9

1|3|4|5|6|7|8|10 |4 |2|9

1|3|4|5|6|7|8|10 |2 9

7 |10 |6|8|3|5|1|4|2|9

7|10| |6 |8|3|5|1|4|2|9

6|7|10 |8 |3|5|1|4|2|9

6|7|8|10 |3 |5|1|4|2|9

3|6|7|8|10 | 5 | 1|4|2|9

3|4|5|6|7|8|10 |1| 4|2|9

1|3|4|5|6|7|8|10 |4 |2|9

1|3|4|5|6|7|8|10 |2 9

1|2|3|4|5|6|7|8|10 |9

1|2|3|4|5|6|7|8|9|10

```
void insertionSort(int vet[],int n) {
    int i, chave, j;
    for (i = 1; i < n; i++)
    {
        chave = vet[i];
        j = i - 1;

        while (j >= 0 && vet[j] > chave)
        {
            vet[j + 1] = vet[j];
            j = j - 1;
        }
        vet[j + 1] = chave;
    }
}
```

Implementação

C++

```python
def insertionSort(vet):

    if len(vet) <= 1: return vet

    for i in range(1, len(vet)):
        posAtual = i
        posAnterior = i - 1

        while posAnterior >= 0 and vet[posAtual] < vet[posAnterior]:
            vet[posAtual] += vet[posAnterior]
            vet[posAnterior] = vet[posAtual] - vet[posAnterior]
            vet[posAtual] = vet[posAtual] - vet[posAnterior]
            posAtual -= 1
            posAnterior -= 1
    return vet
```

Implementação

Python

Merge Sort

Dividir para Conquistar

Divide uma lista de números em listas menores, ordenando-as, e mesclando-as.

Como Funciona?

DIVISÃO

Dado uma lista (vetor) de números, divide-se ao meio de forma recursiva, até que cada lista tenha apenas um valor.

COMPARAÇÃO

Compara cada valor de cada lista para definir qual deles é menor.

COMBINAÇÃO

Os valores são copiados para um vetor auxiliar, de forma a ficarem ordenados.

Características

ONDE USAR?

Ordenação de listas, preferencialmente encadeadas.

VANTAGENS

- Fácil implementação e entendimento;
- Tabalha bem com restrição de tempo - O(n log n);
- Estável;

DESVANTAGENS

- Memória auxiliar - O(n)
- Alto consumo de memória, devido à recursão;

7 10 6 8 3 5 1 4 2 9

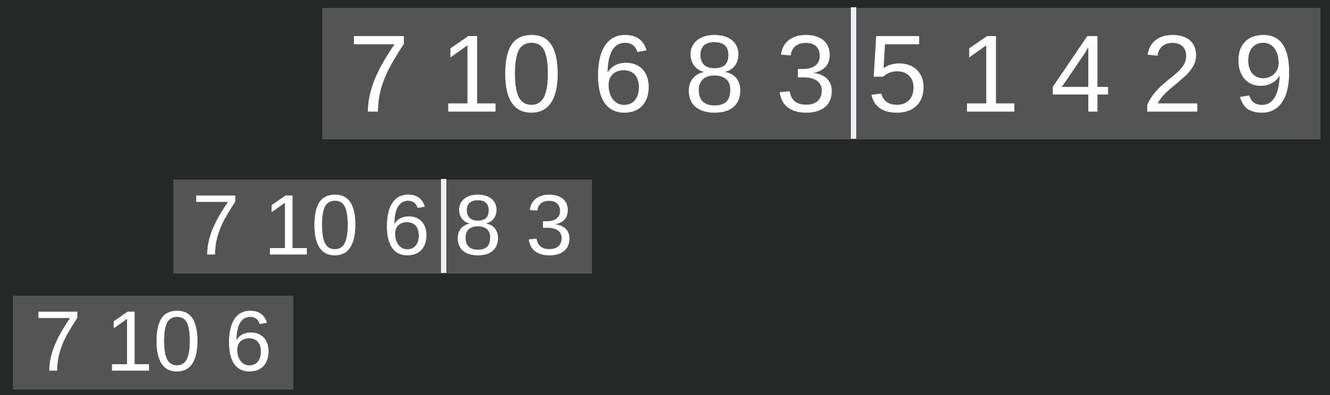
7 10 6 8 3 | 5 1 4 2 9

7 10 6 8 3 | 5 1 4 2 9

7 10 6 8 3

7 10 6 8 3 | 5 1 4 2 9

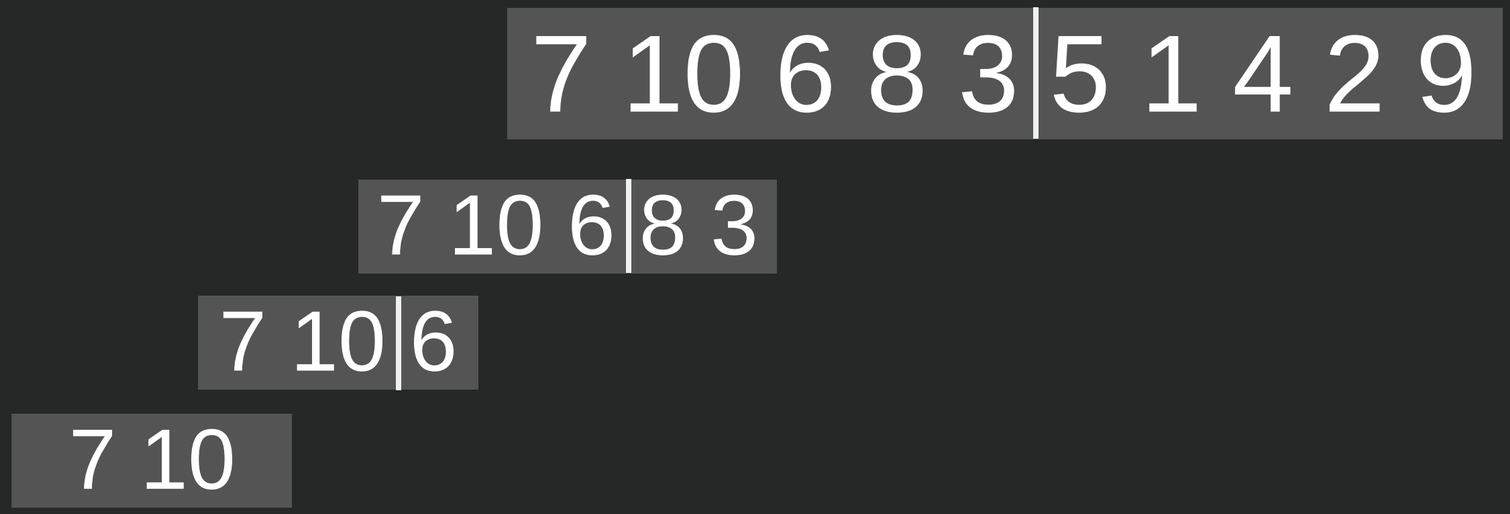
7 10 6 | 8 3

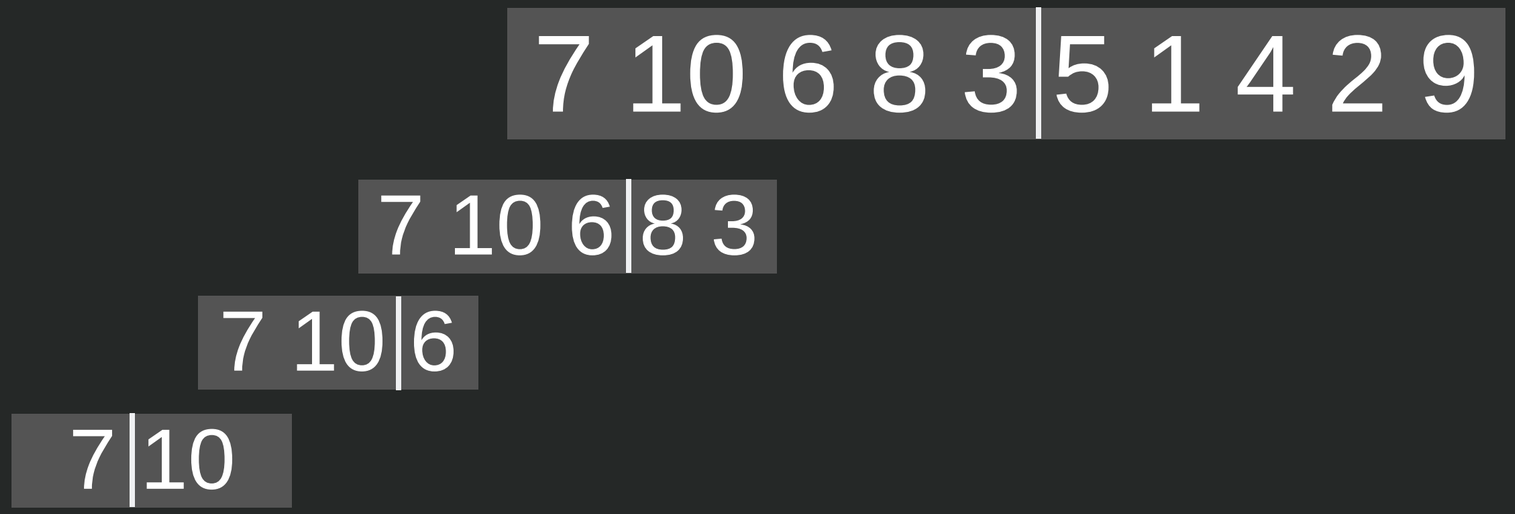

7 10 6 8 3 5 1 4 2 9
7 10 6 8 3
7 10 6

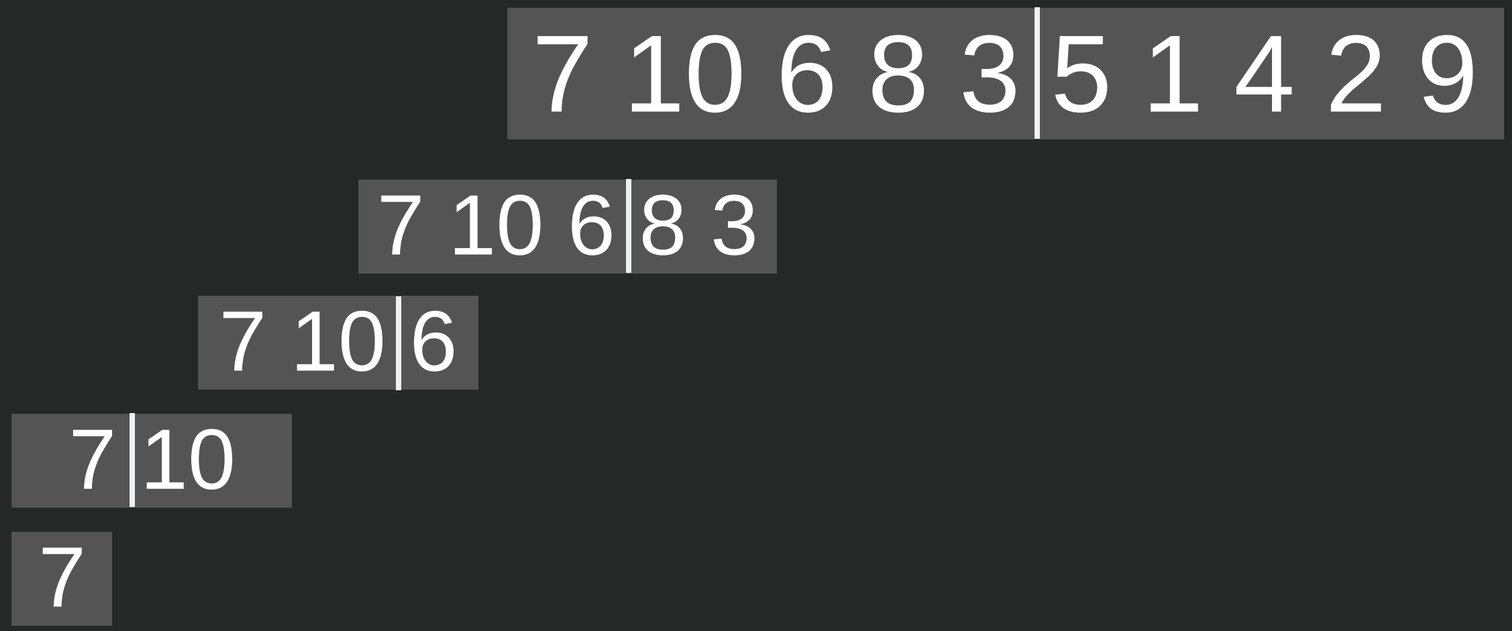
7 10 6 8 3 | 5 1 4 2 9

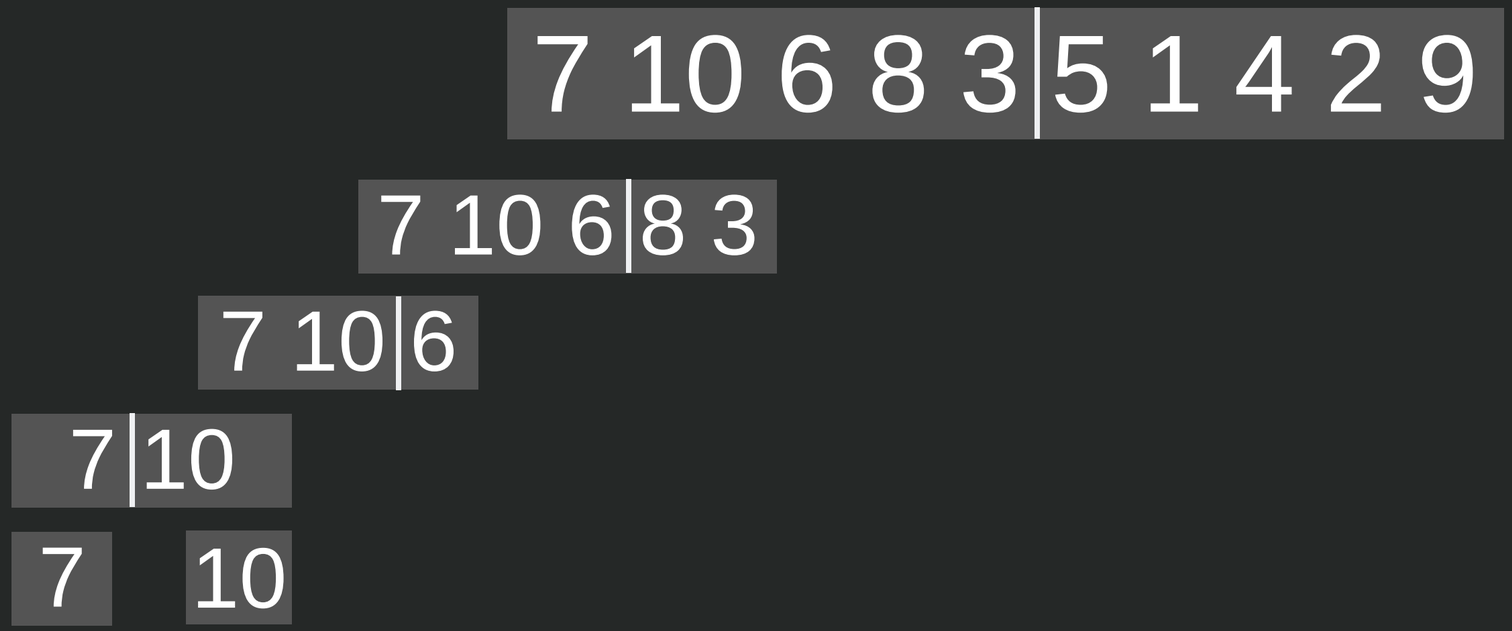
7 10 6 | 8 3

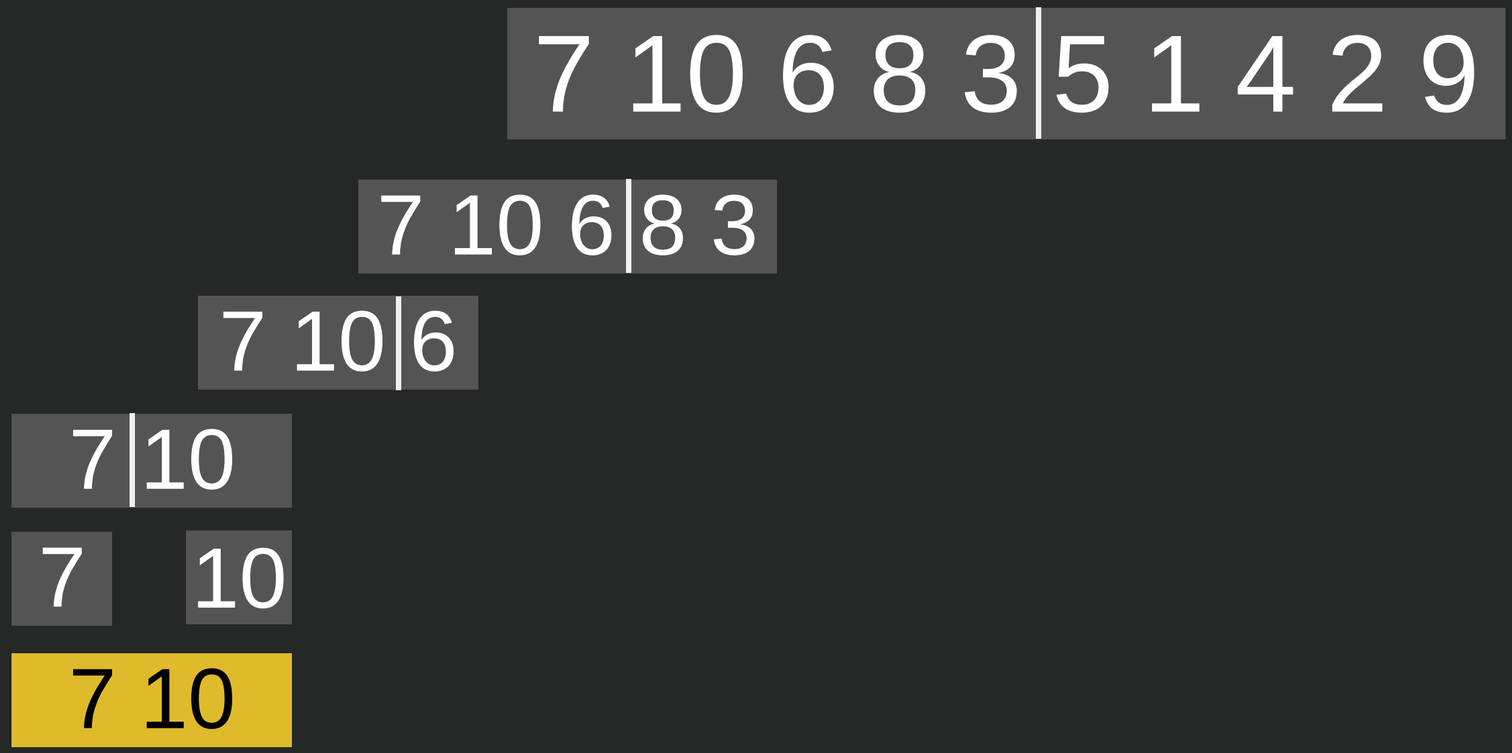
7 10 | 6

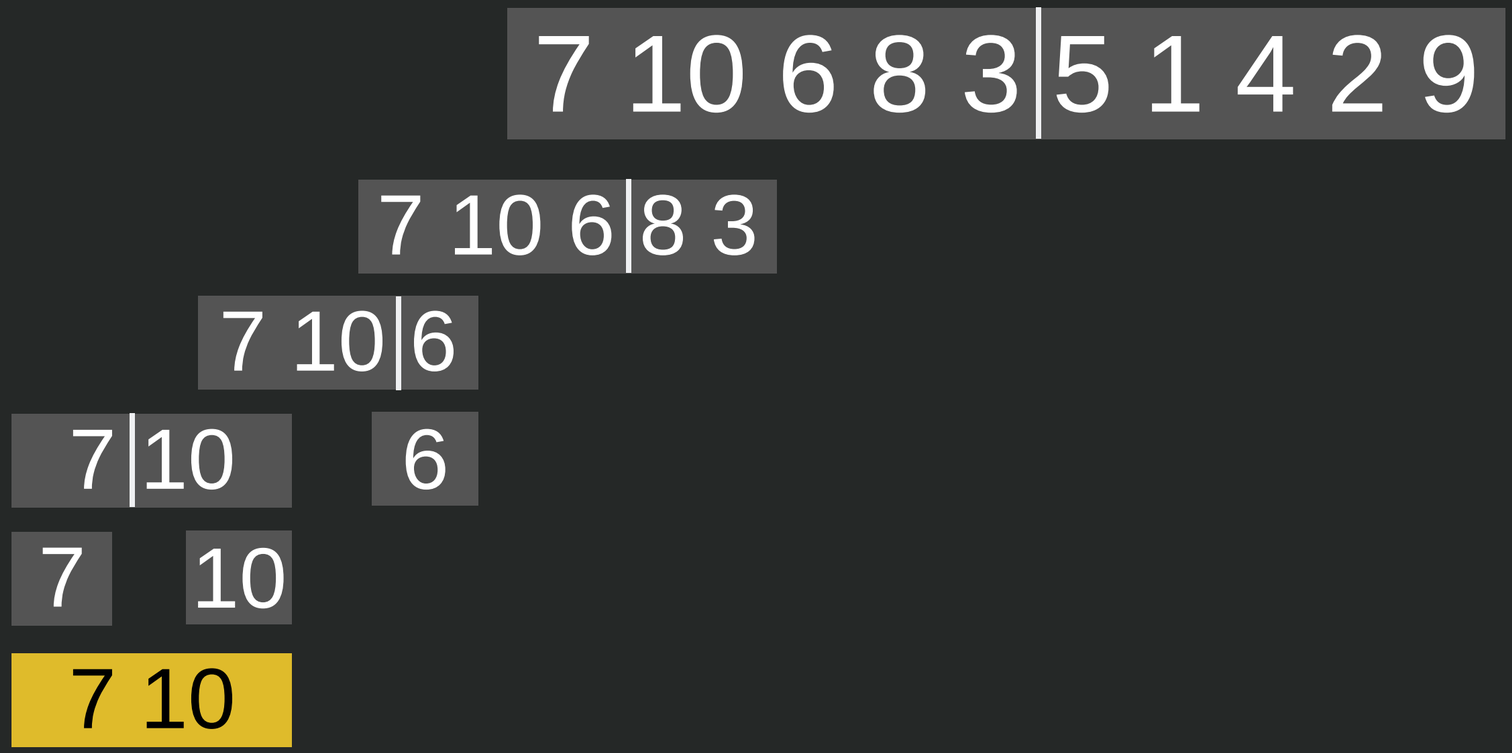

7 10 6 8 3 5 1 4 2 9
7 10 6 8 3
7 10 6
7 10

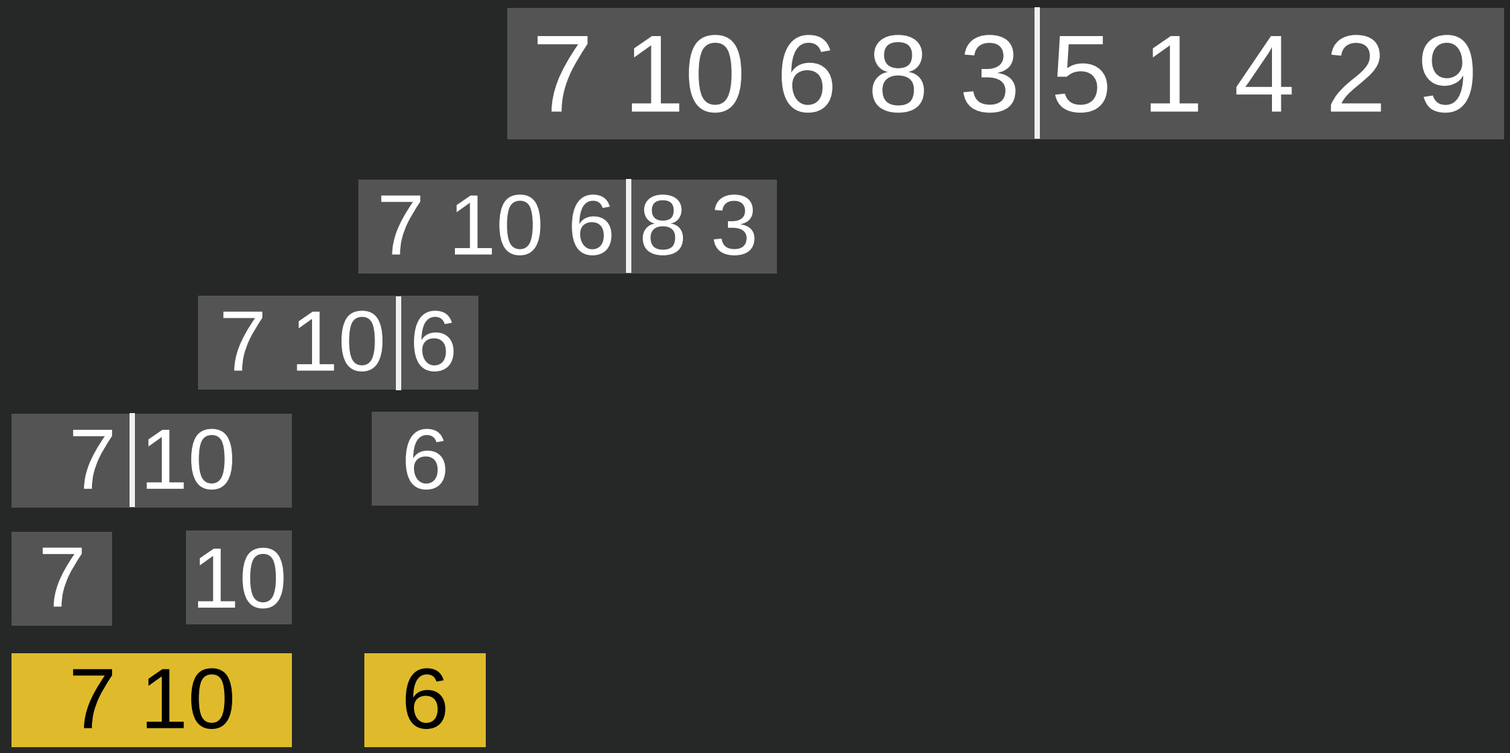

7 10 6 8 3 5 1 4 2 9
7 10 6 8 3
7 10 6
7 10

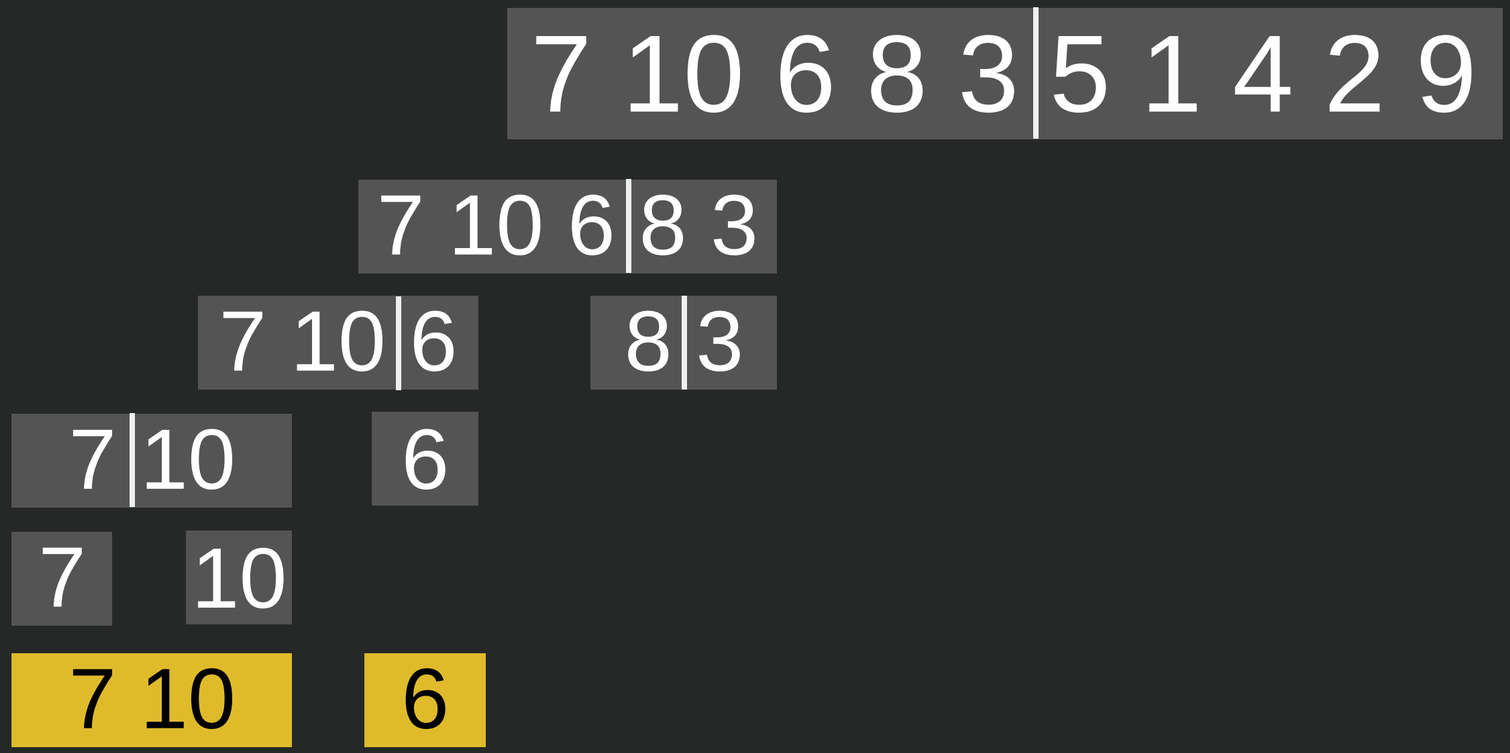

7 10 6 8 3 5 1 4 2 9
7 10 6 8 3
7 10 6
7 10
7

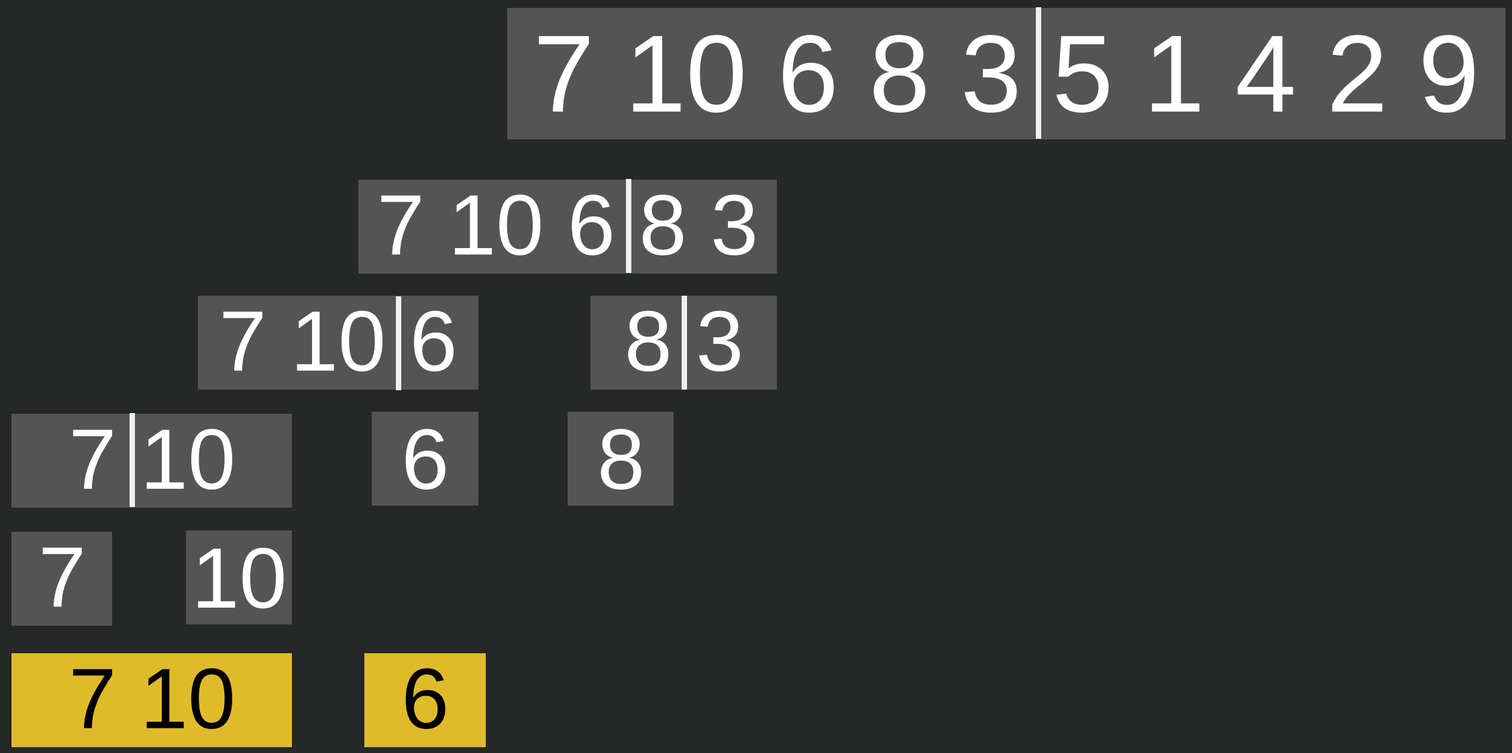

7 10 6 8 3 5 1 4 2 9
7 10 6 8 3
7 10 6
7 10
7
10

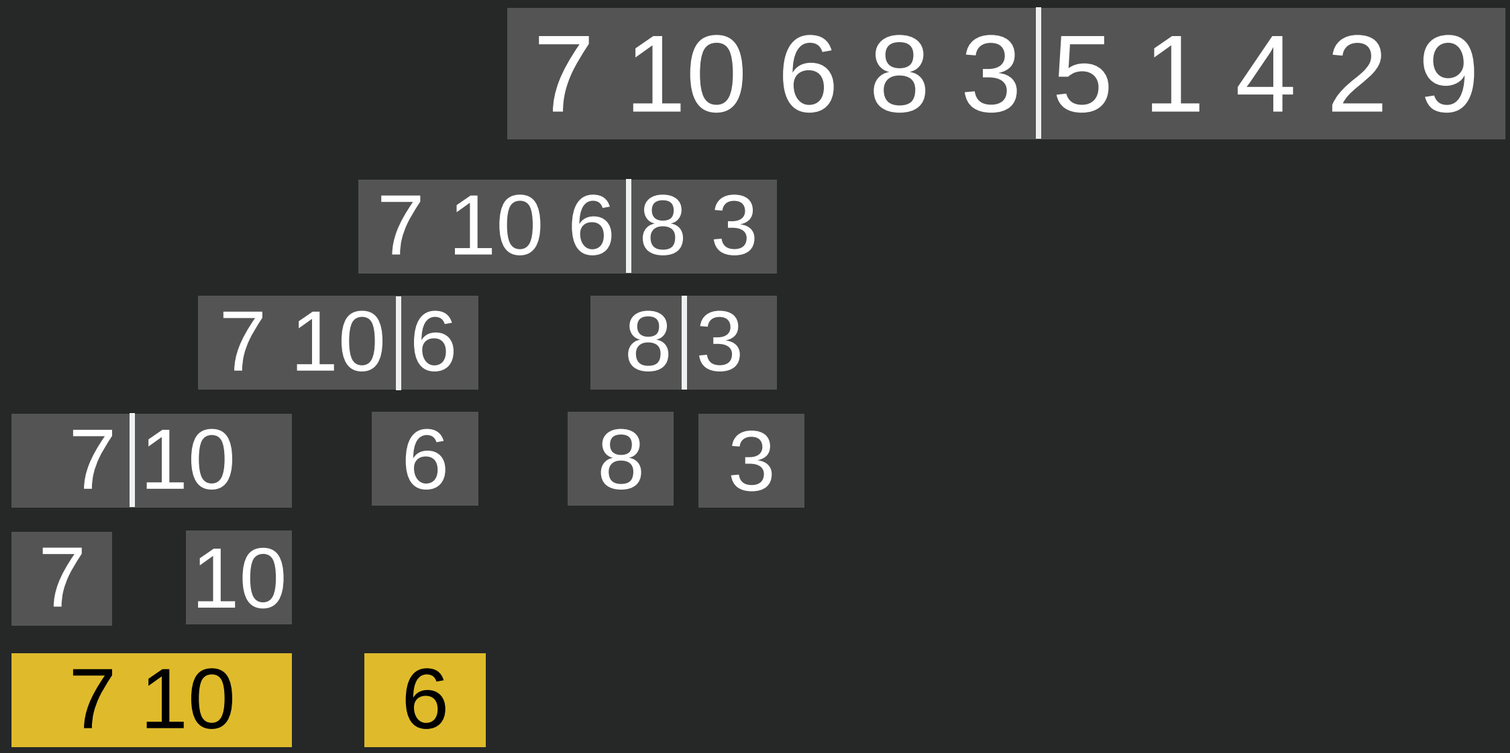

7 10 6 8 3 5 1 4 2 9
7 10 6 8 3
7 10 6
7 10
7
10
7 10

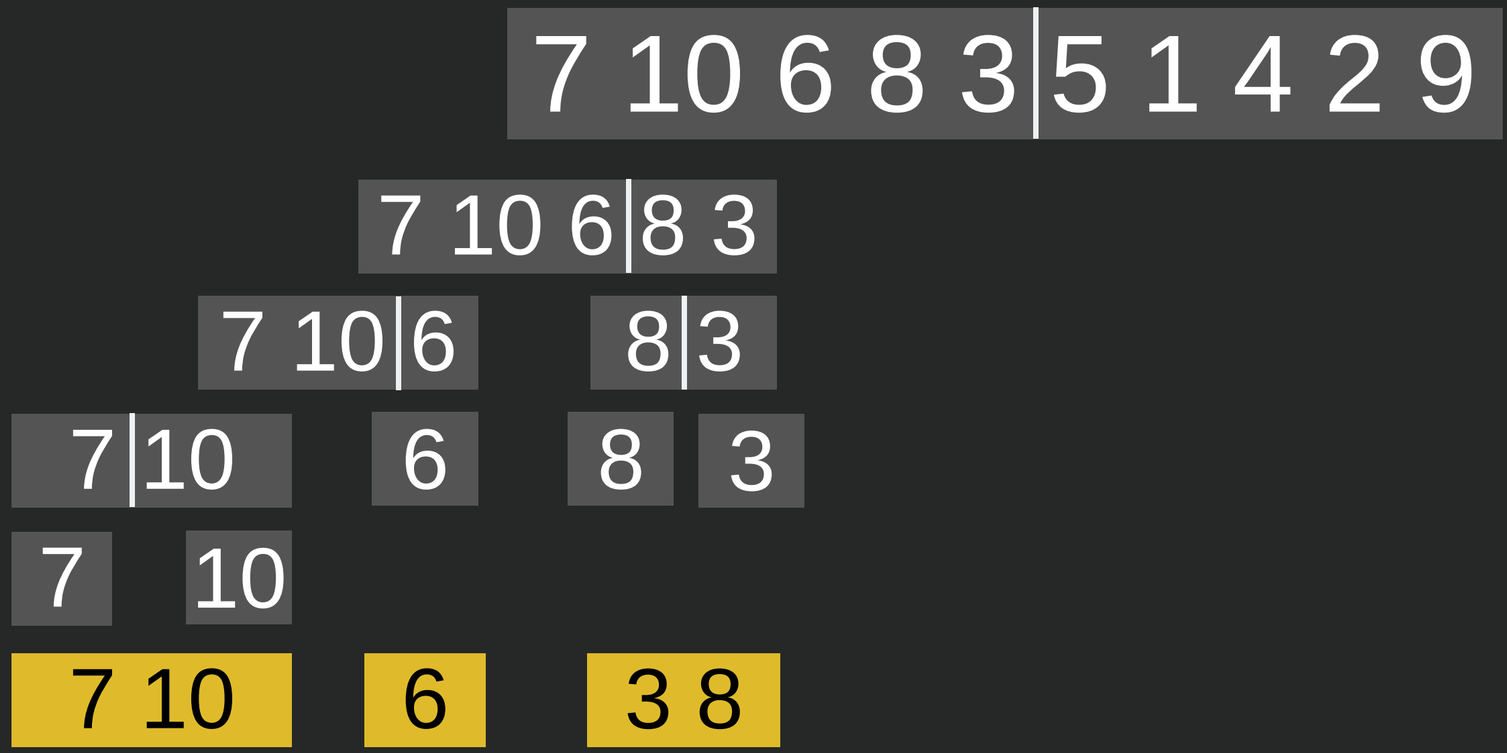

7 10 6 8 3 5 1 4 2 9
7 10 6 8 3
7 10 6
7 10
7
10
7 10

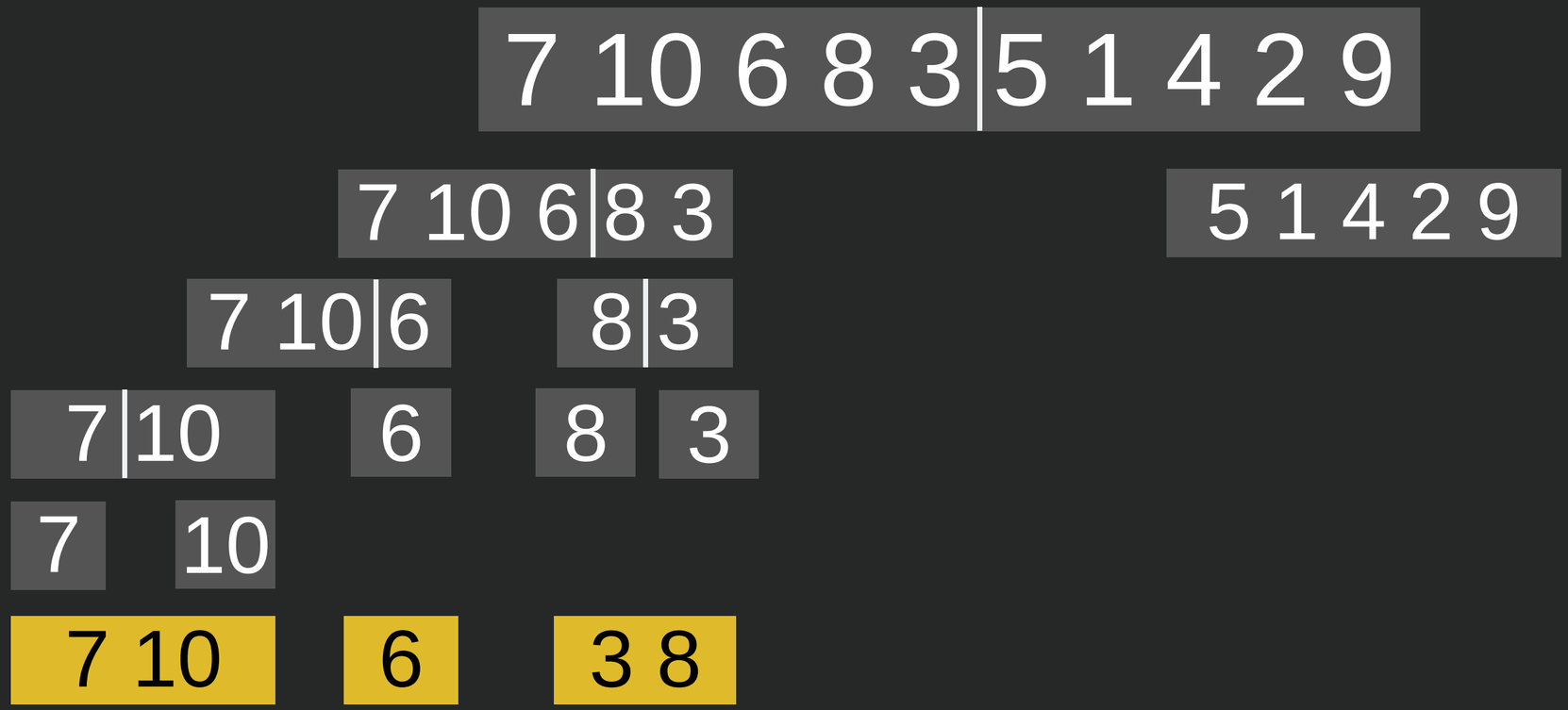

7 10 6 8 3 5 1 4 2 9
7 10 6 8 3
7 10 6
7 10
6
7
10
7 10
6

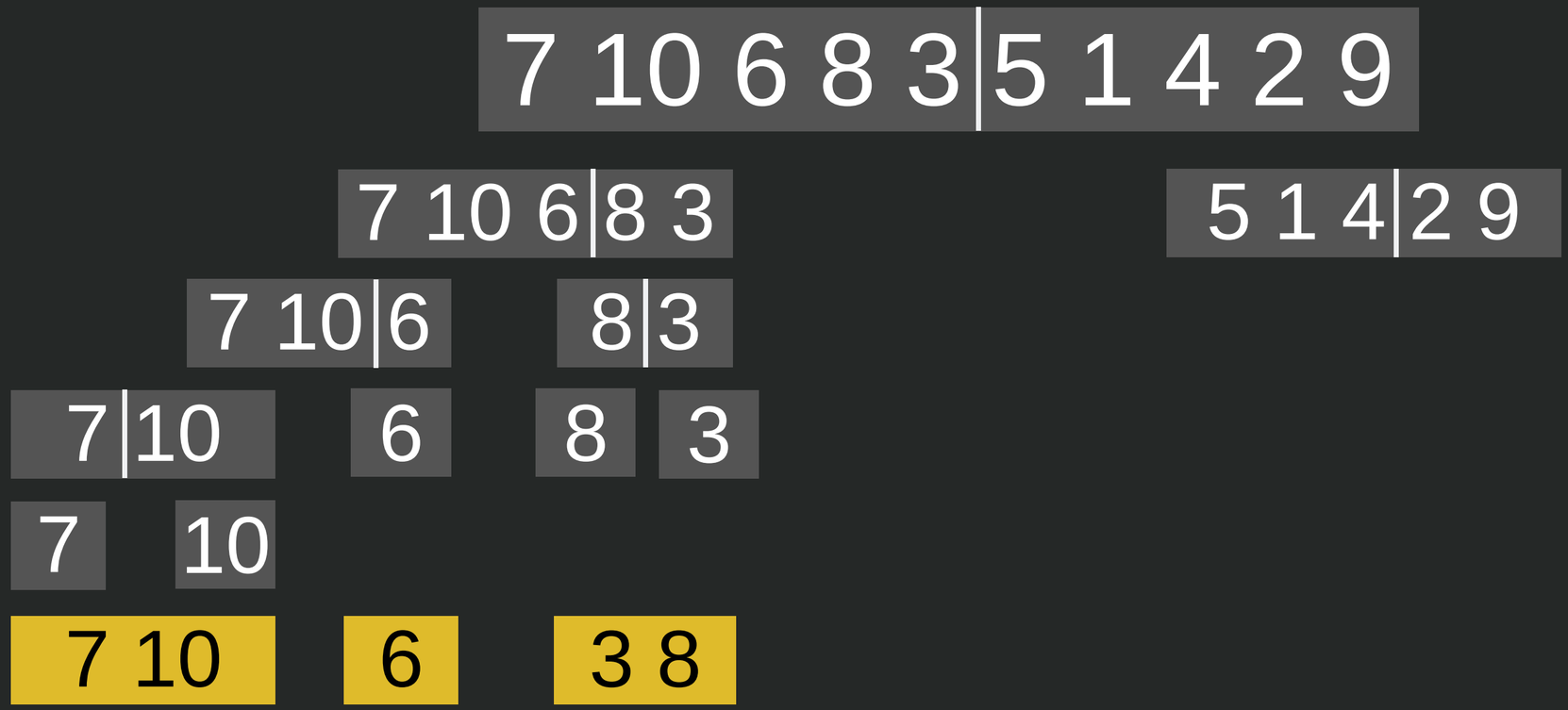

7 10 6 8 3 5 1 4 2 9
7 10 6 8 3
7 10 6 8 3
7 10 6
8 3
7 10
6
7
10
7 10
6

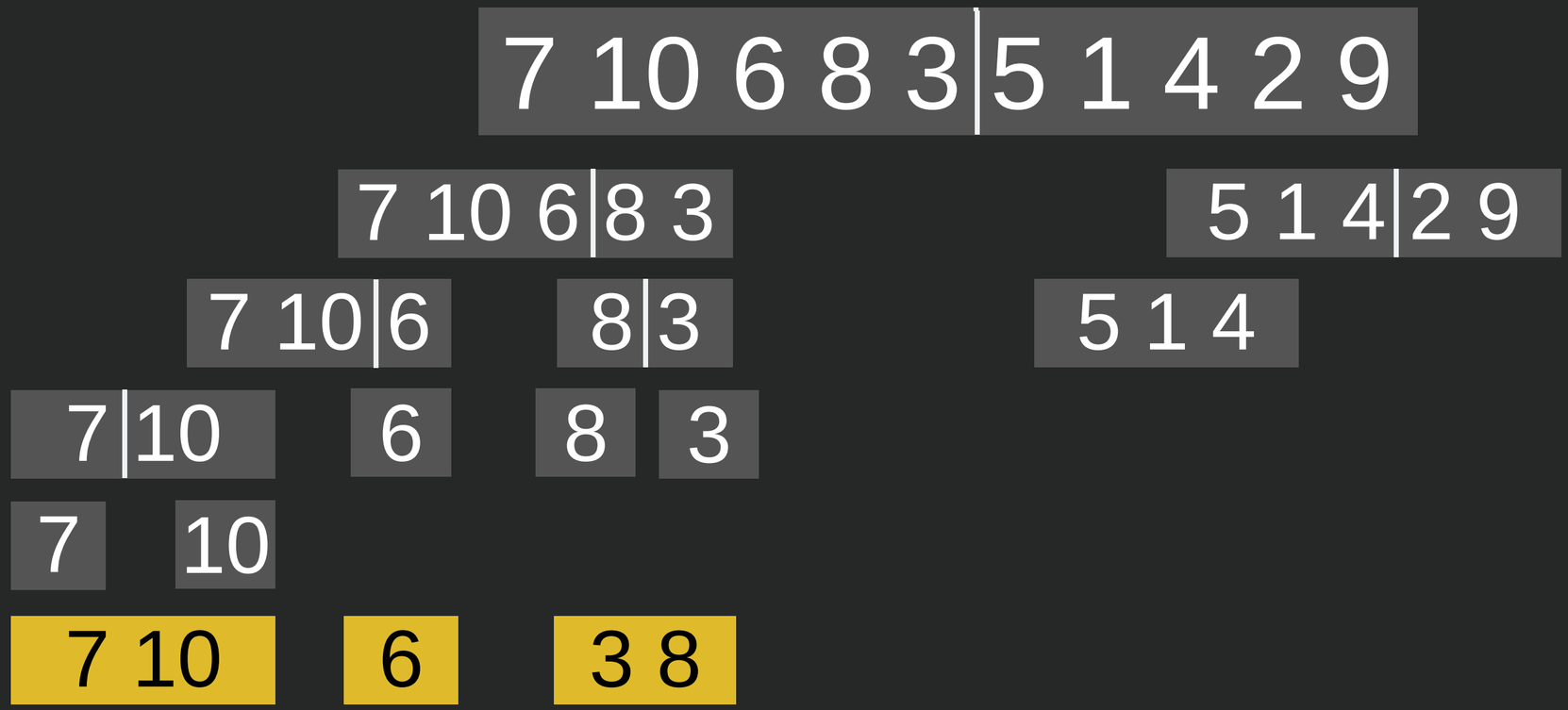

7 10 6 8 3 5 1 4 2 9
7 10 6 8 3
7 10 6 8 3
7 10 6 8
7 10
7
10
7 10
6

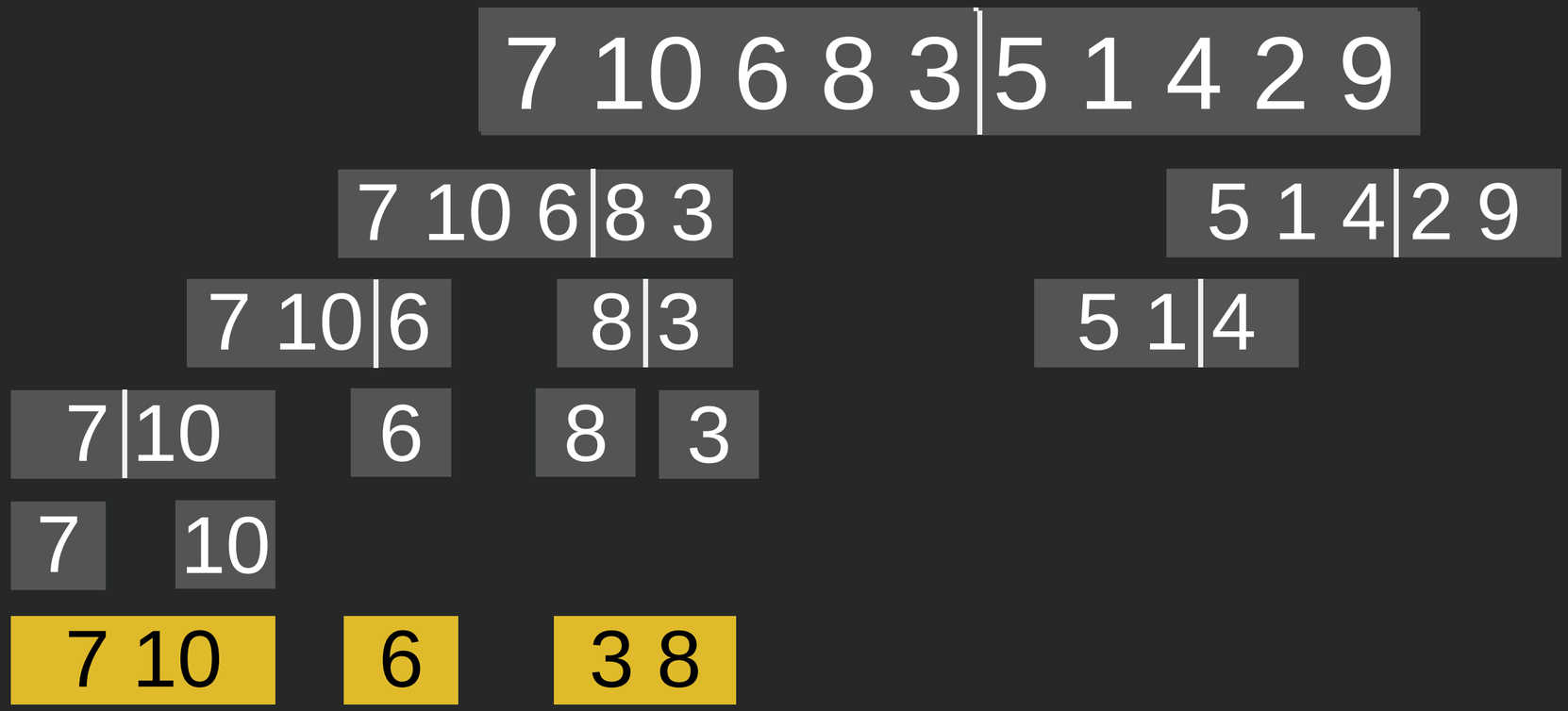

7 10 6 8 3 5 1 4 2 9
7 10 6 8 3
7 10 6
8 3
7 10
6
8
3
7
10
7 10
6

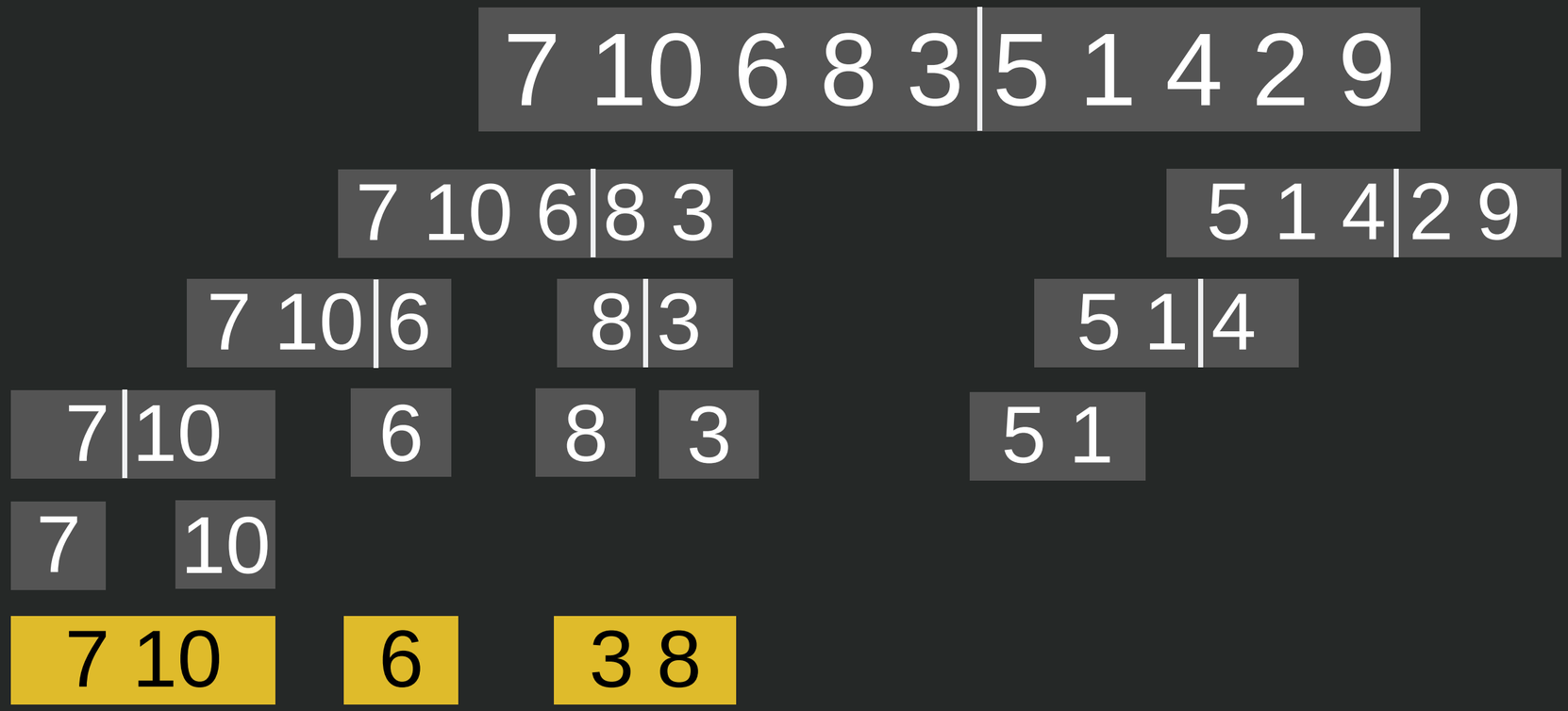

7 10 6 8 3 5 1 4 2 9
7 10 6 8 3
7 10 6
8 3
7 10
6
8
3
7
10
7 10
6
3 8

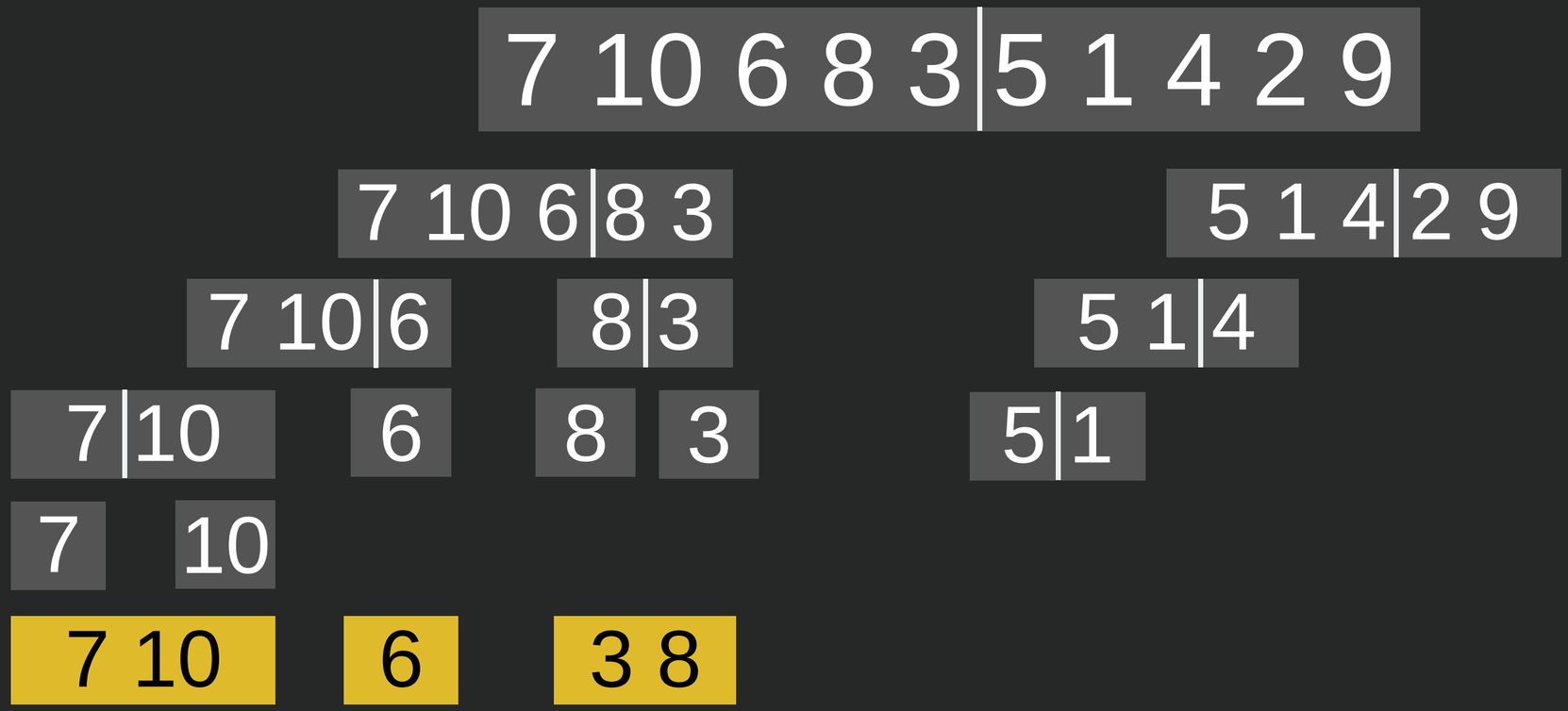

7 10 6 8 3 5 1 4 2 9
7 10 6 8 3
5 1 4 2 9
7 10 6
8 3
7 10
6
8
3
7
10
7 10
6
3 8

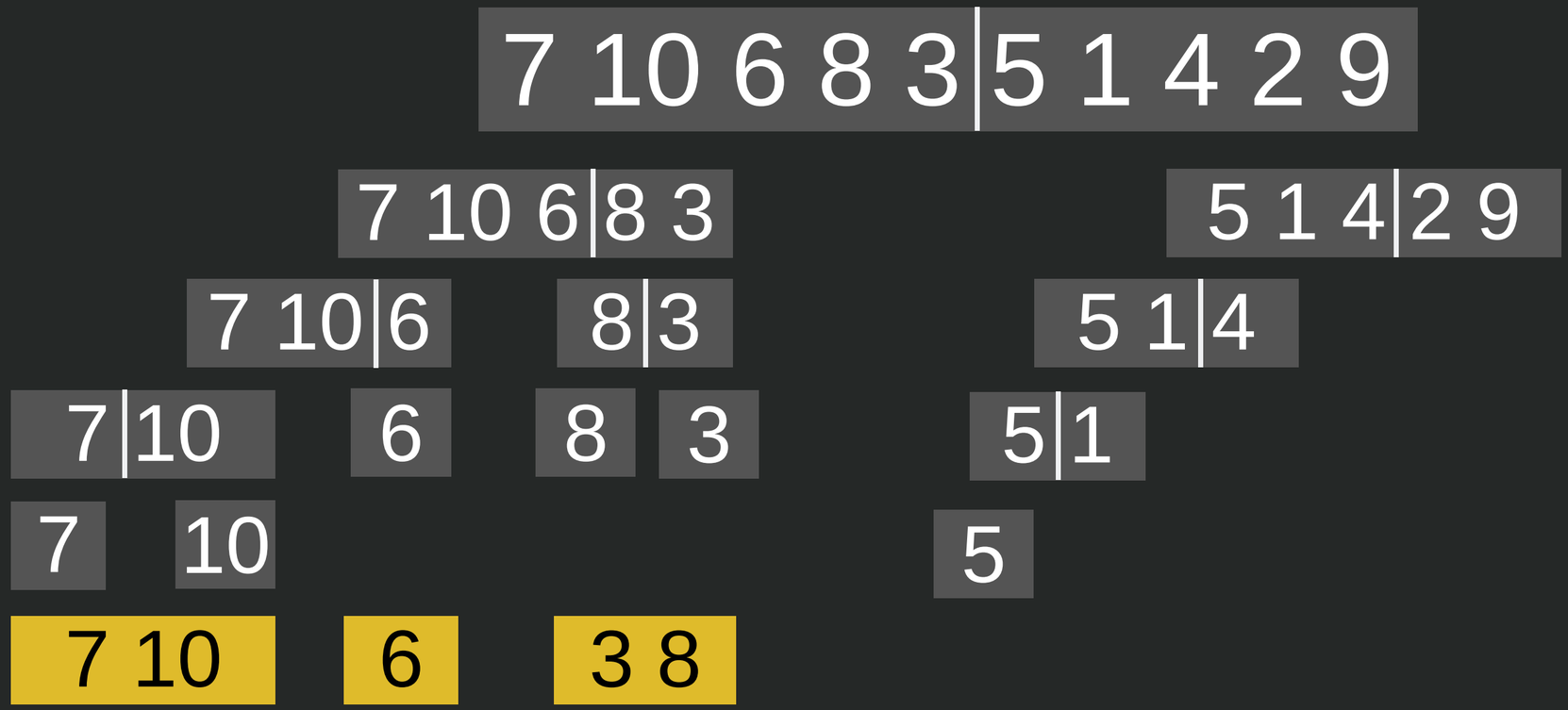

7 10 6 8 3 5 1 4 2 9
7 10 6 8 3
5 1 4 2 9
7 10 6
8 3
7 10
6
8
3
7
10
7 10
6
3 8

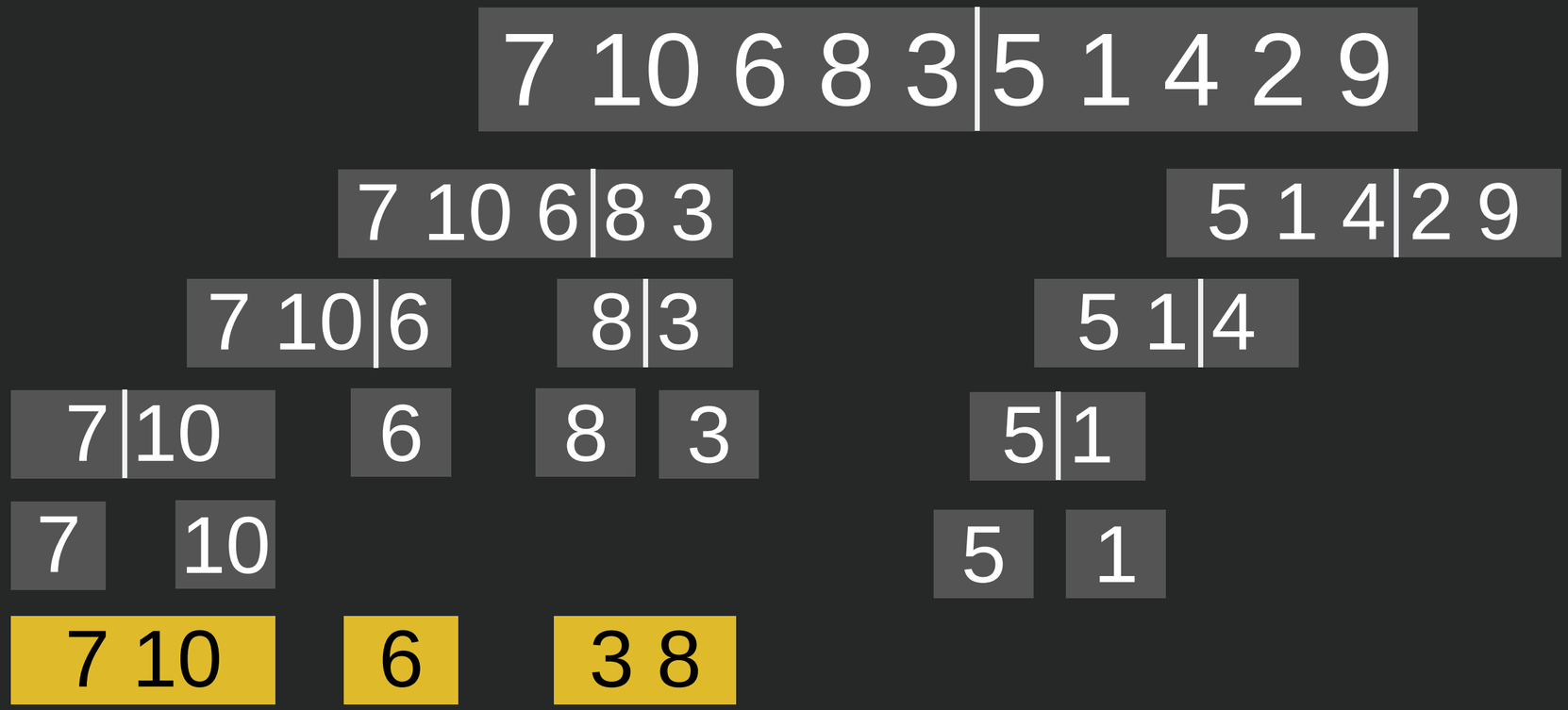

7 10 6 8 3 5 1 4 2 9
7 10 6 8 3
5 1 4 2 9
7 10 6
8 3
5 1 4
7 10
6
8
3
7
10
7 10
6
3 8

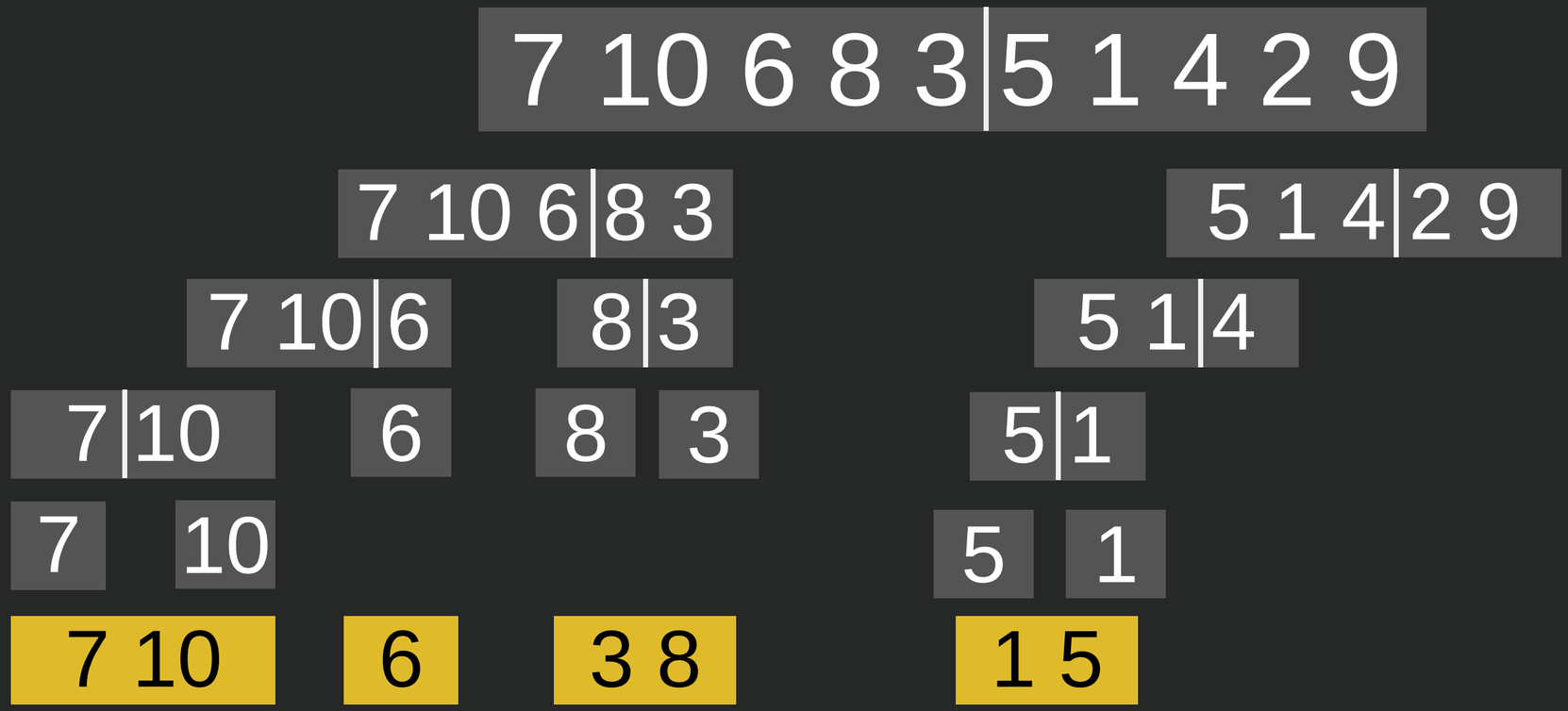

7 10 6 8 3 5 1 4 2 9
7 10 6 8 3
5 1 4 2 9
7 10 6
8 3
5 1 4
7 10
6
8
3
7
10
7 10
6
3 8

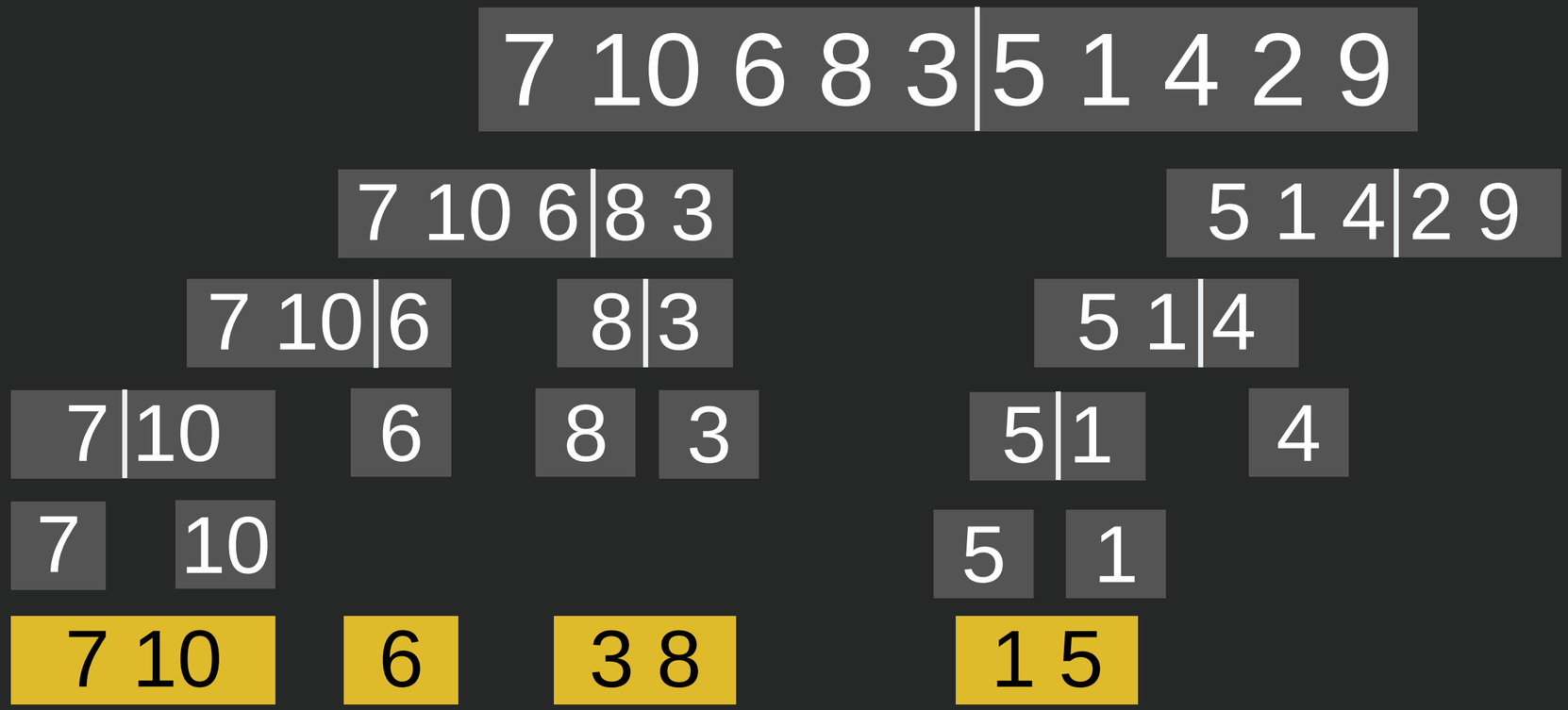

7 10 6 8 3 5 1 4 2 9
7 10 6 8 3
5 1 4 2 9
7 10 6
8 3
5 1 4
7 10
6
8
3
5 1
7
10
7 10
6
3 8

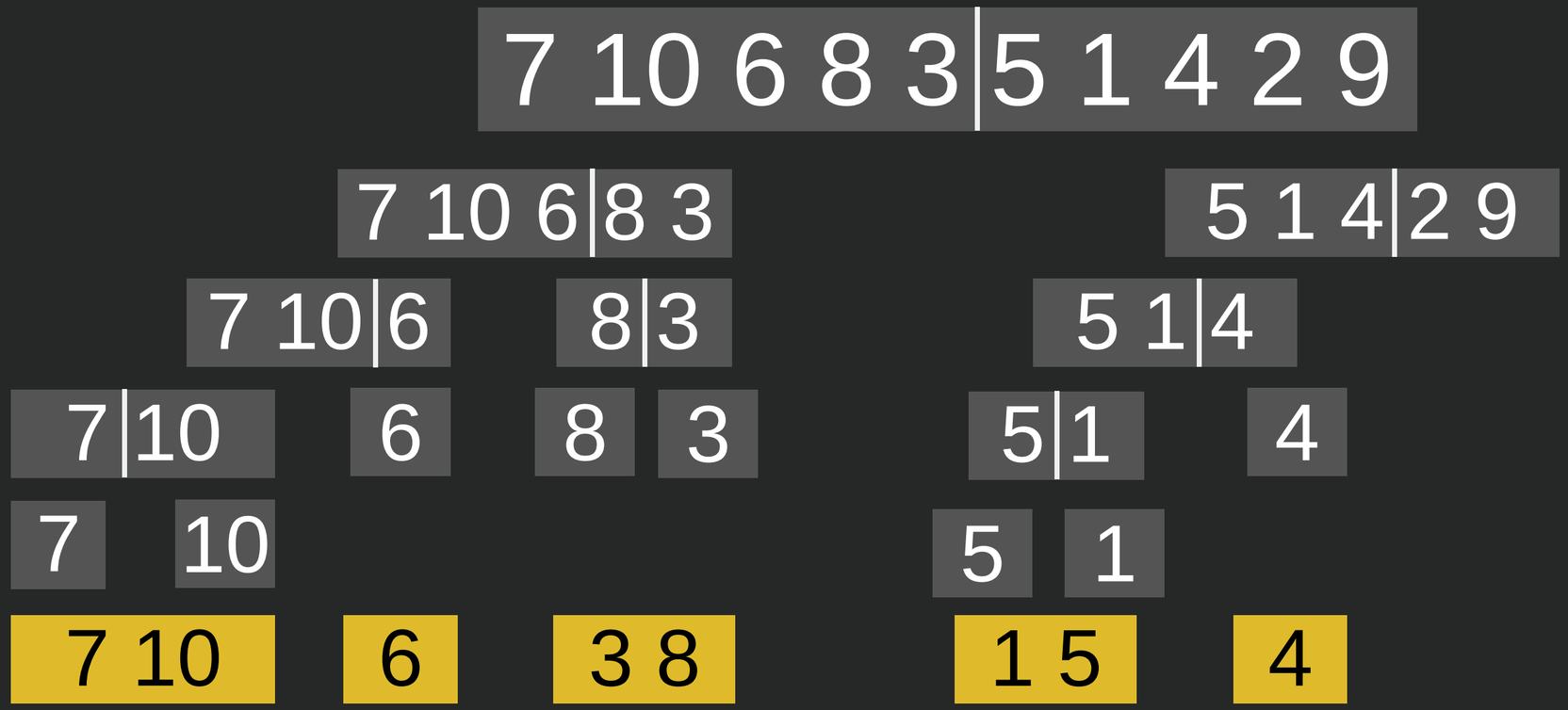

7 10 6 8 3 5 1 4 2 9
7 10 6 8 3
5 1 4 2 9
7 10 6
8 3
5 1 4
7 10
6
8
3
5 1
7
10
7 10
6
3 8

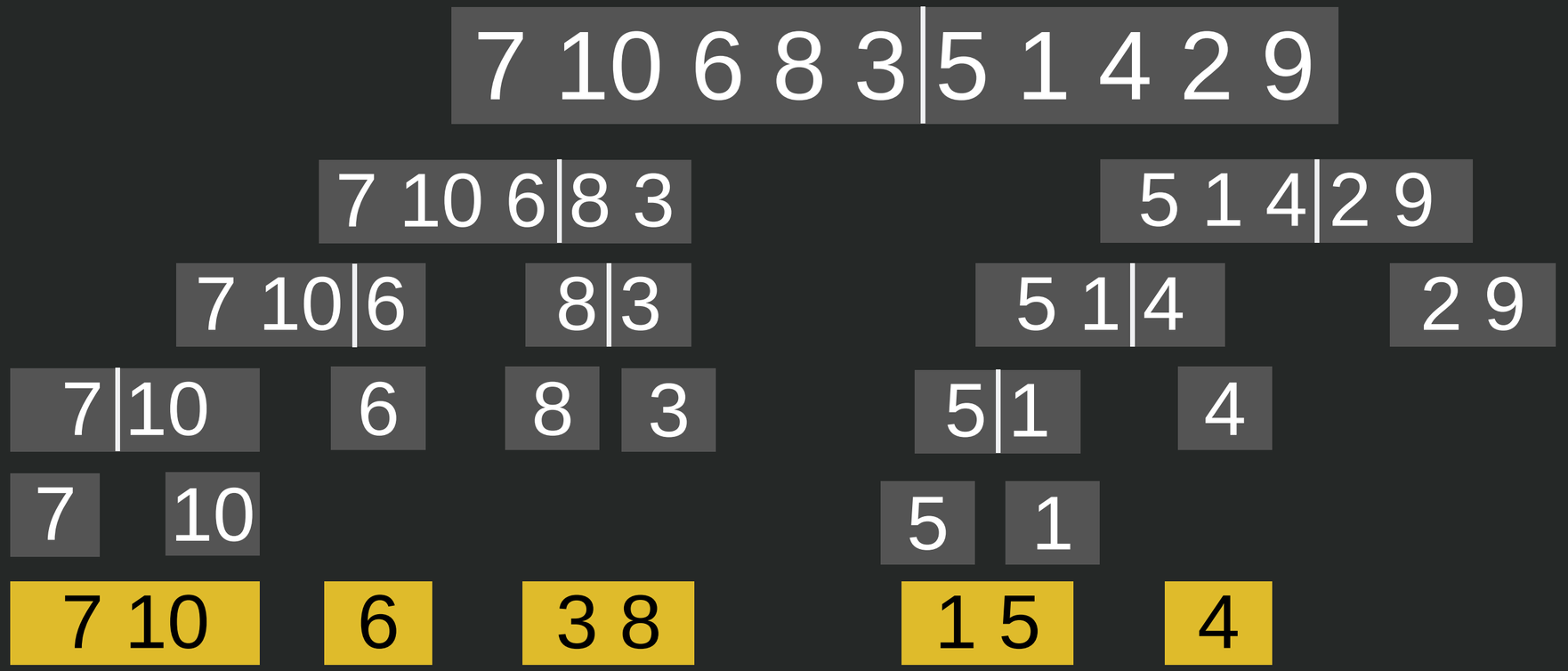

7 10 6 8 3 5 1 4 2 9
7 10 6 8 3
5 1 4 2 9
7 10 6
8 3
5 1 4
7 10
6
8
3
5 1
7
10
5
7 10
6
3 8

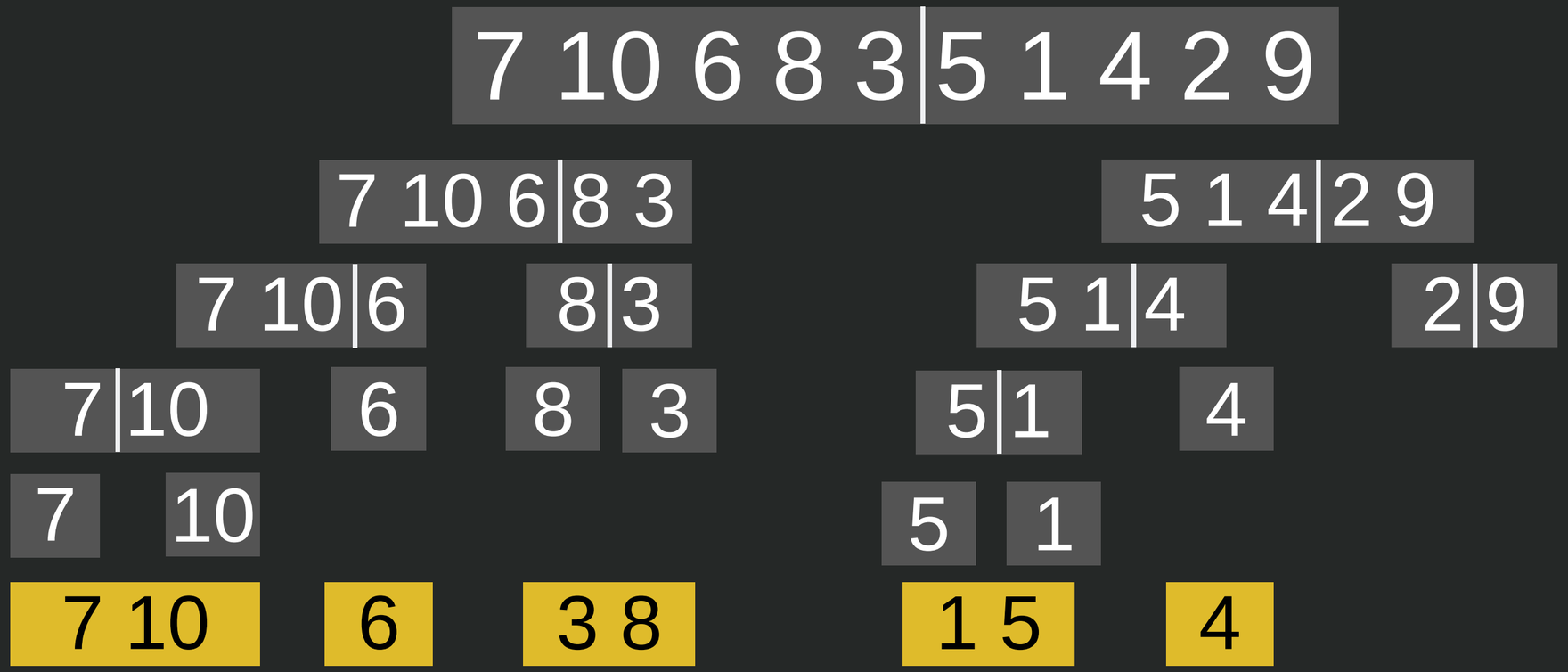

7 10 6 8 3 5 1 4 2 9
7 10 6 8 3
5 1 4 2 9
7 10 6
8 3
5 1 4
7 10
6
8
3
5 1
7
10
5
1
7 10
6
3 8

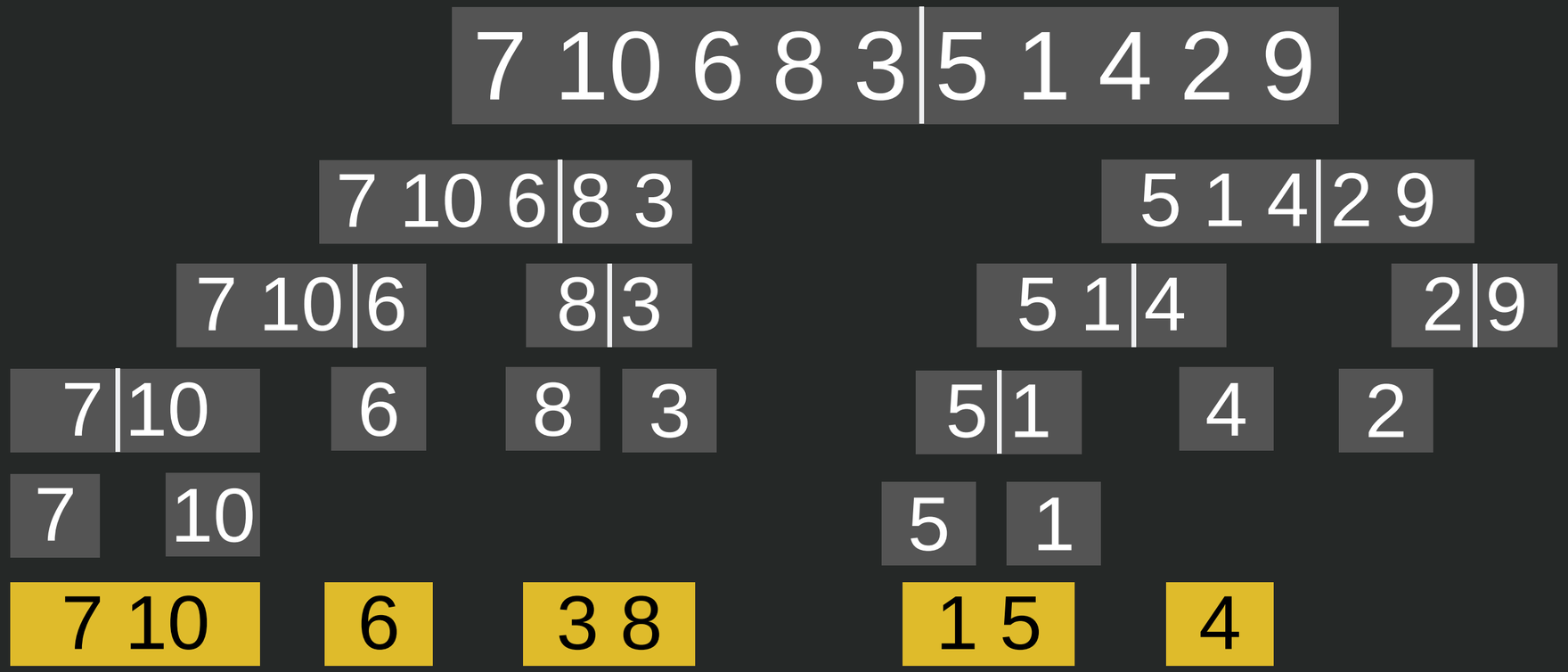

7 10 6 8 3 5 1 4 2 9
7 10 6 8 3
5 1 4 2 9
7 10 6
8 3
5 1 4
7 10
6
8
3
5 1
7
10
5
1
7 10
6
3 8
1 5

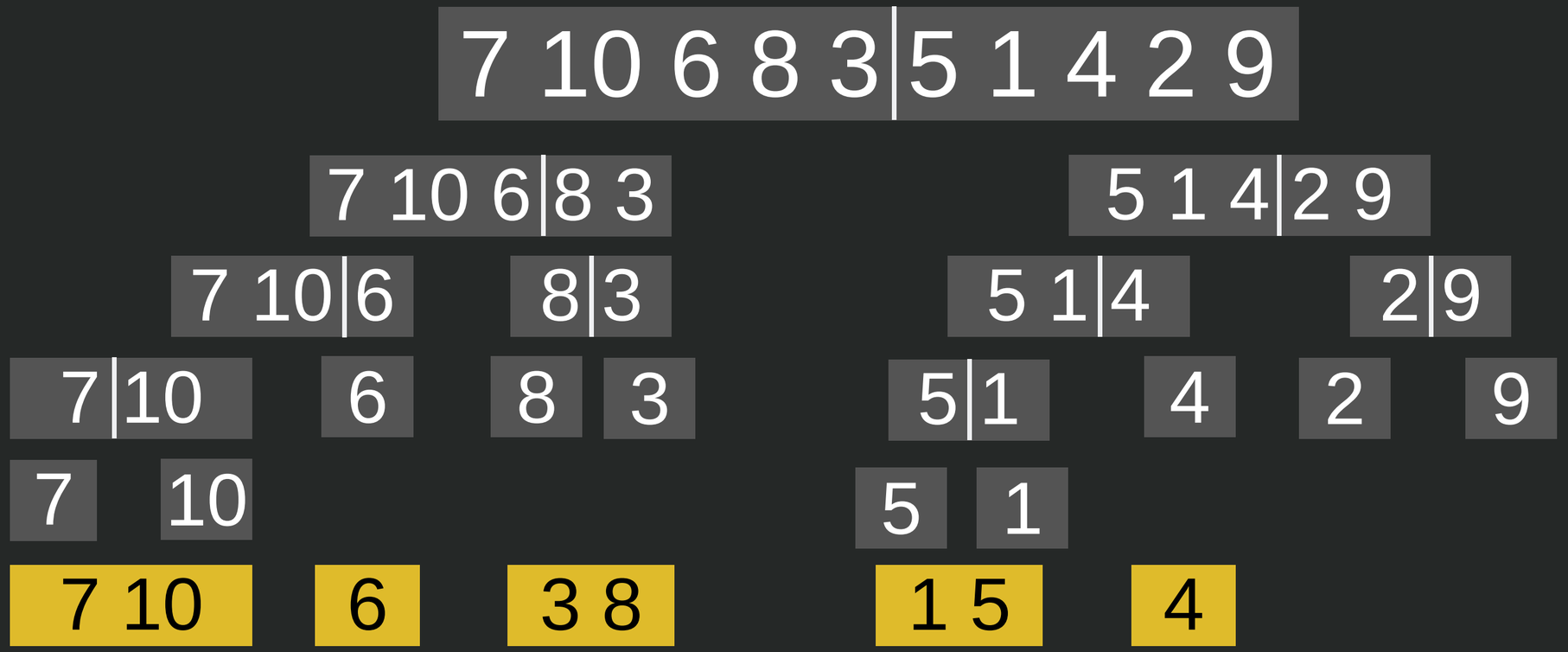

7 10 6 8 3 5 1 4 2 9
7 10 6 8 3
5 1 4 2 9
7 10 6
8 3
5 1 4
7 10
6
8
3
5 1
4
7
10
5
1
7 10
6
3 8
1 5

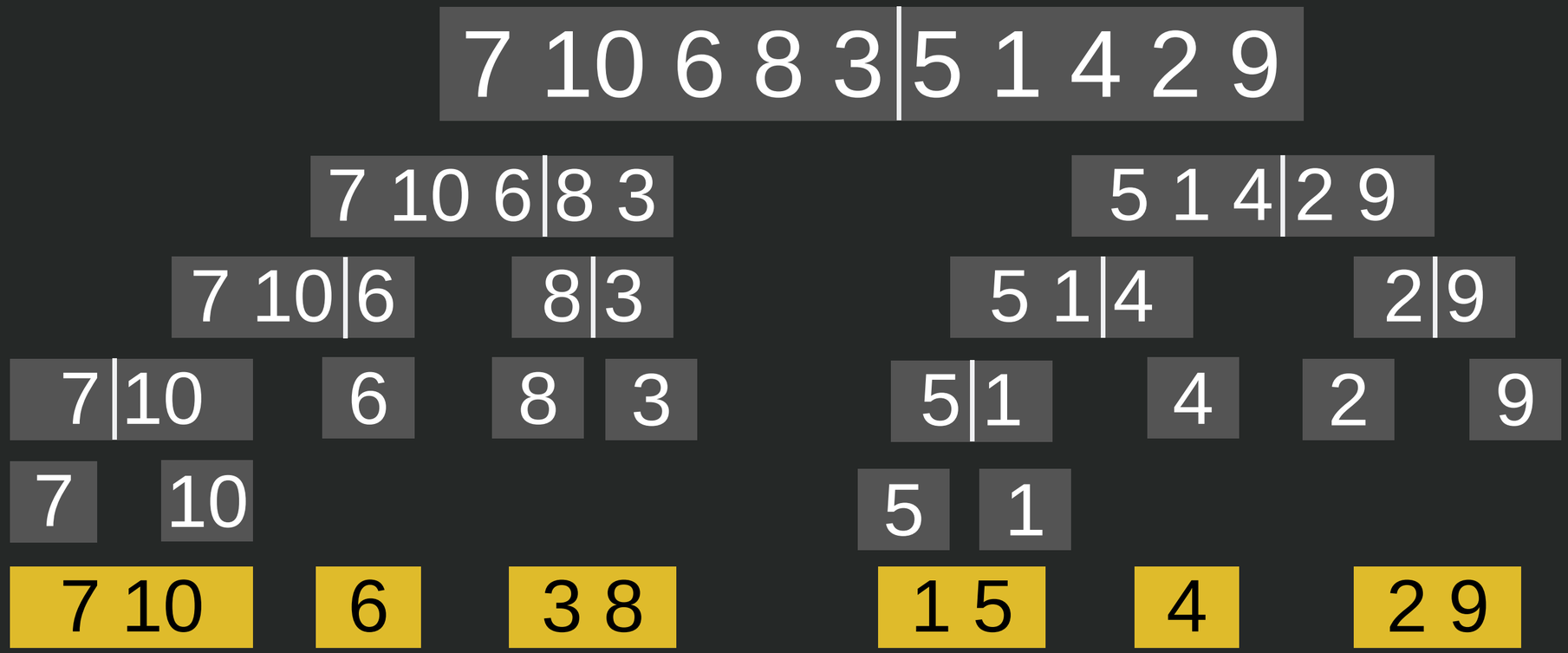

7 10 6 8 3 | 5 1 4 2 9
7 10 6 | 8 3
5 1 4 | 2 9
7 10 | 6
8 | 3
5 1 | 4
7 | 10
6
8
3
5 | 1
4
7
10
5
1
7 10
6
3 8
1 5
4

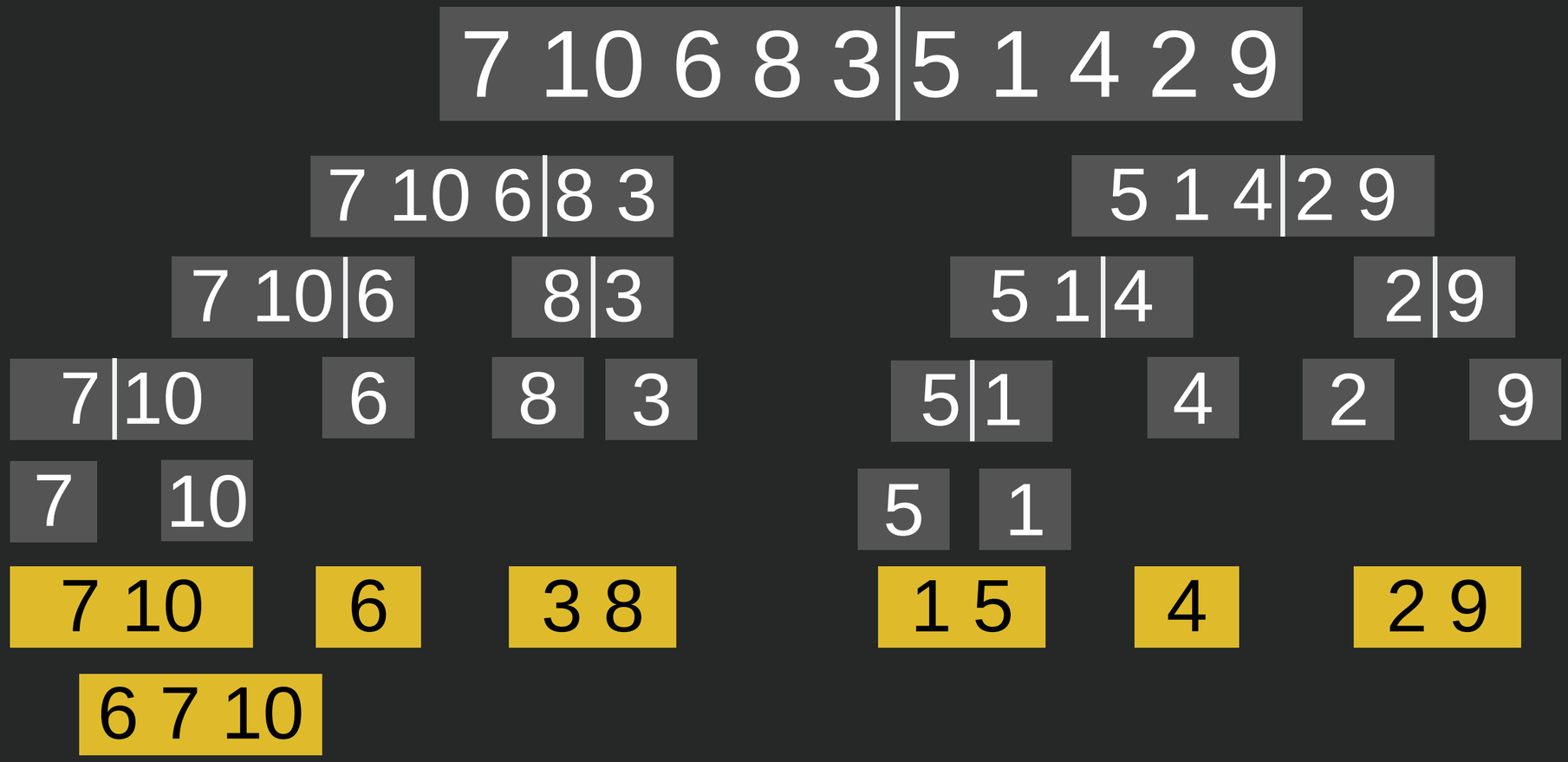

7 10 6 8 3 | 5 1 4 2 9
7 10 6 | 8 3
5 1 4 | 2 9
7 10 | 6
8 | 3
5 1 | 4
2 9
7 | 10
6
8
3
5 | 1
4
7
10
5
1
7 10
6
3 8
1 5
4

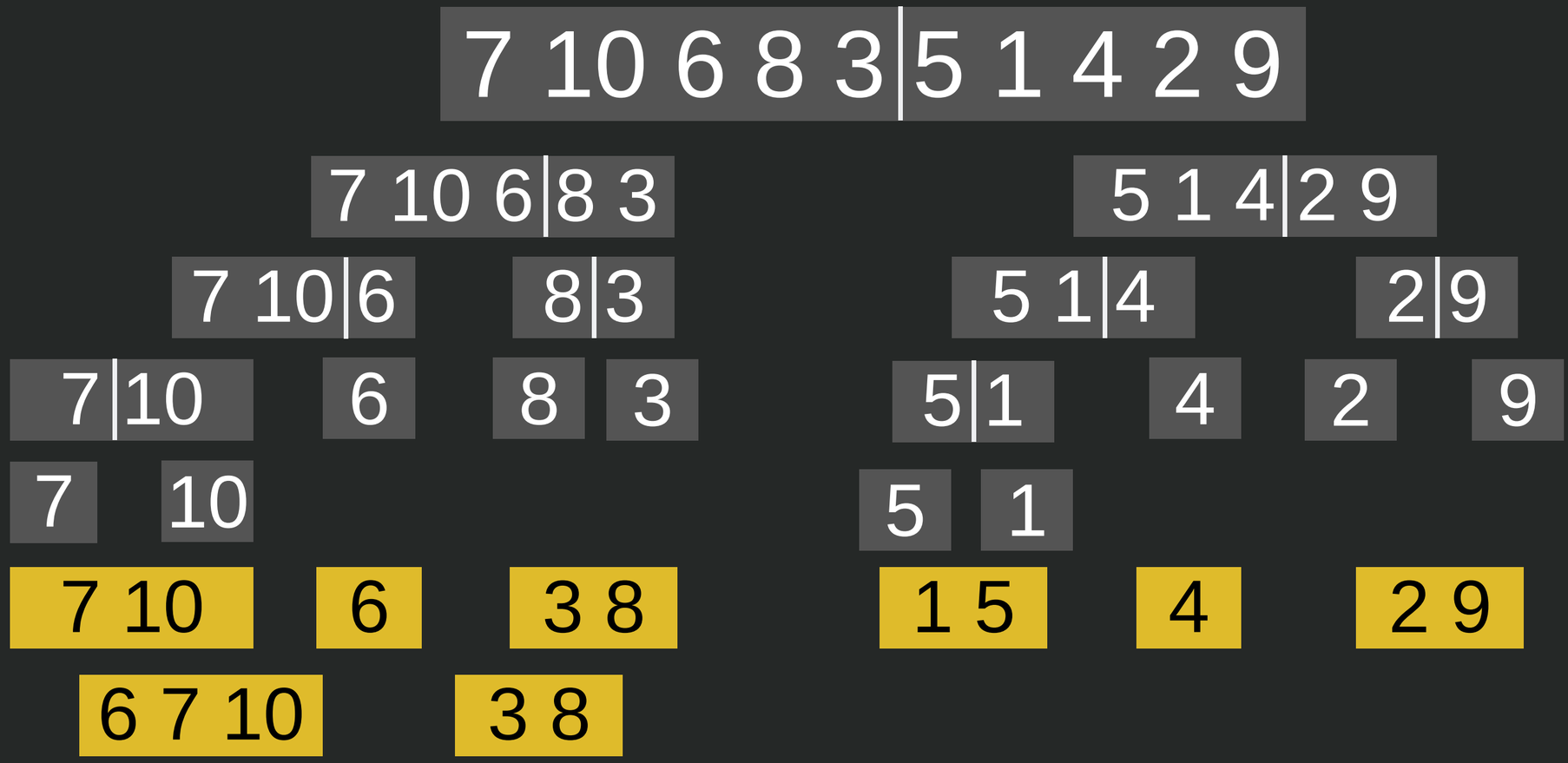

7 10 6 8 3 5 1 4 2 9
7 10 6 8 3
5 1 4 2 9
7 10 6 8 3
5 1 4 2 9
7 10 6 8 3
5 1 4 2 9
7 10 6 8 3
5 1 4 2 9
7 10 6 8 3
5 1 4 2 9

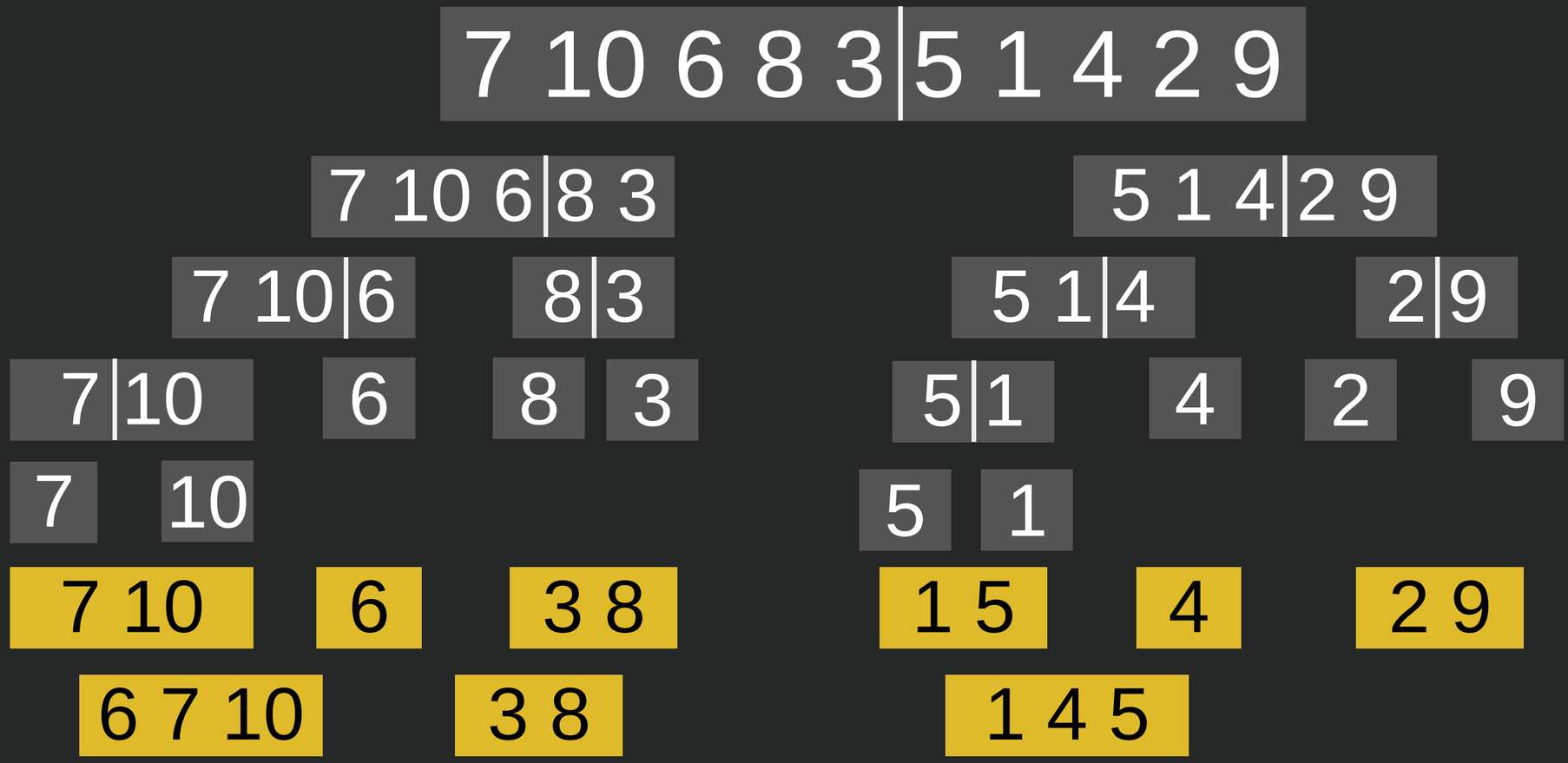

7 10 6 8 3 5 1 4 2 9
7 10 6 8 3
5 1 4 2 9
7 10 6 8 3
5 1 4 2 9
7 10 6 8 3
5 1 4 2 9
7 10 6 8 3
5 1 4 2 9
7 10 6 8 3
5 1 4 2 9

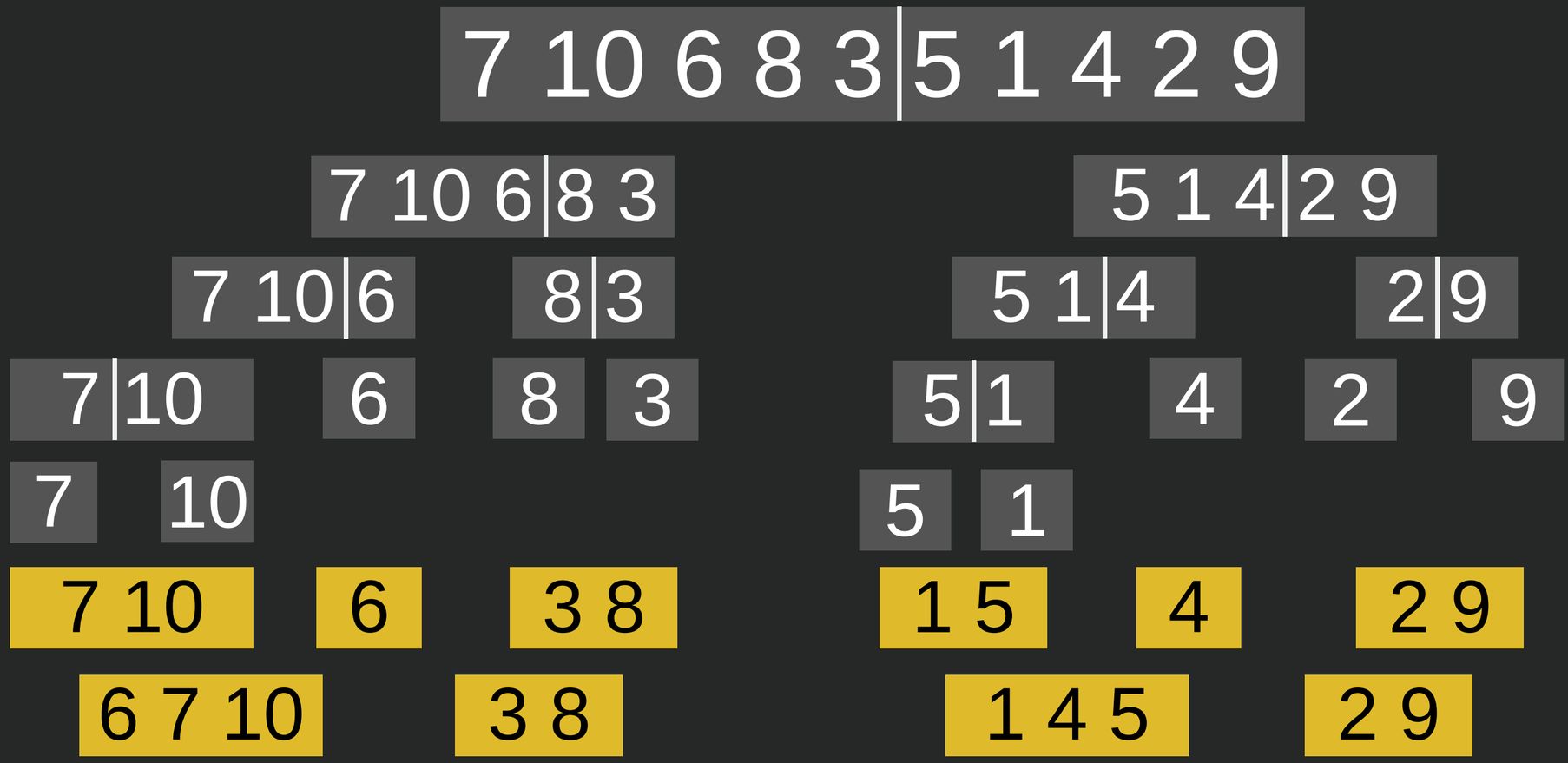

7 10 6 8 3 5 1 4 2 9
7 10 6 8 3
5 1 4 2 9
7 10 6 8 3
5 1 4 2 9
7 10 6 8 3
5 1 4 2 9
7 10 6 8 3
5 1 4 2 9
7 10 6 8 3
5 1 4 2 9
7 10 6 8 3
5 1 4 2 9

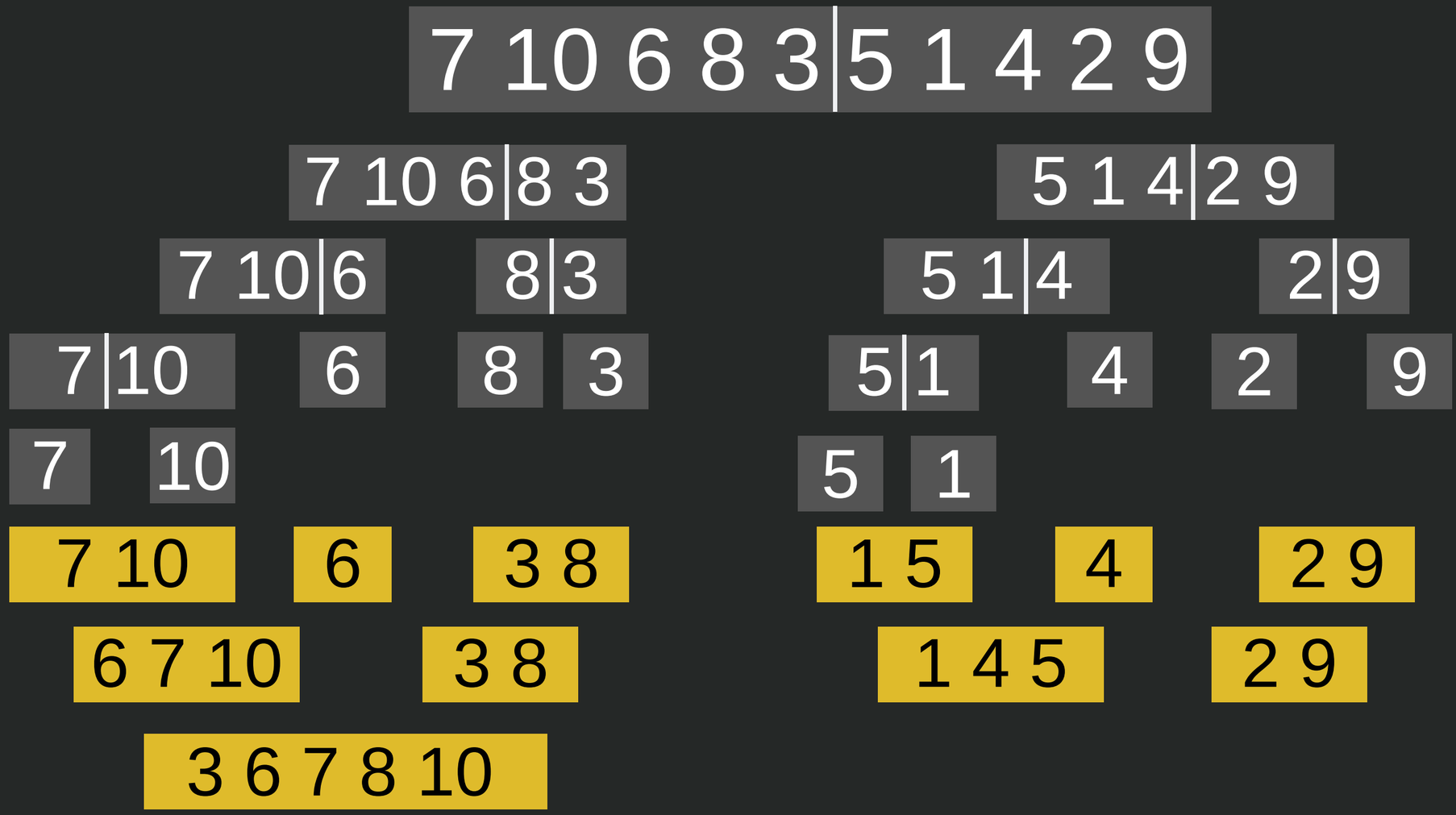

7 10 6 8 3 5 1 4 2 9
7 10 6 8 3
5 1 4 2 9
7 10 6 8 3
5 1 4 2 9
7 10 6 8 3
5 1 4 2 9
7 10 6 8 3
5 1 4 2 9
7 10 6 8 3
5 1 4 2 9
7 10 6 8 3
5 1 4 2 9

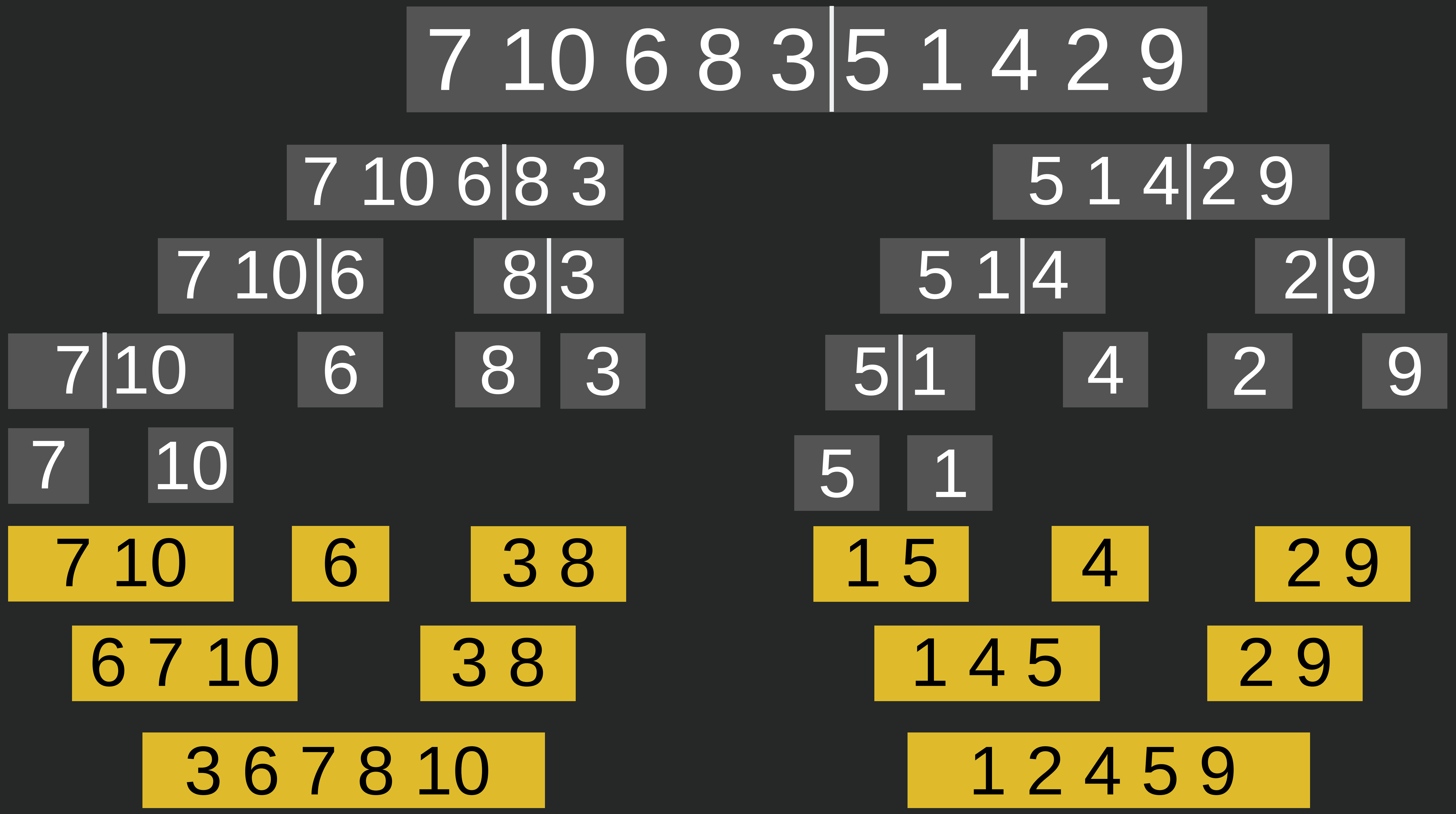

7 10 6 8 3 5 1 4 2 9
7 10 6 8 3
5 1 4 2 9
7 10 6 8 3
5 1 4 2 9
7 10 6 8 3
5 1 4 2 9
7 10 6 8 3
5 1 4 2 9
7 10 6 8 3
5 1 4 2 9
6 7 10

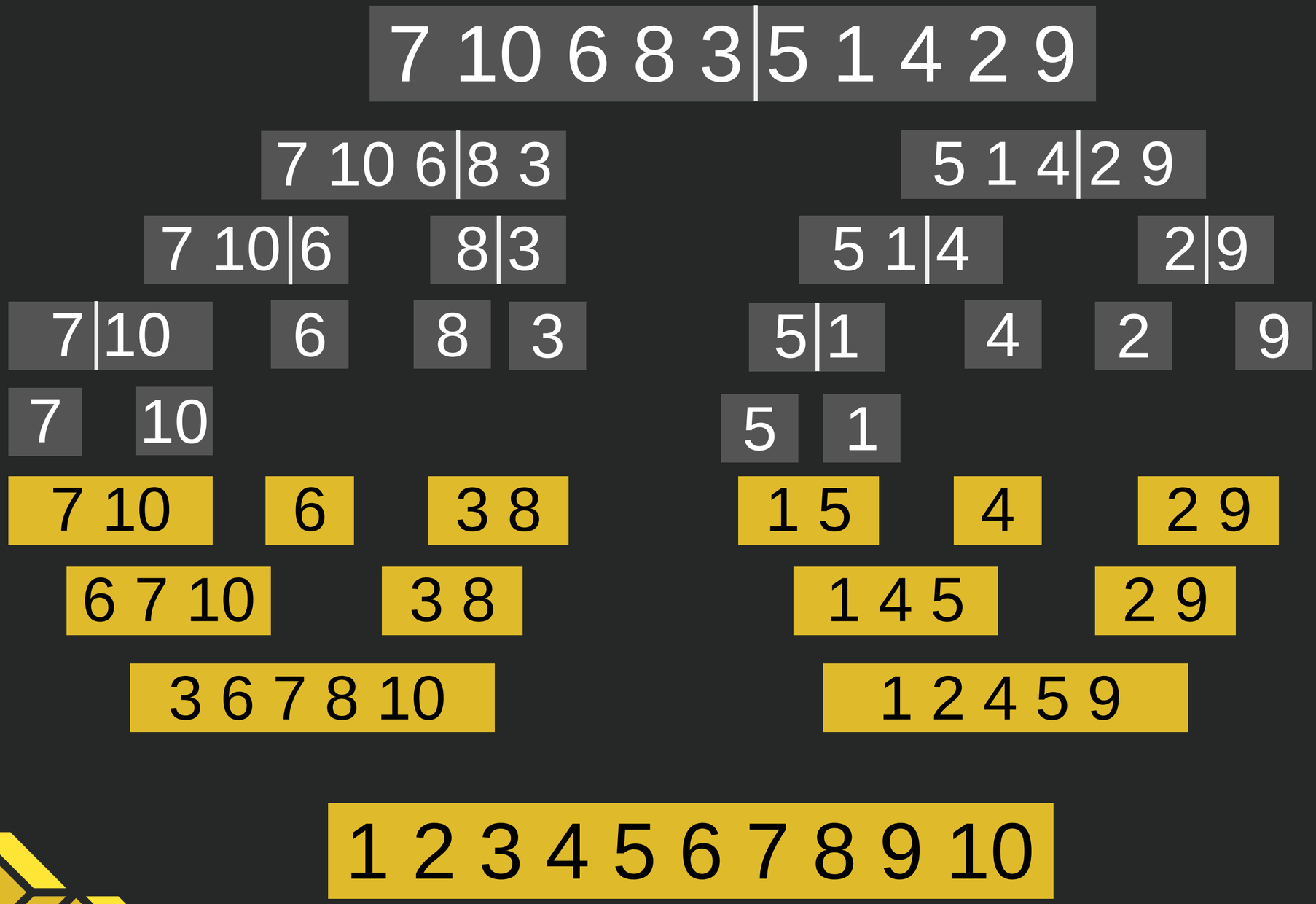

7 10 6 8 3 | 5 1 4 2 9
7 10 6 | 8 3
5 1 4 | 2 9
7 10 | 6
8 | 3
5 1 | 4
2 | 9
7 | 10
6
8
3
5 | 1
4
2
9
7
10
5
1
7 10
6
3 8
1 5
4
2 9
6 7 10
3 8


7 10 6 8 3 5 1 4 2 9
7 10 6 8 3
5 1 4 2 9
7 10 6
8 3
5 1 4
2 9
7 10
6
8
3
5 1
4
2
9
7
10
5
1
7 10
6
3 8
1 5
4
2 9
6 7 10
3 8
1 4 5


7 10 6 8 3 5 1 4 2 9
7 10 6 8 3
5 1 4 2 9
7 10 6 8 3
5 1 4 2 9
7 10 6 8 3
5 1 4 2 9
7 10 6 8 3
5 1 4 2 9
7 10 6 8 3
5 1 4 2 9
7 10 6 8 3
5 1 4 2 9
6 7 10 3 8
1 4 5 2 9


7 10 6 8 3 5 1 4 2 9
7 10 6 8 3
5 1 4 2 9
7 10 6 8 3
5 1 4 2 9
7 10 6 8 3
5 1 4 2 9
7 10 6 8 3
5 1 4 2 9
7 10 6 8 3
5 1 4 2 9
7 10 6 8 3
5 1 4 2 9
6 7 10 3 8
1 4 5 2 9
3 6 7 8 10


7 10 6 8 3|5 1 4 2 9
7 10 6|8 3
5 1 4|2 9
7 10|6
8|3
5 1|4
2|9
7|10
6
8
3
5|1
4
2
9
7
10
5
1
7 10
6
3 8
1 5
4
2 9
6 7 10
3 8
1 4 5
2 9
3 6 7 8 10
1 2 4 5 9


7 10 6 8 3 | 5 1 4 2 9
7 10 6 | 8 3
5 1 4 | 2 9
7 10 | 6
8 | 3
5 1 | 4
2 | 9
7 | 10
6
8
3
5 | 1
4
2
9
7
10
5
1
7 10
6
3 8
1 5
4
2 9
6 7 10
3 8
1 4 5
2 9
3 6 7 8 10
1 2 4 5 9
1 2 3 4 5 6 7 8 9 10

```
void mergeSort(int* vet, int inicio, int fim) {
    if (inicio < fim){
        int meio = floor((fim + inicio)/2);
        mergeSort(vet, inicio, meio);
        mergeSort(vet, meio+1, fim);
        merge(vet, inicio, meio, fim);
    }
}
```

Implementação

C++

```
void merge(int* v, int inicio, int meio, int fim) {
    int *aux, p1, p2, tamanho;
    bool fim1 = false, fim2 = false;

    tamanho = fim - inicio + 1;
    aux = new int[tamanho];
    p1 = inicio;
    p2 = meio + 1;
```

Implementação

C++

```cpp
if (aux != NULL){
    for (int i = 0; i < tamanho; i++){
        if (!fim1 && !fim2){
            if (v[p1] < v[p2]){
                aux[i] = v[p1++];
            } else {
                aux[i] = v[p2++];
            }
            if (p1 > meio) { fim1 = true; }
            if (p2 > fim) { fim2 = true; }
        } else {
            if (!fim1){
                aux[i] = v[p1++];
            } else {
                aux[i] = v[p2++];
            }
        }
    }
```

Implementação

C++

```
        for (int i = 0; i < tamanho; i++){
            v[i+inicio] = aux[i];
        }

    }
    delete(aux);
}
```

Implementação

C++

```python
import math

def mergeSort(vet):
    if len(vet) <= 1:
        return vet
    meio = math.ceil(len(vet)/2)
    a = mergeSort(vet[:meio])
    b = mergeSort(vet[meio:])

    vet = merge(a, b)
    return vet
```

Implementação

Python

```python
def merge(a, b):
    vet = []
    indexA = 0
    indexB = 0
    while indexA < len(a) and indexB < len(b):
            if a[indexA] < b[indexB]:
                vet.append(a[indexA])
                indexA += 1
            elif a[indexA] > b[indexB]:
                vet.append(b[indexB])
                indexB += 1
            else:
                vet.append(a[indexA])
                vet.append(b[indexB])
                indexA += 1
                indexB += 1
```

Implementação

Python

```
while indexA < len(a):
    vet.append(a[indexA])
    indexA += 1
while indexB < len(b):
    vet.append(b[indexB])
    indexB += 1
return vet
```

Implementação

Python

Referências

CORMEN, T.H. Algoritmos: Teoria e Prática. 3 ed. Elsevier, Estados Unidos. 2012.

MERGE SORT. GeeksForGeeks. Disponível em: <https://www.geeksforgeeks.org/merge-sort/>. Acesso em: 25 ago. 2019.

INSERTION SORT. GeekForGeeks. Disponível em: <https://www.geeksforgeeks.org/insertion-sort/>.Acesso em : 24 ago. 2019.

Obrigado!