Trabalho 2

Estrutura de Dados

MEMBROS: BRAYAN MARTINS, CARLOS DANIEL MARTINS

índice

Introdução

- **Filas de prioridade são estruturas de dados essenciais que facilitam a inserção de elementos com prioridades designadas, bem como a remoção do elemento de maior ou menor prioridade. Essas estruturas são cruciais em algoritmos de busca, otimização e em sistemas operacionais. Foram implementados algoritmos para Heap mínimo, Heap máximo e Heap de Fibonacci, explorando os conceitos práticos de cada um. A primeira implementação, o heap mínimo, foi desenvolvida com o objetivo prático de estabelecer padrões hierárquicos na busca de doenças e sintomas no âmbito da saúde. Já o heap máximo utilizou uma estrutura de árvore para otimizar buscas com o conceito de paginação, comum em biologia computacional, visando a agilidade e simplicidade na recuperação de dados. O Heap de Fibonacci, notavelmente o mais complexo, representou um grande desafio tanto em sua codificação quanto na explicação teórica. Seu propósito prático é permitir a manipulação eficiente de grandes volumes de dados em tempos reduzidos. Este trabalho, realizado no curso de Sistemas de Informação da Faculdade Antônio Meneghetti, inclui exemplos práticos aplicáveis a dados de variadas escalas, focando na simplificação e na eficiência da busca e manipulação de informações. O relatório a seguir detalha a implementação desses heaps, elucidando aspectos práticos do emprego dessas estruturas na pesquisa eficiente de dados. A metodologia adotada envolveu a implementação do heap mínimo, máximo e de Fibonacci, visando a otimização da coleta de dados.**

Definições e Propriedades

Heaps são árvores binárias. É importante deixar claro desde já que são árvores binárias, mas não são árvores binárias de pesquisa. Mais especificamente, duas propriedades definem o Heap:

1. **O valor de um nó é maior ou igual ao valor de seus filhos;**

2. **O Heap é uma árvore binária completa ou quase-completa da esquerda para a direita.**

A primeira propriedade é a que difere um Heap de uma árvore binária de pesquisa (BST). Na BST, os valores à esquerda de um nó são menores do que ele e os valores à direita são maiores. No Heap, ambos são menores ou iguais. Heaps que seguem essa propriedade são Heaps Máximos porque o maior valor sempre está na raiz. Há também Heaps Mínimos, onde o nó tem valor sempre menor ou igual ao seus filhos. Neste caso, o menor valor sempre está na raiz. Neste material, vamos utilizar o termo Heap como sinônimo de Heap Máximo.

Heap Minimo

- **Definição: No heap mínimo, o valor de cada nó é sempre maior ou igual ao valor de seu nó pai. Isso garante que o menor elemento esteja sempre na raiz do heap.**

- **Utilização: É particularmente útil em aplicações onde é frequente a necessidade de acessar o menor elemento rapidamente, como em algoritmos de caminho mais curto, onde você pode querer obter repetidamente o próximo vértice com a menor distância estimada.**

Heap Maximo

- **Definição: No heap máximo, o valor de cada nó é sempre menor ou igual ao valor de seu nó pai. Isso coloca o maior elemento na raiz do heap.**

- **Utilização: É ideal para aplicações que requerem acesso frequente ao maior elemento, como em sistemas de gerenciamento de recursos, onde você pode precisar alocar o recurso mais significativo primeiro.**

Heap de Fibonacci

Características

- **Estrutura: Consiste em uma coleção de árvores que são min-heaps, ou seja, o valor de cada nó pai é menor ou igual aos valores de seus nós filhos. No entanto, diferentemente de um heap binário, as árvores no heap de Fibonacci não são necessariamente binárias.**
-
- **Propriedades: Uma propriedade distintiva dos heaps de Fibonacci é que eles permitem uma maior flexibilidade na estrutura das árvores, o que minimiza a necessidade de reorganizações frequentes.**

Heap de Fibonacci

Operações

- **Inserção: As inserções são feitas adicionando a nova árvore à lista de árvores do heap. Esta operação é feita em tempo constante amortizado, o que é extremamente rápido.**
- **Encontrar mínimo: Encontrar o elemento mínimo é uma operação direta, pois o heap mantém um ponteiro para o nó com o valor mínimo.**
- **Remoção do mínimo: A remoção do mínimo é mais complexa e envolve a remoção do nó mínimo, seguida da reestruturação das árvores restantes para manter as propriedades do heap. Esta operação é mais eficiente do que em heaps binários, mas ainda assim pode ser mais lenta do que a inserção.**
- **Diminuir chave: Diminuir o valor de uma chave em um heap de Fibonacci é muito eficiente e é uma das principais vantagens dessa estrutura. A operação pode desvincular e reorganizar subárvores para manter as propriedades de min-heap.**
- **União: A união de dois heaps de Fibonacci é feita concatenando suas listas de árvores, o que também é uma operação rápida.**

Heap de Fibonacci

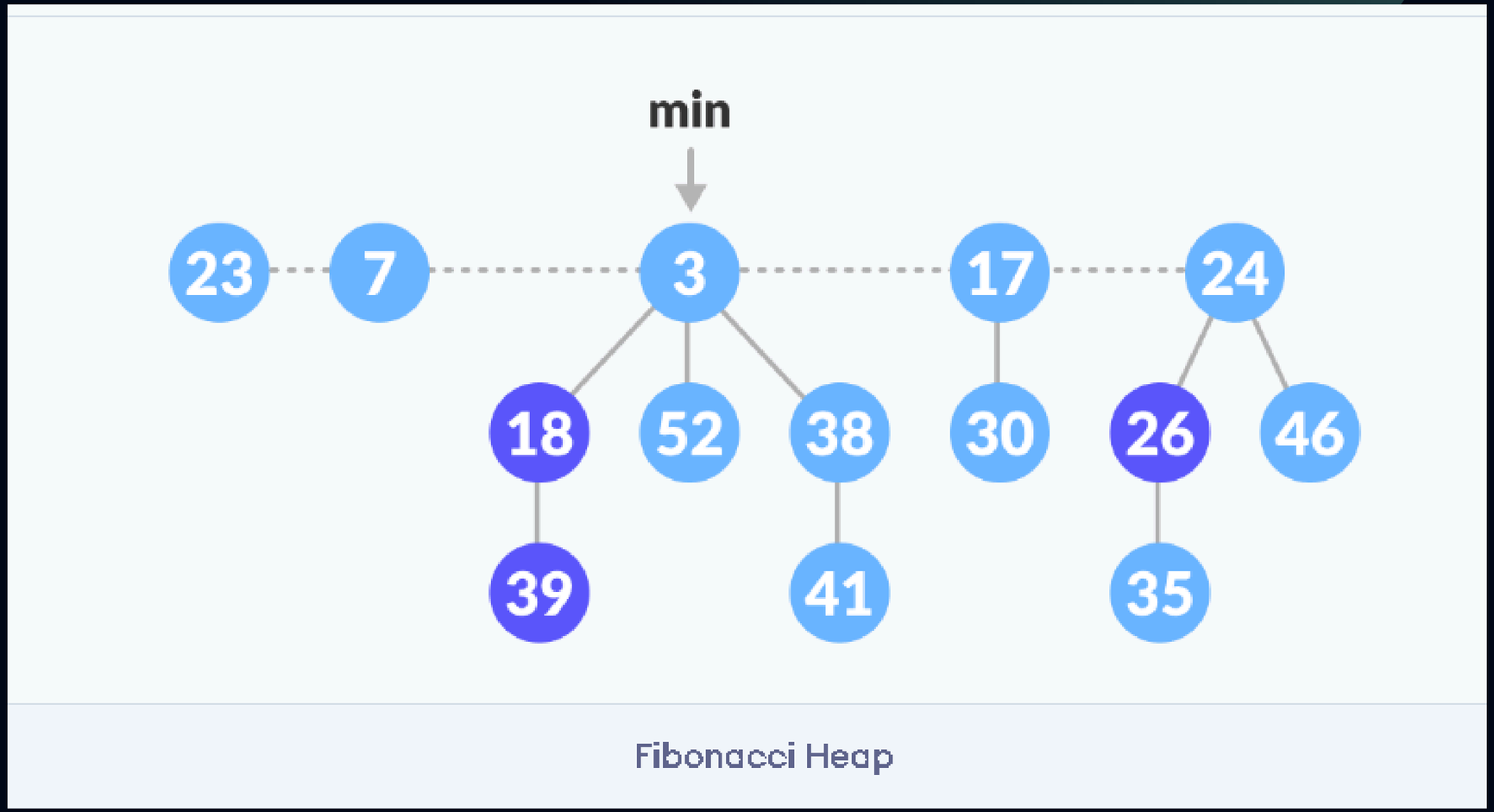
Vantagens

- **Eficiência para aplicações específicas: Em situações onde as operações de diminuição de chave e união são predominantes, os heaps de Fibonacci são extremamente eficientes e podem superar outras estruturas de dados de heap.**

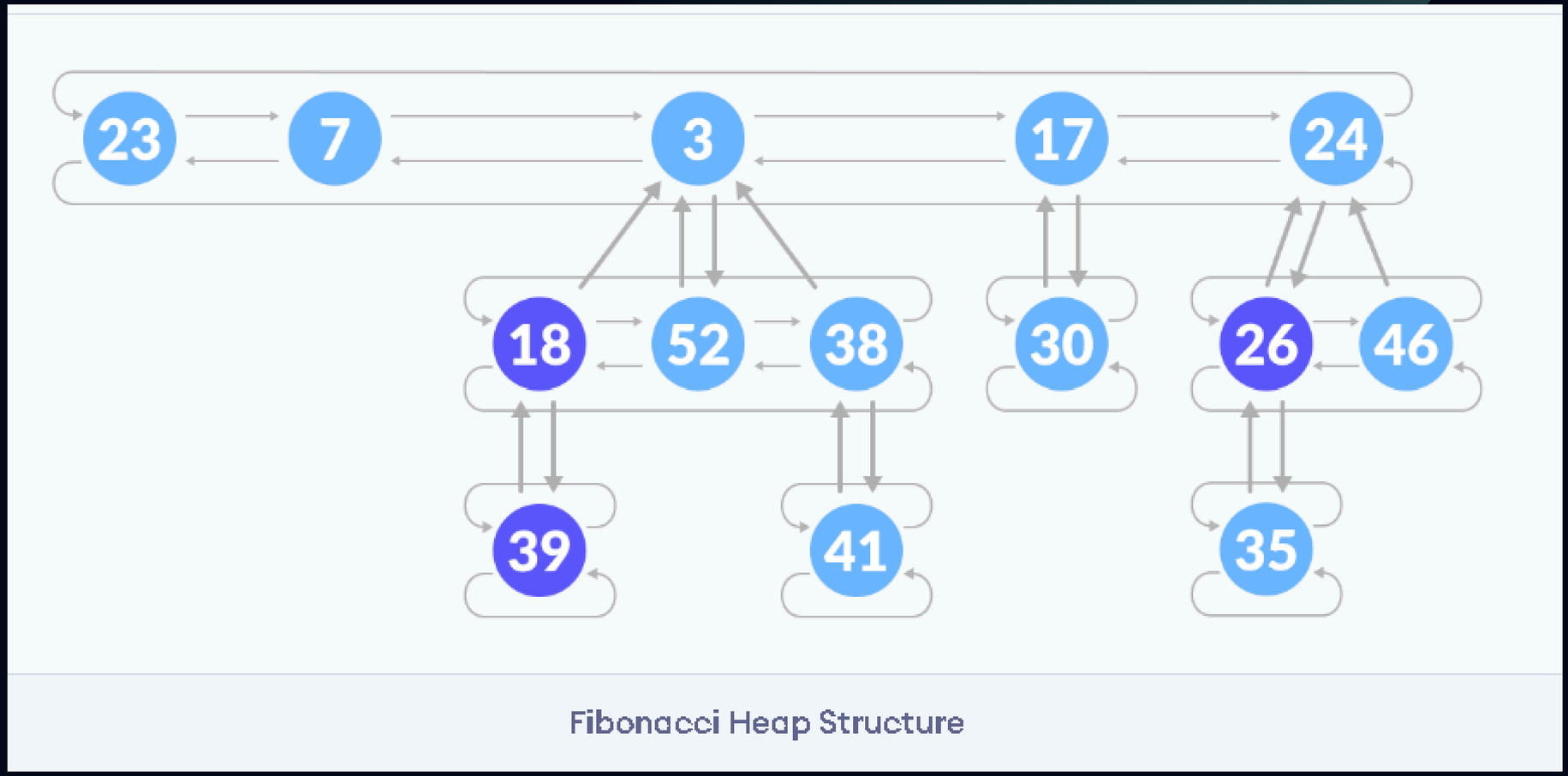
Desvantagens

- **Complexidade: A implementação e a manutenção de um heap de Fibonacci são mais complexas do que as de heaps binários simples, o que pode ser uma desvantagem em situações onde a simplicidade e a previsibilidade são mais valorizadas.**

Heap de Fibonacci

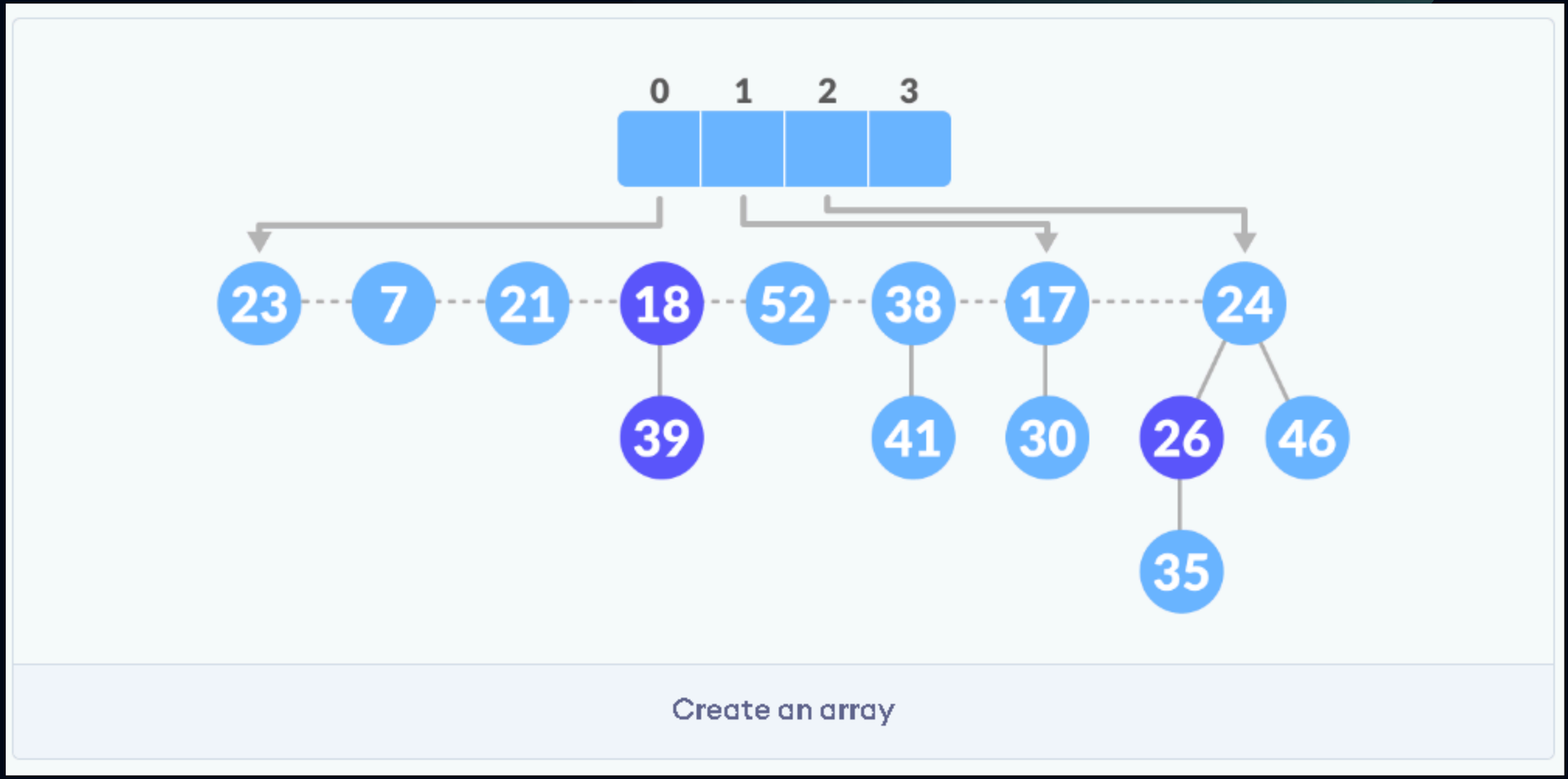


Heap de Fibonacci

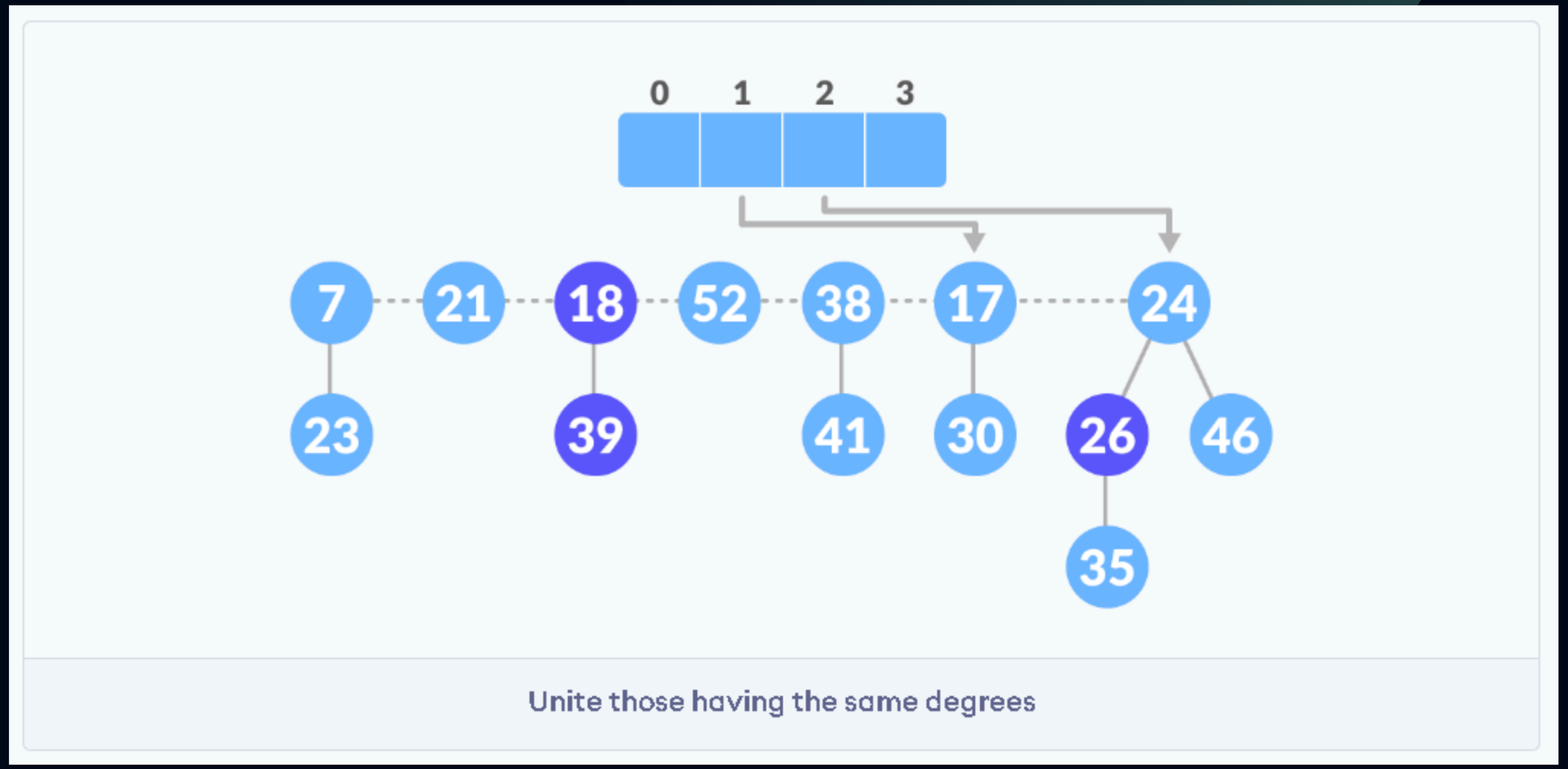


Fibonacci Heap Structure

Heap de Fibonacci



Heap de Fibonacci



Heap de Fibonacci