2
Preliminares

Neste capítulo, revisaremos algumas noções combinatórias básicas relacionadas a complexos simpliciais em dimensão arbitrária, introduziremos as relações topológicas existentes entre as células de um complexo e, por fim, apresentaremos algumas noções básicas de programação genérica.

2.1
Simplexos

Definição 2.1 **Simplexo:** Um *simplexo* σ de dimensão k, é o fecho convexo de $k+1$ pontos $\{v_0,\dots,v_k\}, v_i \in \mathbb{R}^m$, em posição geral, isto é, os vetores $v_1 - v_0, v_2 - v_0, \dots, v_k - v_0$ são linearmente independentes.

Por convenção, chamamos um simplexo de dimensão k de k–simplexo e, quando o contexto for entendido, chamaremos um 0–simplexo de *vértice*, um 1–simplexo de *aresta*, um 2–simplexo de *triângulo* e um 3–simplexo de *tetraedro*(figura 2.1). Os pontos $v_0,\dots,v_k$ de um k–simplexo σ são chamados de *vértices* de σ.

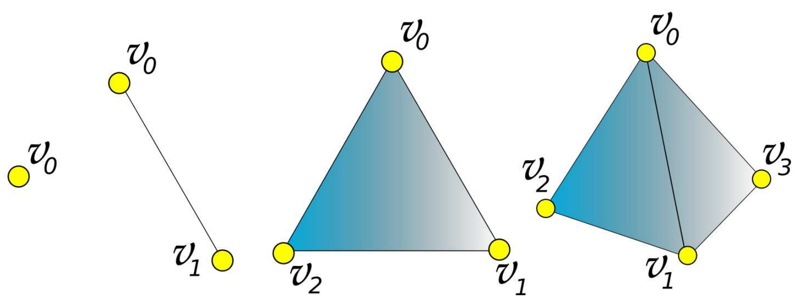

Figura 2.1: Simplexos de dimensão 0, 1, 2 e 3 em $\mathbb{R}^m$.

Dado um k–simplexo σ, denotamos sua dimensão por $dim(\sigma) = k$ e o conjunto dos seus vértices por V_σ.

Definição 2.2 **p–Face:** Um p–simplexo γ gerado a partir de um subconjunto $V_\gamma \subseteq V_\sigma$, dos vértice de um k–simplexo σ, com $p \le k$, é chamado de uma *p–face* de σ(figura 2.2).

Quando não houver ambigüidade, a dimensão de γ será omitida e diremos apenas que γ é uma *face* de σ. Se γ é face de σ dizemos também que σ é *incidente* a γ e que σ é uma *co–face* de γ.

Um simplexo γ é uma *face própria* de um simplexo σ se $dim(\gamma) < dim(\sigma)$. O simplexo gerado a partir do subconjunto vazio $V_\gamma = \emptyset$ é, por convenção, uma (-1)–face de todo k–simplexo, com $k \geq 0$.

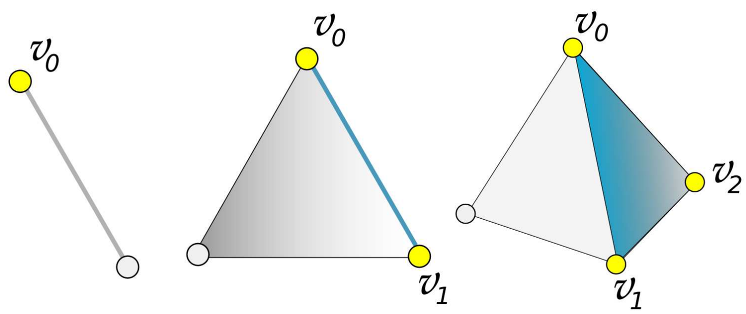

Figura 2.2: Exemplos de faces dos simplexos de dimensão 1, 2 e 3 em $\mathbb{R}^m$

Definição 2.3 Bordo de um Simplexo: O *bordo* de um p–simplexo σ, denotado por $\partial\sigma$, é a coleção de todas as faces próprias de σ(figura 2.3).

O *interior* de um simplexo σ, é definido como $Int(\sigma) = \sigma - \partial\sigma$.

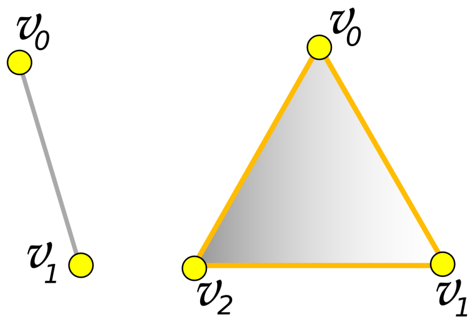

Figura 2.3: Bordo e interior de simplexos de dimensão 1 e 2 em $\mathbb{R}^m$.

2.2 Complexos Simpliciais

Definição 2.4 Complexo Simplicial: Um *complexo simplicial* Σ é um conjunto finito de simplexos tais que:

1. Se $\sigma \in \Sigma$, então todas as faces de σ pertencem a Σ.
2. Se $\sigma, \gamma \in \Sigma$, então $\sigma \cap \gamma$ é uma face própria de σ e γ.

A condição 2 da definição (2.4) impede que existam interseções indevidas entre simplexos de um complexo simplicial.

Por exemplo, 3–simplexos podem ter interseção com outros simplexos apenas em vértices, arestas ou triângulos comuns, 2–simplexos apenas em arestas e vértices comuns, e assim por diante (figuras 2.4 e 2.5).

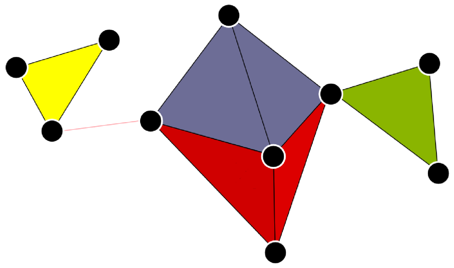

Figura 2.4: Complexo simplicial

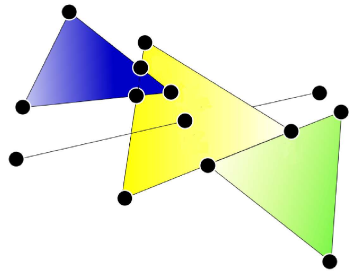

Figura 2.5: Complexo inválido

Definimos a dimensão de um complexo simplicial Σ como o número inteiro $d = max\{dim(\sigma) \mid \sigma \in \Sigma\}$, e dizemos que Σ é um *complexo simplicial d–dimensional*, ou simplesmente de *d–complexo simplicial*.

O *poliedro* de um d–complexo simplicial Σ imerso em $\mathbb{R}^m$, com $0 \leq d \leq m$, denotado por $|\Sigma|$, é o subconjunto de $\mathbb{R}^m$ definido pela união, como conjunto de pontos, de todos os simplexos de Σ. Um *subcomplexo simplicial* de Σ é qualquer subconjunto Σ^* composto por simplexos de Σ tal que Σ^* é também um complexo simplicial.

2.3
Relações entre Simplexos

Definição 2.5 Estrela: A *estrela* de um simplexo $\sigma \in \Sigma$, denotada por $\text{star}(\sigma, \Sigma)$, é a união de todos os simplexos $\gamma \in \Sigma$ que são co–face de σ.

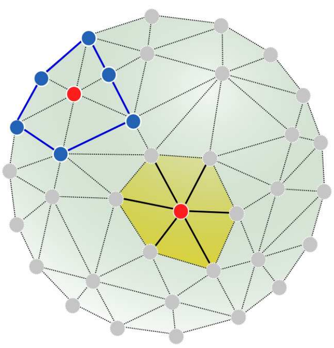

Figura 2.6: Estrela e Elo de vértices de uma esfera.

Definição 2.6 Elo: O *Elo* de um simplexo $\sigma \in \Sigma$, denotado por $\text{link}(\sigma, \Sigma)$, é o conjunto de simplexos $\gamma \in \Sigma$ tais que:

1. γ é face de algum simplexo $\psi \in \text{star}(\sigma, \Sigma)$.
2. $\gamma \notin \text{star}(\sigma, \Sigma)$.

A figura 2.6 ilustra a estrela (em amarelo) e o elo (em azul) de um dos vértices de um 2–complexo em $\mathbb{R}^3$.

Um simplexo $\sigma \in \Sigma$ é chamado de *simplexo topo* se $\text{star}(\sigma, \Sigma) = \{\sigma\}$. Se um d–complexo Σ é tal que todos os seus simplexos topo são d–simplexos, então Σ é um complexo *regular* (figuras 2.8, 2.7).

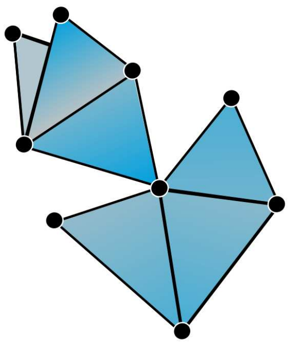

Figura 2.7: 2–complexo regular

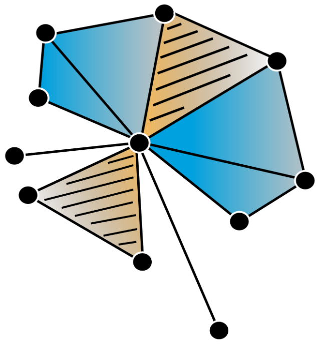

Figura 2.8: 3–complexo não regular

Definição 2.7 Simplexos p–Adjacentes: Dois simplexos σ e γ são *p–adjacentes* quando existe uma p–face comum a eles (figuras 2.10, 2.9).

Em alguns momentos omitiremos o grau de adjacência entre simplexos para tornar a notação mais simples. Por exemplo, dois k–simplexos que compartilham uma $(k-1)$–face e dois vértices incidentes a uma mesma aresta serão chamados apenas de *adjacentes*.

Dois simplexos que não são nem incidentes e nem adjacentes são denominados *disjuntos*.

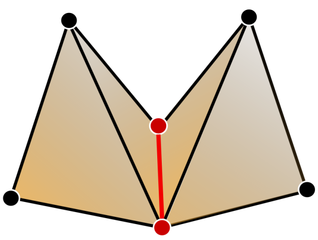

Figura 2.9: Simplexos 1–adjacentes

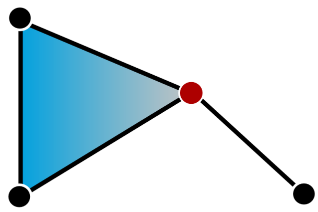

Figura 2.10: Simplexos 0–adjacentes

Definição 2.8 Simplexos h–Conectados: Dois simplexos ψ e γ são *h–conectados* se e somente se existe uma seqüência de simplexos $(\sigma_i)_{i=0}^{n}$ tal que:

1. Dois simplexos consecutivos σ_{i-1}, σ_i são h–adjacentes;
2. ψ e γ são faces de σ_0 e σ_n respectivamente.

A figura 2.11 mostra exemplos de simplexos 0–conectados e 1–conectados.

2.4 Componentes Conexas

Um subcomplexo Σ^* de um complexo simplicial Σ é *h–conexo* se e somente se todos os seu vértices são h–conectados. Um subcomplexo maximal em Σ^*, h–conexo, é chamado *h–componente conexa* de Σ.

A figura 2.11 mostra um complexo simplicial composto de uma 0–componente conexa.

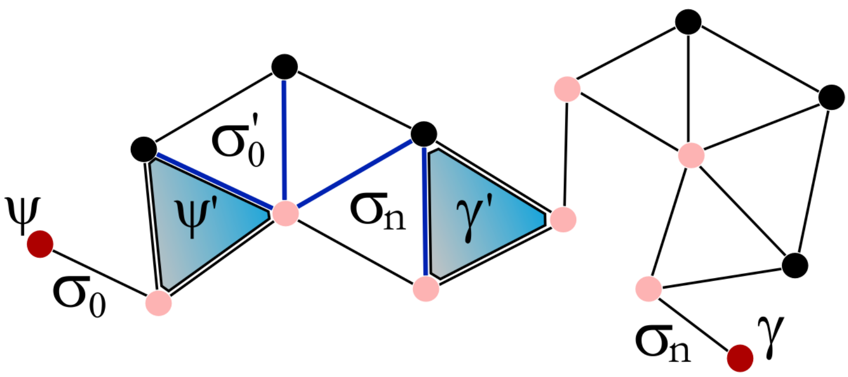

Figura 2.11: ψ e γ são 0–conectados e ψ' e γ' são 1–conectados. O complexo simplicial Σ é uma 0–componente conexa.

Definição 2.9 Componente Conexa: Seja Σ^* um subcomplexo simplicial de Σ, dizemos que Σ^* é uma *componente conexa* de Σ se e somente se Σ^* é uma 0–componente conexa de Σ.

2.5 Variedades

Um $(d$–$1)$–simplexo σ de um d–complexo simplicial Σ é um *$(d$–$1)$–simplexo variedade*(figura 2.12) se e somente se existem no máximo dois d–simplexos em Σ incidentes a σ. Caso σ não seja um simplexo variedade, o chamamos de *simplexo não–variedade*(figura 2.13).

Definição 2.10 Pseudo–Variedade Combinatória: Um d–complexo Σ, $|\Sigma| \subset \mathbb{R}^m$, é uma *pseudo–variedade combinatória* de dimensão d se, e só se:

1. Σ é um complexo regular $(d-1)$–conectado.
2. Todo $(d-1)$–simplexo $\sigma \in \Sigma$ é um $(d-1)$–simplexo variedade.

Para simplificar a notação, chamaremos uma pseudo–variedade combinatória de dimensão de d de *pseudo–d–variedade*. A figura 2.14 mostra um exemplo de pseudo–2–variedade. Um m–complexo simplicial Σ, imerso em $\mathbb{R}^m$ tem as seguintes propriedades:

1. Σ é uma pseudo–m–variedade.
2. Os simplexos topo incidentes a um $(d$–$2)$–simplexo σ podem ser ordenados em torno de σ.

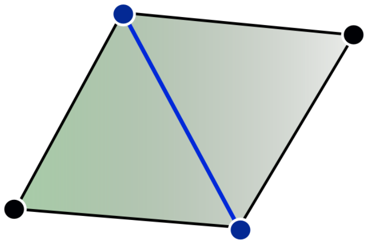

Figura 2.12: Simplexos variedade

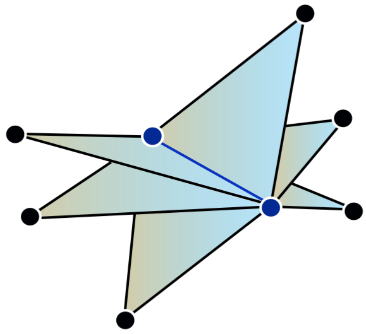

Figura 2.13: Simplexos não–variedade

Observe que se Σ é um d–complexo simplicial imerso em $\mathbb{R}^m$, $m > d$, então Σ não é necessariamente uma pseudo–variedade, pois podem existir muitos d–simplexos na estrela de um $(d-1)$–simplexo $\in \Sigma$. Entretanto, a propriedade 2 continua valendo para todos $(d-1)$–simplexos.

Definição 2.11 Variedade Combinatória: Uma pseudo–k–variedade Σ, $|\,\Sigma\,| \subset \mathbb{R}^m$, tal que, para todo vértice $v \in \Sigma$, a $\text{star}(v, \Sigma)$ é homeomorfa a $\mathbb{R}^k$ ou $\mathbb{R}^k_+$ é chamada de *Variedade Combinatória de dimensão k*.

Por simplicidade, chamaremos uma variedade combinatória de dimensão k de k–variedade (figura 2.15).

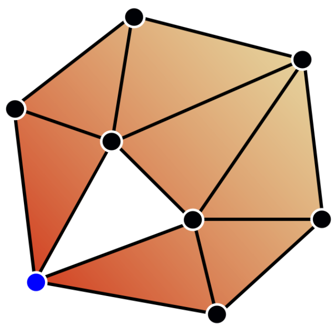

Figura 2.14: Pseudo–2–variedade

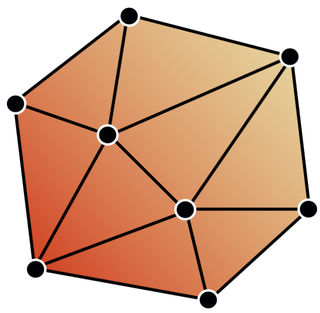

Figura 2.15: 2–variedade

Definição 2.12 Orientabilidade: Seja Σ uma k–variedade combinatória. A orientação de dois k–simplexos adjacentes σ e γ pertencentes a Σ é *coerente* se o $(k-1)$–simplexo ψ que compartilham tem orientação oposta em cada um dos simplexos. A variedade Σ é *orientável* se podemos escolher uma orientação coerente para todos os seus simplexos (figura 2.16).

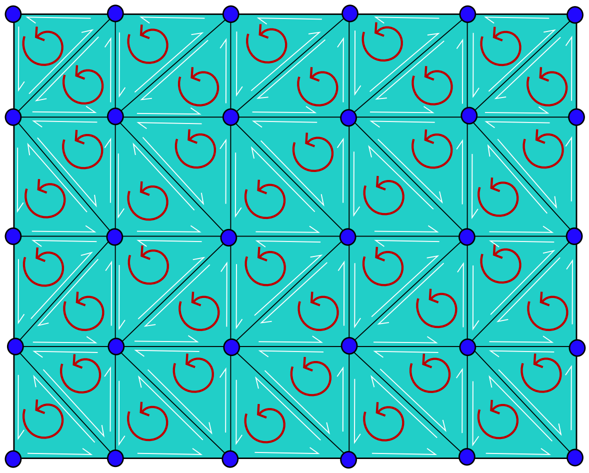

Figura 2.16: 2–variedade combinatória orientada no sentido anti–horário.

Deste ponto em diante, uma k–variedade Σ sempre significará uma k–variedade combinatória orientada, e denotaremos por $\mathbf{n}_k$ o número de k–simplexos existentes em Σ.

2.6 Bordo de Variedades

Definição 2.13 Simplexos de Bordo: Um $(k-1)$–simplexo de uma k–variedade incidente a apenas um k–simplexo é chamado de *simplexo de bordo*.

Todas as faces de um simplexo de bordo também são simplexos de bordo. Os simplexos que não são de bordo são chamados *simplexos interiores*.

Definição 2.14 Bordo de Variedades: O *bordo* de uma k–variedade Σ, denotado por $\partial\Sigma$, é a união de todos os seus simplexos de bordo.

Observe que o bordo de uma k–variedade é uma $(k-1)$–variedade sem bordo. A figura 2.17 ilustra uma 3–variedade e seus simplexos de bordo e interior.

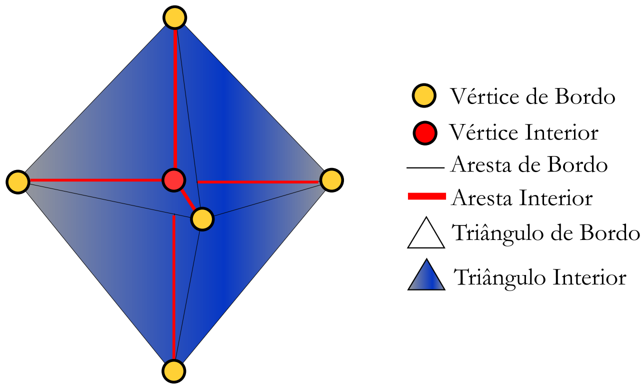

Figura 2.17: Simplexos de bordo e interior de uma 3–variedade.

2.7
Relações Topológicas em Variedades

De acordo com (Floriani and Hui 2003), uma estrutura de dados otimizada deve apresentar soluções eficientes para o cálculo de relações topológicas entre simplexos. Tais operações podem ser representadas através das *funções de resposta*.

Definição 2.15 Funções de resposta às interrogações topológicas: Seja Σ um m–complexo e $\sigma \in \Sigma$ um p–simplexo. As *funções de resposta às interrogações topológicas do tipo* $R_{pq}(\sigma)$ são relações topológicas que retornam, todos os q–simplexos γ de Σ que não são disjuntos de σ.

Por exemplo, $R_{00}(\sigma)$ retorna todos os vértices σ^* tais que σ, σ^* são incidentes a uma mesma aresta de Σ (figura 2.18).

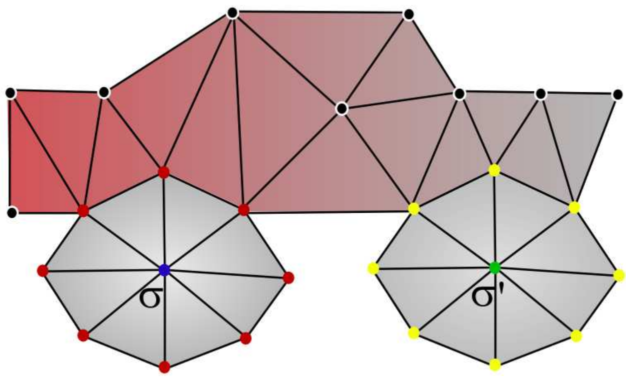

Figura 2.18: $R_{00}(\sigma)$ e $R_{00}(\sigma^{'})$

De maneira mais geral, quando $p < q$, $R_{pq}(\sigma)$ representa o conjunto de q–simplexos pertencentes a star(σ). Quando $p > q$, $R_{pq}(\sigma)$ consiste no conjunto dos q–simplexos que são face de σ. Por fim, quando $p = q$, $R_{pq}(\sigma)$ obtém o conjunto dos p–simplexos adjacentes ao simplexo σ.

A figura 2.19 mostra todos os casos possíveis para a função de retorno de uma aresta em uma 2–variedade.

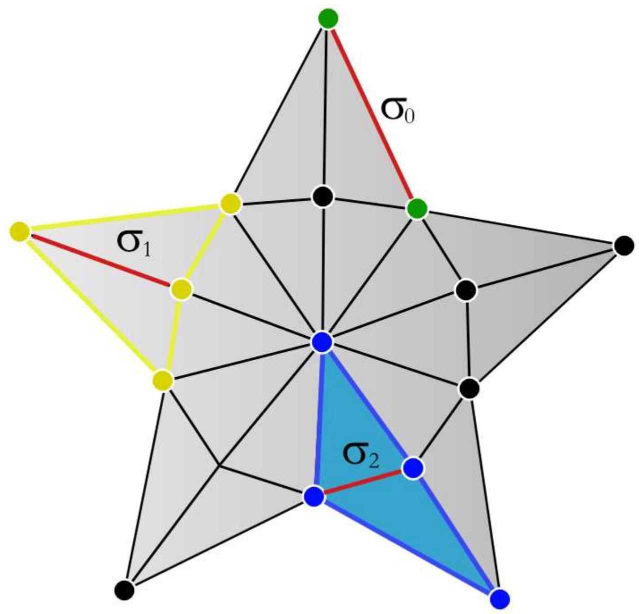

Figura 2.19: $R_{10}(\sigma_0)$, $R_{11}(\sigma_1)$ e $R_{12}(\sigma_2)$

2.8 Programação Genérica

Utilizaremos conceitos de programação genérica na construção das estruturas de dados propostas neste trabalho. O conceito mais importante que abordaremos é o de contêiner. Em termos gerais, um *contêiner* é um objeto que contém e pode organizar outros objetos. Existem vários tipos de contêineres conhecidos, que permitem trabalhar com os objetos de diversas maneiras. Cada tipo de contêiner deve fornecer recursos e procedimentos que nos permitam manipular os dados armazenados. Neste trabalho, utilizamos a implemetação de contêineres genéricos da biblioteca STL (*Standart Template Library*) da linguagem C++.

A seguir faremos uma breve descrição alguns tipos de contêiner:

- **Vector:** Um vetor é um *array* unidimensional cujos dados podem ser acessados aleatoriamente.
- **Deque:** Uma deque é uma fila com dois finais, ou seja, permite adicionar e remover elementos de ambas as extremidades. Uma vantagem de uma deque é que esta suporta o acesso aleatório de elementos.
- **Queue:** Um contêiner queue é uma fila de elementos que permite que adicionemos objetos em seu final, e que elementos do início sejam

removidos. Não podemos acessar elementos de uma fila aleatóriamente, assim como não é possível seu percorrimento.

- **Stack:** Uma stack é uma construção de programação que opera de forma análoga a uma pilha de pratos: elementos são inseridos e removidos do topo da pilha.
- **List:** Um contêiner List é uma lista de elementos e pode ser simplesmente ou duplamente encadeada. No caso de ser simplesmente encadeada, cada elemento da lista é ligado ao próximo elemento do conjunto. Já no caso de uma lista duplamente encadeada, cada elemento é ligado ao elemento anterior e posterior na lista. Uma lista duplamente encadeada suporta a remoção e inserção de elementos em tempo constante, independente do seu tamanho. Já numa lista simplesmente encadeada apenas a inserção é realizada em tempo constante.
- **Heap:** Uma priority queue é uma fila cujos elementos são ordenados de acordo com um critério de prioridade pré–definido.
- **Set:** Um set se assemelha a um conjunto matemático de objetos, sem repetição de elementos. Operações como união, interseção, diferença, etc são alguns dos recursos proporcionados pelo contêiner.
- **Multiset:** Um multiset é também tratado como um conjunto matemático, mas não existe a restrição de que os elementos sejam únicos.
- **Map:** Um contêiner map permite armazenar dados em pares chave/valor onde a chave é utilizada para acessar o valor associado. Em um contêiner map, cada chave é única.
- **Multimap:** Um contêiner multimap é análogo a um map, mas permite a associação de mais de um valor por chave.

Detalharemos as características técnicas do contêiner map já que ele será utilizado em vários algoritmos durante a construção das estruturas CHE e CHF.

Contêiner Map: Um map é um contêiner que nos permite associar um objeto conhecido como chave, a outro objeto chamado de valor. O objeto chave é utilizado como identificador do objeto valor, e assim não é permitida a utilização de chaves iguais para dois ou mais valores. O contêiner map é, internamente, uma *RB–Tree (Red–Black Tree)*. As *Red–Black Trees* são árvores binárias balanceadas de forma a garantir que operações como a busca, remoção e inserção de nós sejam feitas, no pior caso, em tempo $log(N)$, onde N é o número de nós que compõem a árvore.

Em cada nó de uma *RB–Tree* armazenamos uma cor: vermelho (*red*) ou preto (*black*). De acordo com as regras impostas na coloração dos nós, *RB–Trees* garantem que nenhum caminho, de uma mesma raiz até uma folha, é mais de duas vezes maior que qualquer outro.

Ainda, os nós de uma *RB–Tree* contém as informações sobre a chave, o filho à esquerda, o filho à direita e o pai. Se um filho ou o pai de um nó não existir, o ponteiro do campo correspondente aponta para *null*. Supondo que cada informação armazenada em um nó da árvore ocupe 4 bytes de memória, seu custo total é de $20\mathsf{N}$ bytes, onde N é o número de nós que compõem a árvore.

Uma árvore binária é uma *RB–Tree* se satisfaz as seguintes propriedades:

1. Todo nó é vermelho ou preto.

2. Toda folha(*null*) é preta.

3. Se um nó é vermelho, então seus dois filhos são pretos.

4. Todo caminho simples de um dado nó até uma folha contém o mesmo número de nós preto.

A figura 2.20 mostra o exemplo de uma *RB–Tree*. Repare que cada nó da árvore é vermelho ou preto, toda folha (*null*) é preta, os filhos de um nó vermelho são pretos e todo caminho simples de um nó a uma folha contém o mesmo número de nós pretos.

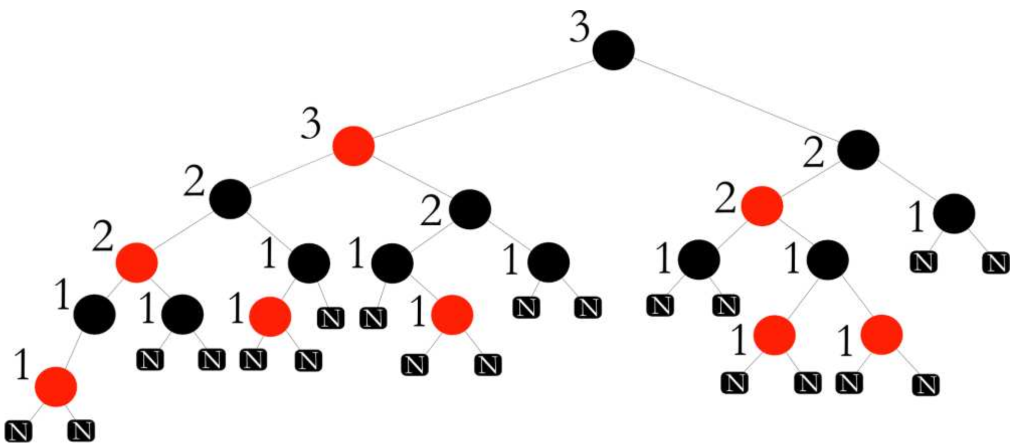

Figura 2.20: Exemplo de uma *RB–Tree*.

Podemos encontrar um estudo mais detalhado sobre *RB–Trees* em (Cormen *et al.* 1990).