DCC-IME-USP Tópicos de Análise de Algoritmos
Professor Fernando Mário de Oliveira Filho

Skip Lists

Gláucio Alves de Oliveira
Thales A. B. Paiva
{glaucioaorj, thalespaiva}@gmail.com

24 de abril de 2016

Resumimos os principais resultados sobre Skip Lists aleatorizadas e determinísticas. Skip Lists foram introduzidas por W. Pugh como alternativa às árvores balanceadas. Seu uso é justificado pelos algoritmos mais eficientes e de fácil implementação.

Sumário

1 *Skip Lists*

Introduzidas por Pugh em [?], *skip lists* são uma extensão natural de listas ligadas ordenadas. Ambas podem ser usadas para manter um conjunto ordenado de chaves, e eventualmente seus valores associados. Mostramos nas próximas seções que operações de busca, inserção, e remoção em *skip lists* são feitas em $\mathcal{O}(\log n)$, como para árvores balanceadas de busca. Isso faz com que a escolha entre usar *skip lists* ou árvores balanceadas numa determinada aplicação seja baseada nas dificuldades de implementação, no uso de memória, e no tamanho das constantes multiplicativas dos custos de operação.

Uma *skip list* é determinada por um conjunto de nós, que têm chave e dados associados, e apontadores de nós de chave menor a nós de chave maior. Enquanto nas listas ligadas, um nó aponta a um só outro nó, em *skip lists*, um nó pode apontar a vários outros nós de chave maior. O número máximo de nós a que um nó pode apontar é definido aleatoriamente quando este é inserido na lista, e é chamado de nível do nó. Na definição do nível um nó, usa-se um algoritmo que garante que, em média, a proporção de nós com ao menos $i + 1$ níveis em relação àqueles com ao menos i níveis, seja de p, uma probabilidade fixada na criação da *skip list*.

A interface de operações básicas permitidas sobre uma lista ligada é a mesma que a interface para árvores de busca, com a adição da probabilidade p passada em sua inicialização:

- INIT(p): Cria uma *skip list* com parâmetro p vazia e a devolve.
- INSERT(S, k, d): Insere um nó de chave k, e conjunto de dados d, em S.
- REMOVE(S, k): Remove o nó de chave k de S.
- SEARCH(S, k): Busca o nó de chave k em S e o devolve.

Dentre essas operações, a mais importante é a busca. Isso pois, uma vez que o algoritmo de busca é entendido, as outras funções são facilmente descritas.

A Figura 1.1 mostra um exemplo de *skip list*. O nó de chave $-\infty$ indica o começo da lista, e o nó de chave $+\infty$ indica o final. Note também que cada nível de 0 a 3, representado por cada $S_0, \ldots, S_3$, forma uma lista ligada ordenada.

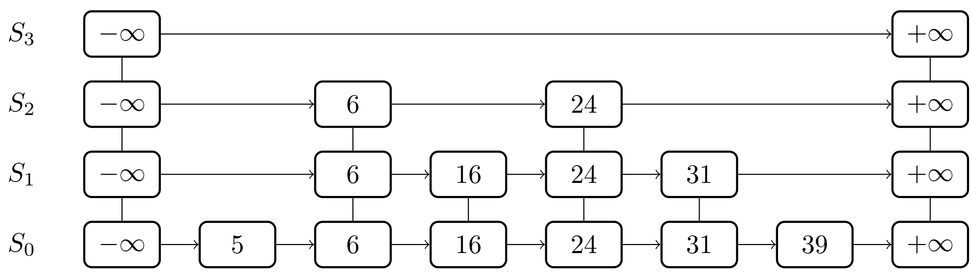

Figura 1.1: Exemplo de skip list.

Estamos prontos para dar uma definição mais precisa de *skip lists*, que nos permite provar alguns fatos interessantes. A definição apresentada não é diretamente implementável em computadores, pois usa conjuntos infinitos. Mostraremos como implementar a definição abaixo na Seção 2.

Definição 1.1. *Uma* **Skip List** *de elementos* $c_1, c_2, \ldots, c_n$*, em ordem crescente, de um conjunto totalmente ordenado, é uma dupla* $\langle S, p \rangle$ *tal que:*

- $S = \{S_0, S_1, S_2, \ldots\}$.
- $S_0 = \{c_1, c_2, \ldots, c_n\}$.
- *Cada* S_i*, para* $i \geq 1$*, é construído de forma que:*
  1. $\Pr(c \in S_i | c \in S_{i-1}) = p$.
  2. $\Pr(c \in S_i | c \notin S_{i-1}) = 0$.

Definição 1.2. *O nível de um elemento* c *de uma skip list é o número de conjuntos de* S *a que* c *pertence.*

2 Estruturas e Algoritmos

2.1 Estruturas de Dados

Como uma *skip list* é composta por nós, primeiro vamos definir a estrutura `Node`. Um nó `node` tem os seguintes atributos:

- `node.key` : é a chave do nó, que é única para cada nó de uma *skip list*.
- `node.level` : é o número de níveis atribuído ao nó em sua criação.
- `node.next[i]` : é o nó para que `node` aponta em seu `i`-ésimo nível.
- `node.data` : os dados associadas ao nó, que são irrelevantes neste trabalho.

E agora podemos definir a estrutura `SkipList`. Uma instância `skiplist` que representa a *skip list* $\langle S, p \rangle$ tem os seguintes atributos:

- `skiplist.p` : é a probabilidade p.
- `skiplist.level` : é o nível do maior nó, e pode variar a cada inserção.
- `skiplist.next[i]` : é o primeiro nó do `i`-ésimo nível da *skip list*.

2.2 Algoritmos

3 Análise Assintótica do Custo Médio

Definição 3.1. *O* ***custo de busca*** *de um elemento de uma skip list é o número de comparações feitas durante a busca por ele.*

Definição 3.2. *Um* ***caminho*** *induzido por uma busca é uma sequência* $(o_1, o_2, \ldots, o_m)$ *em que cada passo* o_i *pertence a* $\{\rightarrow, \downarrow\}$*, tal que:*

- *No início da busca, a sequência é vazia.*
- $\rightarrow$ *é adicionado à sequência a cada novo nó visitado na busca.*
- $\downarrow$ *é adicionado à sequência a cada descida de nível na busca.*

Lema 3.1. *O custo de busca de um elemento é igual ao tamanho do caminho de busca por esse elemento.*

Demonstração. Cada $\rightarrow$ é adicionado pela chamada `x = x.next[i]`, que ocorre quando a condição `x.next[i].key < searched_key` é verificada. Cada $\downarrow$ é adicionado quando ocorre quando a condição `x.next[i].key < searched_key` não é verificada.

Logo, cada comparação adiciona um passo ao caminho de busca, e não há outro modo de adicionar um passo ao caminho de busca. □

Queremos mostrar que o custo médio de busca em *skip lists* é $\mathcal{O}(\log n)$. Pelo lema 3.1, podemos simplificar a demonstração dividindo a busca nas componentes vertical e horizontal.

Lema 3.2. *O número esperado de passos* $\downarrow$ *em qualquer caminho de busca é* $\mathcal{O}(\log n)$.

Demonstração. Note que toda busca desce até o nível 1 da *skip list*, então o número esperado de passos é o número de níveis, ou altura, da *skip list*. Seja H a variável aleatória que representa a altura de uma *skip list* L. E sejam $H_1, H_2, \ldots, H_n$ as variáveis aleatórias que representam as alturas de $c_1, c_2, \ldots, c_n$, respectivamente.

Como $H = \max\{H_i : i = 1, \ldots, n\}$, é claro que

$$\Pr(H \geq k) \leq \sum_{i=1}^{n} \Pr(H_i \geq k).$$

As H_i seguem a mesma distribuição e essas variáveis são tais que $\Pr(H_i \geq k) = p^k$. Então

$$\Pr(H \geq k) \leq np^k.$$

Como a H é discreta e toma valores positivos

$$\begin{aligned}
\mathbb{E}(H) &= \sum_{k=0}^{\infty} \Pr(H \geq k) \\
&= \sum_{k=0}^{\lceil 2\log_{1/p} n\rceil - 1} \Pr(H \geq k) + \sum_{k=\lceil 2\log_{1/p} n\rceil}^{\infty} \Pr(H \geq k).
\end{aligned}$$

Vamos analisar cada parcela calculada. Para a primeira, temos

$$\begin{aligned}\sum_{k=0}^{\lceil 2\log_{1/p} n\rceil - 1} \Pr(H \geq k) &\leq \sum_{k=0}^{\lceil 2\log_{1/p} n\rceil - 1} 1 \\ &\leq \lceil 2\log_{1/p} n\rceil.\end{aligned}$$

E para a segunda parcela, temos

$$\begin{aligned}\sum_{k=\lceil 2\log_{1/p} n\rceil}^{\infty} \Pr(H \geq k) &\leq \sum_{k=\lceil 2\log_{1/p} n\rceil}^{\infty} np^k \\ &= n \sum_{k=\lceil 2\log_{1/p} n\rceil}^{\infty} p^k \\ &= np^{\lceil 2\log_{1/p} n\rceil} \sum_{k=0}^{\infty} p^k \\ &= np^{\lceil 2\log_{1/p} n\rceil} \frac{1}{1-p} \\ &= n \left(\frac{1}{p}\right)^{-\lceil 2\log_{1/p} n\rceil} \frac{1}{1-p} \\ &= n \left(\frac{1}{p}\right)^{-\lceil 2\log_{1/p} n\rceil} \frac{1}{1-p} \\ &\leq nn^{-2}\frac{1}{1-p} \\ &= \frac{1}{n(1-p)}.\end{aligned}$$

Somando os resultados

$$\mathbb{E}(H) \leq \lceil 2\log_{1/p} n\rceil + \frac{1}{n(1-p)}.$$

Portanto

$$\mathbb{E}(H) = \mathcal{O}(\log n).$$

□

Lema 3.3. *O número esperado de passos* $\rightarrow$ *em qualquer caminho de busca é* $\mathcal{O}(\log n)$.

Demonstração. Considere que estamos percorrendo ao contrário um caminho de busca qualquer e nos encontramos num nível qualquer de um nó qualquer. Temos duas opções:

1. Se possível, subir mais um nível.
2. Se não, ir para a esquerda.

Note que, se for possível subir, o caminho invertido deve subir, caso contrário, não estamos percorrendo um caminho válido. Isso pois qualquer caminho válido encontra sempre o nível mais alto do nó buscado.

A probabilidade de ser possível subir mais um nível neste nó é justamente a probabilidade de haver ao menos $i+1$ níveis no nó dado que há ao menos i níveis, ou seja, p. Logo a probabilidade de termos de andar à esquerda é $1-p$.

Seja R_i a variável aleatória que conta o número de passos $\rightarrow$ feitos no nível i, num caminho de busca. Ou seja, R_i conta o número de observações **antes** de um evento com probabilidade p ser observado. Essa interpretação mostra que $R_i \sim \mathrm{Geom}(p)$.

A esperança de R_i é:

$$\mathbb{E}(R_i) = \sum_{k=1}^{\infty} k \Pr(R_i = k) = \sum_{k=1}^{\infty} k(1-p)^k p$$

Fazendo $q = 1-p$, temos

$$\begin{aligned}
\mathbb{E}(R_i) &= \sum_{k=1}^{\infty} kq^k(1-q) = \sum_{k=1}^{\infty} \left(kq^k - kq^{k+1}\right) \\
&= \sum_{k=0}^{\infty} \left((k+1)q^{k+1} - kq^{k+1}\right) \\
&= \sum_{k=0}^{\infty} q^{k+1} = \sum_{k=1}^{\infty} q^k = \frac{q}{1-q} \\
&= \frac{1-p}{p}.
\end{aligned}$$

E assim, para todo nível i, $\mathbb{E}(R_i) = \mathcal{O}(1)$.

Seja R a variável aleatória que conta o número de passos $\rightarrow$ no caminho, e seja H a variável aleatória que representa a altura da *skip list*. Então:

$$R = \sum_{i}^{H} R_i.$$

E temos que

$$\mathbb{E}(R) = \mathbb{E}\left(\sum_{i=1}^{H} R_i\right) = \sum_{i=1}^{\mathbb{E}(H)} \mathbb{E}(R_i) = \mathbb{E}(H)\mathbb{E}(R_i) = \mathcal{O}(\log n)\mathcal{O}(1).$$

Portanto, $\mathbb{E}(R) = \mathcal{O}(\log n)$.

□

Teorema 3.4. *O custo médio de busca em skip lists é* $\mathcal{O}(\log n)$.

Demonstração. É corolário dos lemas 3.2 e 3.3. Como num caminho o número de passos num caminho é a soma do número esperado de passos $\downarrow$ com o de passos $\rightarrow$. Como ambos são $\mathcal{O}(\log n)$, a soma também é $\mathcal{O}(\log n)$. □