Курс kiev-clrs – Лекция 12. Списки с пропусками

Олег Смирнов

20 июня 2009 г.

Содержание

1 Цель лекции

- Рассмотреть списки с пропусками
- Рассмотреть анализ “с высокой вероятностью”

2 Связные списки и списки с пропусками

Список с пропусками является структурой данных, аналогичной сбалансированным деревьям:

- RB-деревья
- декартовы деревья (treaps)
- B-деревья

Операций поиска, вставки и удаления работают в списке с пропусками за время $\lg(n)$ для n элементов *с большой вероятностью*.

Списки с пропусками были предложены в 1990 году [Pug90]. Отличительной особенностью этой структуры является простота имплементации.

Сортированный связный список является простейшей структурой со временем поиска $\Theta(n)$. Работу такой структуры можно улучшать разными способами. Одной из идей является добавление еще одного уровня (связного списка), обеспечивающего быстрый доступ через несколько элементов. Хорошим примером такой структуры является метро с

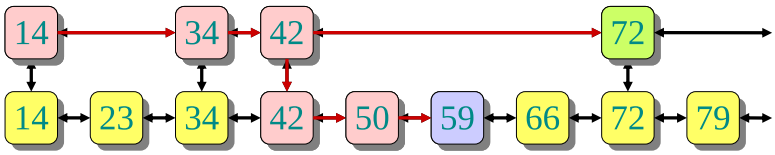

Рис. 1: Связные списки

экспресс-линией между несколькими удаленными станциями и местной линией, которая останавливается на каждой станции.

Итак, на нижнем уровне L_2 присутствуют все элементы, на верхнем L_1 – только некоторое количество. Между одинаковыми элементами уровней существуют ссылки. Это инвариант структуры.

В таком случае алгоритм поиска $Search(x)$ будет состоять из шагов:

1. Начало поиска в левом верхнем углу (на экспресс-линии)
2. Передвигаться вправо по верхнему списку L_1 пока это возможно ($x <$ ключа)
3. Переместиться в нижний список и передвигаться до искомого элемента x

2.1 Анализ поиска

Скорость работы поиска будет зависить от количества элементов в верхнем списке $|L_1|$. Для анализа преположим, что в L_1 попало несколько элементов в случайном порядке.

Тогда время поиска в худшем случае равно:

$$\approx \underbrace{|L_1|}_{\text{в } L_1} + \underbrace{\frac{|L_2|}{|L_1|}}_{\text{в } L_2} =$$

$$= |L_1| + \frac{n}{|L_1|}$$

(минимизируя)

$$|L_1|^2 = |L_2| = n \Rightarrow |L_1| = \sqrt{n}$$

Итак:

$$|L_1| + \frac{|L_2|}{|L_1|} = \sqrt{n} + \frac{n}{\sqrt{n}} = 2\sqrt{n}$$

Полученная структура выглядит сбалансированной: на каждом уровне будут просмотрены цепочки одинаковой длины.

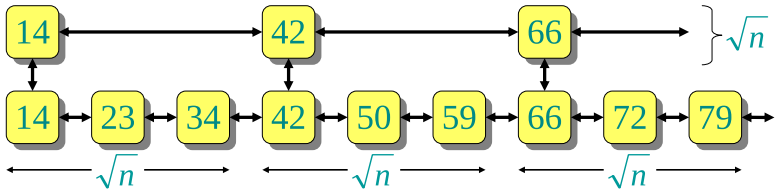

Рис. 2: Связные списки

Можно показать, что время поиска можно улучшать, добавляя новые экспресс-уровни:

- 2 отсортированных списка $\Rightarrow 2\sqrt{n}$
- 3 отсортированных списка $\Rightarrow 3\sqrt[3]{n}$
- k отсортированных списка $\Rightarrow k\sqrt[k]{n}$
- $\lg n$ отсортированных списка $\Rightarrow \lg n \sqrt[\lg n]{n} = \lg n \cdot n^{\frac{1}{\lg n}} = 2\lg n$

$\lg n$ сортированных связных списков ведут себя похоже на бинарное дерево. В идеальном списке с пропусками отношение количества элементов между уровнями постоянно и равно 2. В этом случае в нижнем списке будет $2^{\lg n} = n$ элементов. Поиск в этой структуре будет выпол-

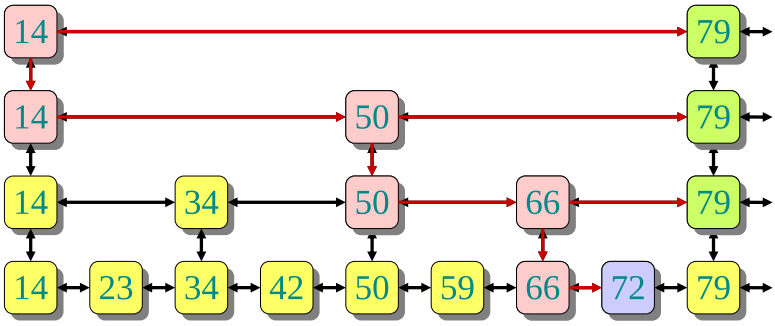

Рис. 3: Идеальный список с пропусками

нятся за $O(\lg n)$ шагов.

3 Алгоритм поиска

В практической реализации обычно добавляют специальные элементы-ограничители в начале каждого уровня. Их значение принимают равным $-\infty$.

Алгоритм на псевдокоде:

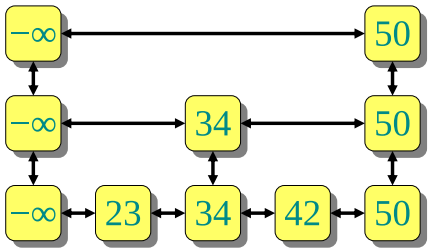

Рис. 4: Список с ограничителями

SEARCH(L, x)

```
1  v ← L
2  while v ≠ NULL and key(v) ≠ x
3        do
4           if key(right(v)) > x
5              then v ← down(v)
6              else  v ← right(v)
7  return v
```

4 Алгоритмы модификации

Необходимо разработать алгоритмы вставки и удаления элемента, которые бы сохраняли свойства списка с пропусками и работали за логарифмическое время. Алгоритм вставки $Insert(x)$ включает следующие шаги:

- найти место для вставки в самом нижнем списке $Search(x)$
- вставить элемент в нижний список (поддержка инварианта структуры)
- “бросить монетку” и в зависимости от результата (например, если выпала “решка”) добавить элемент уровнем выше (вероятность $1/2$)
- повторять предыдущий шаг, уменьшая вероятность с каждым уровнем (бросая монетку при каждой вставке)

Таким образом, на нижний уровень элемент будет вставлен всегда, на уровень выше – с вероятностью $1/2$, выше – $1/4$ и т.д. Элемент попадет на уровень $\lg n$ с вероятностью $1/n$.

Алгоритм удаления элемента $Delete(x)$ тривиален: элемент удаляется из всех списков, которые его содержат.

5 Анализ "с высокой вероятностью"

Теорема: каждый поиск в списке с пропусками из n элементов *с высокой вероятностью* выполняется за $O(\lg n)$.

Событие E случается *с высокой вероятностью*, если существует $\alpha \geqslant 1$, такое, что для выбранных соответствующим образом констант событие E случается с вероятностью $\geqslant 1 - O(\frac{1}{n^\alpha})$. Константы в определениях связаны, т.е. если необходимо, чтоб условие поиска выполнялось с вероятностью $1 - O(\frac{1}{n^{100}})$, то сам поиск будет выполнятся за время, предположим $O(100 \lg n)$, которое всё равно является логарифмическим.

В доказательстве используется неравеноство Буля:

$$Pr\{E_1 \cup E_2 \cup \ldots \cup E_k\} \leqslant Pr\{E_1\} + Pr\{E_2\} + \ldots + Pr\{E_k\}$$

Лемма: с высокой вероятностью кол-во уровней списка с пропусками равно $O(\lg n)$.

Вероятность обратного события $Pr\{\# > c \lg n\}$ должна быть полиноминально малой:

$$\begin{aligned} Pr\{\# > c \lg n\} \leqslant \\ \text{(суммируем по всем элементам} \\ \text{списка по неравенству Буля)} \\ \leqslant n Pr\{X \text{ выше } c \lg n\} = \\ = n\left(\frac{1}{2}\right)^{c \lg n} = \frac{n}{n^c} = \\ = \frac{1}{n^{c-1}} = \frac{1}{n^\alpha} \\ \text{выбирая } \alpha = c - 1 \end{aligned}$$

Информации о высоте уровней недостаточно. Для доказательства теоремы необходимо еще знать длину цепочек между элементами, соединяющими уровни списка.

Идея доказательства: рассмотреть "обратный" поиск от искомого элемента в левый верхний угол списка

Этот процесс будет выполнять в точности такое же кол-во шагов, что и прямой поиск:

- поиск начинается с элемента в нижнем списке
- каждое перемещение между элементами происходит влево или вверх по списку
- переход влево происходит, когда у текущего элемента нет соседа сверху (при построении выпал "орел")
- переход вверх происходит, когда сосед есть (выпала "решка")
- поиск останавливается в корне или в $-\infty$

Для доказательства общего случая достаточно доказать, что каждый поиск выполнятся за логарифмическое время с высокой вероятностью. Тогда общий случай будет получен из неравенства Буля.

1. Поиск остановится, когда будет достигнут корень структуры
2. Количесво переходов вверх в поиске $<$ кол-ва уровней $\leqslant c \lg n$ с высокой вероятностью (по лемме)
3. Переход вверх выполнятся с вероятностью $1/2$ (по построению списка)
4. Следовательно, общее количество переходов $\leqslant$ вероятности построения уровня высотой $c \lg n$ (количеству подбрасываний монетки для получения $c \lg n$ "решек")

Т.о. необходимо доказать, что количество подбрасываний монетки, необходимое для получения $c \lg n$ "решек" равно $\Theta(\lg n)$ с высокой вероятностью.

Можно доказать для частного случая границу $O(\lg n)$: рассмотрим

$10c \lg n$ подбрасываний монетки. Когда выпадет ровно $c \lg n$ решек?

$$Pr\{\#c \lg n \text{ решек}\} = \underbrace{\binom{10c \lg n}{c \lg n}}_{\text{перестановки}} \cdot \underbrace{\left(\frac{1}{2}\right)^{c \lg n}}_{\text{решки}} \cdot \underbrace{\left(\frac{1}{2}\right)^{9c \lg n}}_{\text{орлы}}$$

$$\begin{aligned} Pr\{\#c \lg n \text{ решек}\} &\leqslant \binom{10c \lg n}{c \lg n} \cdot \left(\frac{1}{2}\right)^{9c \lg n} \leqslant \\ &\leqslant \left(e \frac{10c \lg n}{c \lg n}\right)^{c \lg n} \cdot \left(\frac{1}{2}\right)^{9c \lg n} = \\ &= (10e)^{c \lg n} 2^{-9c \lg n} = \\ &= 2^{\lg (10e) \cdot c \lg n} 2^{-9c \lg n} = \\ &= 2^{(\lg (10e) - 9) \cdot c \lg n} = 1/n^{\alpha} \\ &\text{для } \alpha = (9 - \lg (10e)) \cdot c \end{aligned}$$

Таким образом $Pr\{\text{не более } c \lg n \text{ решек}\} \leqslant 1/n^{\alpha}$, для $\alpha = (9 - \lg (10e)) \cdot c$. Здесь $\alpha \to \infty$ при $10 \to \infty$ для любого c.

Таким образом константа, скрытая внутри границы $O(\lg n)$ удовлетворяет требование для α.

6 Заключительные замечания

- с теоретической точки зрения список с пропусками ничем не лучше сбалансированного бинарного дерева. С практической точки зрения эта структура намного проще в реализации
- списки с пропусками широко используются в пиринговых сетях для реализации распределенных хэш-таблиц [SMK+01] [Kha05] [HAJ+03]

Список литературы

[HAJ+03] Nicholas J.A. Harvey, Nicholas J. A, Harvey John, John Dunagan, Michael B. Jones, Stefan Saroiu, Marvin Theimer, and Alec Wolman. Skipnet: A scalable overlay network with practical locality properties, 2003.

[Kha05] Qasem Kharma. *Enhanced skip-list search algorithm in 3-layer mediator framework*. PhD thesis, Miami, FL, USA, 2005. Major Professor-Ege, Raimund K.

[Pug90] William Pugh. Skip lists: A probabilistic alternative to balanced trees, 1990.

[SMK+01] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. pages 149–160, 2001.