A Tour of Understanding Linear Probing Hashing

Zhenghua Lyu
zlyu@vmware.com

Greenplum Developer @ VMware

June 8, 2022

> The author cannot resist inserting a biographical note at this point: I first formulated the following derivation in 1962, shortly after beginning work on *The Art of Computer Programming*. Since this was the first nontrivial algorithm I had ever analyzed satisfactorily, it had a strong influence on the structure of these books. Ever since that day, the analysis of algorithms has in fact been one of the major themes of my life.
>
> Donald Knuth

Abstract

The analysis of Linear Probing Hashing algorithm plays a critical role in many interesting problems of computer science. Date back to 1960s, Knuth firstly analyzed this [3] and then Flajolet analyzed this using the tool of analytical combinatorics in late 1990s [2]. This article follows Flajolet's paper step by step with some mathematical tool warming-ups.

Contents

1 Introduction **2**

2 Mathematical Warm-ups **2**

2.1 Bivariate Generating Functions 2
2.2 Case Study: not-move elements in a random shuffle 3
2.3 Tree Function 4
2.4 Ramanujan $\mathcal{Q}$-function 5
2.5 More on Tree function 7

3 Linear Probing Hashing **8**
3.1 The Algorithm 8
3.2 Total displacement 8

4 Full table and almost full table **8**
4.1 Simple enumerating 8
4.2 Generating Functions 9
4.3 Moments analysis 10

5 Sparse Table **13**
5.1 Simple enumerating 13
5.2 Generating Function 14
5.3 Moments analysis 14

6 The end and start of the tour **16**

1 Introduction

At the end of May 2020, I gave an internal techtalk on Hashing Tables in Greenplum (Postgres). During the preparation of the topic, I found Flajolet and Knuth's work on the analysis of Linear Probing Hashing [2, 3]. Flajolet's paper looks too complex at the first glance, however Knuth described the method as *surprisingly simple solution*. I wrote this article to show the steps of learning **the simple solution**.

2 Mathematical Warm-ups

2.1 Bivariate Generating Functions

Generating Function is a powerful tool to count the combinatorial structure (Class). Sometimes we are interested in the parameters of the Class, e.g. the question how many leaves in a random tree. To answer such kind of question, we can model the parameter as a cost function on the object in this Class and thus get a bivariate generating function. $A_{n,k}$ means the number of elements with size n and cost k.

- Given a Class $\mathcal{A}$,
- with two functions size and cost are both the type of $\mathcal{A} \mapsto \mathbb{N}$

We define the OBGF for unlabelled Class and EBGF for labelled Class as:

$$\begin{aligned} \textbf{BGF}: A(z,u) &= \sum_{a\in\mathcal{A}} z^{|a|}u^{\text{cost}(a)} = \sum_{n,k\geq 0} A_{n,k} z^n u^k \\ \textbf{EBGF}: A(z,u) &= \sum_{a\in\mathcal{A}} \frac{1}{|a|!} z^{|a|}u^{\text{cost}(a)} = \sum_{n,k\geq 0} A_{n,k} \frac{1}{n!} z^n u^k \end{aligned} \tag{1}$$

The following formulas show how easy you can use **BGF** to compute the parameter's moment:

- cumulative generating function: $\frac{\partial A(z,u)}{\partial u}\Big|_{u=1} = \sum_n \big(\sum_{k\geq 0} kA_{n,k}\big) z^n$
- the total number of size n elements: $[z^n]A(z,1)$
- average cost of size n elements: $\frac{[z^n]\frac{\partial A(z,u)}{\partial u}\Big|_{u=1}}{[z^n]A(z,1)}$

We will use the following operators and notations on BGFs in later sections' analysis:

$$\begin{aligned}
\mathsf{U}A(z,u) &= A(z,1)\\
\mathsf{Z}A(z,u) &= zA(z,u)\\
\mathsf{H}A(z,u) &= \frac{A(z,u) - uA(uz,u)}{1-u}\\
\partial_u A(z,u) &= \frac{\partial A(z,u)}{\partial u}\\
\partial_z A(z,u) &= \frac{\partial A(z,u)}{\partial z}
\end{aligned}$$

The H operator on EBGF $A(z,u) = \sum_{n,k} A_{n,k}\frac{1}{n!}z^n u^k = \sum_n A_n(u)\frac{z^n}{n!}$ has following interesting property:

$$\mathsf{H}A(z,u) = \sum_n A_n(u)(1+u+u^2+\cdots+u^n)\frac{z^n}{n!}. \tag{2}$$

2.2 Case Study: not-move elements in a random shuffle

Suppose we have n different cards, after random shuffling, some cards might still at the original position. The question is what is the average number of cards that do not change position after a random shuffling. If we model the number of not-move cards as the cost function of a permutation, then we can simply apply the average cost formula in the previous section.

Permutations are equivalent to Set of Cycles [4], those cards that do not change position are those in a cycle of length 1. We are interested in the parameter of the number of cycle with length 1 and this leads directly to the symbolic method to build the specification (we use EBGF here since the cards are different):

$$P = \mathrm{SET}(u\mathrm{CYC}_1(z) + (\mathrm{CYC}(z) - \mathrm{CYC}_1(z))) \implies P(z,u) = e^{uz + \ln\frac{1}{1-z} - z} \tag{3}$$

Then it is easy to compute the average cost (average number of cards that not move):

$$\begin{aligned}
&\#\text{ permutations for } n \text{ cards} : n![z^n]P(z,1) = n![z^n]\frac{1}{1-z} = n!\\
&\text{accumulated cost for } n \text{ cards} : n![z^n]\frac{\partial A(z,u)}{\partial u}\Big|_{u=1} = n![z^n]\frac{z}{1-z} = n!\\
&\text{average cost (not-move) of } n \text{ cards} : 1
\end{aligned} \tag{4}$$

2.3 Tree Function

Consider a class of rooted unordered labelled trees. Such kind of trees are called Cayley trees. It is enough to enumerate all Cayley trees of size 3 to capture the concept of its name (see Figure 1).

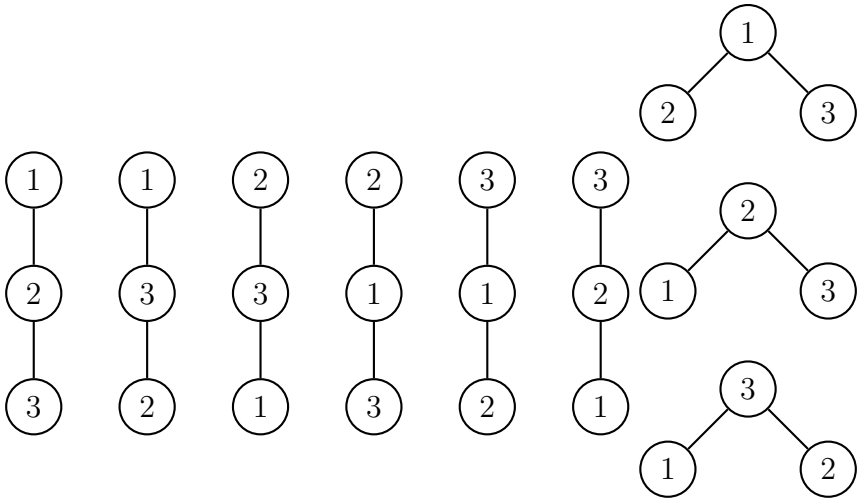

Figure 1: All Cayley trees of size 3

The specification of the combinatorial structure of Cayley trees are: a root node with a set of Cayley trees. With symbolic method, we can get the corresponding EGF:

$$T = \bigcirc \circ \mathrm{SET}(T) \implies T(z) = ze^{T(z)} \tag{5}$$

We call $T(z)$ the tree function. To get the coefficient of the tree function we need use **Lagrange inversion**:

Theorem 1 (**Lagrange inversion theorem**). *Generating function* $A(z) = \sum_{n\geq 0} a_n z^n$ *satisfies the functional equation* $z = f(A(z))$*, where* $f(0) = 0, f'(0) \neq 0$*, then we can have the following three formulas:*

$$a_n = [z^n]A(z) = \frac{1}{n}[u^{n-1}]\big(\frac{u}{f(u)}\big)^n$$
$$[z^n](A(z))^m = \frac{m}{n}[u^{n-m}]\big(\frac{u}{f(u)}\big)^n$$
$$[z^n]g(A(z)) = \frac{1}{n}[u^{n-1}]g'(u)\big(\frac{u}{f(u)}\big)^n.$$

For the tree function, $T(z) = ze^{T(z)} \implies z = T(z)e^{-T(z)}$, then we know here $f(x) = xe^{-x}, f(0) = 0, f'(0) = 1 \neq 0$, so we can apply theorem 1:

$$[z^n]T(z) = \frac{1}{n}[u^{n-1}]\big(\frac{u}{ue^{-u}}\big)^n = \frac{1}{n}[u^{n-1}]e^{nu} = \frac{1}{n}[u^{n-1}]\sum_{k\geq 0}\frac{1}{k!}(nu)^k = \frac{1}{n!}n^{n-1}$$

$$\implies T_n = n![z^n]T(z) = n^{n-1} \implies T(z) = \sum_{n\geq 1}\frac{n^{n-1}}{n!}z^n$$

We can find more generating functions of $T(z)^m$ and $\frac{1}{1-T(z)}$:

$$[z^n]T(z)^m = \frac{m}{n}[u^{n-m}]\sum_{k\geq 0}\frac{1}{k!}(nu)^k = m\frac{n^{n-m-1}}{(n-m)!}$$
$$\Longrightarrow T(z)^m = \sum_{n\geq m} m\frac{n^{n-m-1}}{(n-m)!n!}n! = m\sum_{n\geq m}\frac{n^{n-m-1}}{n!}n^{\underline{m}}z^n$$

And

$$\frac{1}{1-T(z)} = 1 + T(z) + T(z)^2 + \cdots + T(z)^m + \ldots$$
$$\Longrightarrow n \geq 1, [z^n]\frac{1}{1-T(z)} = \sum_{m=1}^{n}[z^n]T(z)^m$$
$$\Longrightarrow n \geq 1, [z^n]\frac{1}{1-T(z)} = \sum_{m=1}^{n} m\frac{n^{n-m-1}}{n!}n^{\underline{m}} = \frac{1}{n!}\sum_{m=1}^{n} mn^{n-m-1}n^{\underline{m}}.$$

With the following formula from [1]:

$$n^n = \sum_{k=1}^{n} kn^{n-k-1}n^{\underline{k}},$$

we know

$$\begin{aligned} n \geq 1, [z^n]\frac{1}{1-T(z)} &= \frac{n^n}{n!} \\ \Longrightarrow \frac{1}{1-T(z)} &= 1 + \sum_{n\geq 1}\frac{n^n}{n!}z^n. \end{aligned} \tag{6}$$

2.4 Ramanujan $\mathcal{Q}$-function

$\mathcal{Q}$ function is defined as:

$$Q(n) = 1 + \frac{n-1}{n} + \frac{(n-1)(n-2)}{n^2} + \cdots + = \sum_{i\geq 0}\frac{(n-1)^{\underline{i}}}{n^i},$$

use Laplace method [4] we can get its asymptotic representation:

$$Q(n) \sim \sqrt{\frac{\pi n}{2}}$$

A very interesting application for the above formula is to answer the question: *what is the average number of the people you need to ask the first time you find the one whose birthday's date is the same as yourself?* This can be solved by:

$$\text{average waiting time } = \sum_{k\geq 1} Pr\{\text{k people no same birthday}\}$$

$\mathcal{Q}$ function can be related to the tree function, note:

$$\begin{aligned}
&\ln\frac{1}{1-T(z)} = 1 + T(z) + \frac{1}{2}T(z)^2 + \frac{1}{3}T(z)^3 + \cdots + \frac{1}{m}T(z)^m + \ldots \\
\Longrightarrow\ & n > 0, [z^n]\ln\frac{1}{1-T(z)} = \sum_{m=1}^{n}\frac{1}{m}[z^n]T(z)^m = \sum_{m=1}^{n}\frac{1}{n!}n^{n-m-1}n^{\underline{m}} \\
\Longrightarrow\ & n > 0, [z^n]\ln\frac{1}{1-T(z)} = \frac{n^{n-1}}{n!}\sum_{m=1}^{n}\frac{n^{\underline{m}}}{n^m} = \frac{n^{n-1}}{n!}\sum_{m=1}^{n} = \frac{n^{n-1}}{n!}Q(n) \\
\Longrightarrow\ & \ln\frac{1}{1-T(z)} = 1 + \sum_{n\geq 1}Q(n)\frac{n^{n-1}}{n!}z^n
\end{aligned} \tag{7}$$

Knuth extended $\mathcal{Q}$ function in [3] and gave the asymptotic analysis*:

$$\begin{aligned}
Q_r(m,n) &= \sum_{k\geq 0}\binom{r+k}{k}\frac{n^{\underline{k}}}{m^k} \\
Q_0(m,n) &= \sum_{k\geq 0}\frac{n^{\underline{k}}}{m^k} \\
Q_0(m,n) &= 1 + \sum_{k\geq 1}\frac{n^{\underline{k}}}{m^k} = 1 + \frac{n}{m}\sum_{k\geq 1}\frac{(n-1)^{\underline{k-1}}}{m^{k-1}} = 1 + \frac{n}{m}Q_0(m,n-1) \\
Q(n) &= Q_0(n,n-1) = \sum_{k\geq 0}\frac{(n-1)^{\underline{k}}}{n^k} \\
Q_0(m,\alpha m-1) &= \frac{1}{1-\alpha} - \frac{1}{(1-\alpha)^3}m^{-1} + \frac{2+\alpha}{(1-\alpha)^5}m^{-2} + O(m^{-3})
\end{aligned}$$

An important generating function involving Q_0 is:

$$\frac{e^{mz}}{1-z} = \sum_{n\geq 0}Q_0(m,n)m^n\frac{z^n}{n!} \tag{8}$$

The above formula is deducted by:

$$\begin{aligned}
\frac{e^{mz}}{1-z} &= \Big(\sum_{n\geq 0}z^n\Big)\Big(\sum_{n\geq 0}\frac{1}{n!}(mz)^n\Big) \\
&= \sum_{n\geq 0}\sum_{k\geq 0}z^k\frac{1}{(n-k)!}(mz)^{n-k} \\
&= \sum_{n\geq 0}\sum_{k\geq 0}\frac{n!}{m^k}\frac{1}{(n-k)!}m^n\frac{z^n}{n!} \\
&= \sum_{n\geq 0}\sum_{k\geq 0}\frac{n^{\underline{k}}}{m^k}m^n\frac{z^n}{n!} = \sum_{n\geq 0}Q_0(m,n)m^n\frac{z^n}{n!}
\end{aligned}$$

*Seems the original paper [2] make a mistake for the relationship between $Q_0(m,n)$ and $Q_0(m,n-1)$.

2.5 More on Tree function

Tree function has many interesting and useful properties that will help us in later section's asymptotic analysis. Let's review some of them here.

First equation is the relation ship between $T'(z)$ and $T(z)$:

$$\begin{aligned} T(z) = ze^{T(z)} &\implies T'(z) = e^{T(z)} + zT'(z)e^{T(z)} = \frac{1}{z}T(z) + T'(z)T(z) \\ \implies (1 - T(z))T'(z) = \frac{1}{z}T(z) &\implies zT'(z) = \frac{T(z)}{1 - T(z)} \end{aligned} \tag{9}$$

Next, based on equation 6, apply $\frac{d}{dz}$ both sides, we have:

$$\begin{aligned} \frac{T'(z)}{(1 - T(z))^2} &= \sum_{n\geq 1} \frac{n^n}{n!} n z^{n-1} \\ \implies \frac{zT'(z)}{(1 - T(z))^2} &= \sum_{n\geq 1} \frac{n^{n+1}}{n!} z^n \\ \implies \frac{T(z)}{(1 - T(z))^3} &= \sum_{n\geq 1} \frac{n^{n+1}}{n!} z^n \end{aligned} \tag{10}$$

With the above skill working on equation 7, we have:

$$\begin{aligned} \frac{T'(z)}{1 - T(z)} &= \sum_{n\geq 1} Q(n) \frac{n^{n-1}}{n!} n z^{n-1} \\ \implies \frac{zT'(z)}{1 - T(z)} &= \sum_{n\geq 1} Q(n) \frac{n^{n-1}}{n!} n z^n \\ \implies \frac{T(z)}{(1 - T(z))^2} &= \sum_{n\geq 1} Q(n) \frac{n^n}{n!} z^n \end{aligned} \tag{11}$$

Let's try to find out $\frac{1}{(1-T(z))^2}$'s generating function:

$$\begin{aligned} \frac{1 - T(z)}{(1 - T(z))^2} &= \frac{1}{(1 - T(z))^2} - \frac{T(z)}{(1 - T(z))^2} \\ \implies \frac{1}{(1 - T(z))^2} &= \frac{1}{1 - T(z)} + \frac{T(z)}{(1 - T(z))^2} \\ \implies [z^n] \frac{1}{(1 - T(z))^2} &= [z^n] \frac{1}{1 - T(z)} + [z^n] \frac{T(z)}{(1 - T(z))^2} \\ \implies [z^n] \frac{1}{(1 - T(z))^2} &= \frac{n^n}{n!} + Q(n) \frac{n^n}{n!} = (1 + Q(n)) \frac{n^n}{n!}, \text{ for } n \geq 1 \\ \implies \frac{1}{(1 - T(z))^2} &= 1 + \sum_{n\geq 1} (1 + Q(n)) \frac{n^n}{n!} z^n \end{aligned} \tag{12}$$

Same skill (*apply* $\frac{d}{dz}$ to both sides and then multiply z) can use any times thus we can have the generating functions for any form of $\frac{P(T(z))}{(1-T(z))^r}$ for any polynomial P with degree less than r.

3 Linear Probing Hashing

3.1 The Algorithm

Linear Probing Hashing is a very simple algorithm and is efficient when the load of the hash table is not large. A typical Java implementation can be found in LinearProbingHashST [5]. The algorithm is described below:

> *A Hash Table H contains m slots, H*$[1..m]$ *with a hash function h* : *Key* $\mapsto$ $[1, 2, \ldots, m]$. $n(n \leq m)$ elements enter the hash table one by one, and if collision happens when we insert x, just try next empty slot in the cyclic order: $h(x), h(x)+1, h(x)+2, \ldots, h(1), h(2), \ldots, h(x)-1$.

We call a hash table:

- full: when $n = m$,
- almost full: when $n + 1 = m$,
- sparse: when filling ratio $\alpha = \frac{n}{m}$ is bounded away from 1.

3.2 Total displacement

We use total displacement to measure the performance of the hash table. When element x is in the slot $s_x (1 \leq s_x \leq m)$, then this element's displacement is $(h(x) - s_x) \mod m$, this quantity shows how many probes before we find x. Total displacement is the sum of each element's displacement. The remaining of this article study of probability distribution of total displacement. The total number of possible hash configurations is m^n (each elememt has m choices), we talk about average total displacement under the assumption that each hash configuration is in equal probability. $d_{m,n}$ is the total displacement for a hash table with m slots and n elements.

4 Full table and almost full table

4.1 Simple enumerating

Without loss of generality, a full table can be thought as creating in two steps:

(1) creating an almost full table, with the last slot empty,

(2) insert the last element into the last slot.

A Hash Table with m slots and n elements $(m \geq n)$ always has $m - n$ empty slots. The probability of slot $[m]$ to be empty is $\frac{m-n}{m}$. Each element has m possible slots to enter thus totally possible ways to create the hash table is m^n, among these there are $\frac{m-n}{m} m^n = (m-n)m^{n-1}$ possible configurations that leaving final slot empty. Consider almost full $(n = m - 1)$, there are $(n+1)^{n-1} = m^{m-2}$ possible configurations. The last element can be hashed to any slot, so there is m choices, thus the number of full tables is $mm^{m-2} = m^{m-1} = n^{n-1}$ (now full $n = m$).

4.2 Generating Functions

The combinatorial class is the set of all hash configruations. We define the size to be the elements in the hash table and the cost to be total displacement, then we can define the bivariate generating function for almost full hash table (with the final slot empty):

$$F(z,u) = \sum_{n,k\geq 0} F_{n,k} u^k \frac{z^n}{n!} = \sum_{n\geq 0} (\sum_{k\geq 0} F_{n,k} u^k) \frac{z^n}{n!} = \sum_{n\geq 0} F_n(u) \frac{z^n}{n!},$$

$F_{n,k}$ is the number of almost full hash table configurations with n elements and total displacement equals k. $F_n(u)$ encodes the distribution information when the we have n elements. Next, let's try to find a recurrent representation of F, in other words, we are going to decompose the problem into the same pattern sub-problems.

Consider the previous status of the almost full table, it has $n-1$ elements and contains two empty slots, one the last slot and the other empty slot might be $1, 2, \ldots, n$. If the slot $k+1$ is empty, then we can think of the hash table as two parts:

- the first part: $H[1, \ldots, k+1]$, this is an almost full table with k elements and the last slot is empty,
- the second part: $H[k+2, k+3, \ldots, n+1]$, this is also an almost full table with $n-k-1$ elements and the last slot is empty.

Thanks to the last slot's emptiness, we do not need to consider wrap-around, this make life much easier. The element inserted in slot $k+1$, its orignal hash address might be at $1, 2, \ldots, k+1$, adding $k, k-1, \ldots, 0$ to the total displacement respectively. Then the $F_n(u)$'s decomposition form is (it is a star production of sub-structures with a factor of cost:

$$F_n(u) = \sum_{k=0}^{n-1} \binom{n-1}{k} F_k(u)(1 + u + u^2 + \cdots + u^k) F_{n-k-1}(u) \tag{13}$$

Let $G_k(u) = F_k(u)(1 + u + u^2 + \cdots + u^k)$, then

$$F_n(u) = \sum_{k=0}^{n-1} \binom{n-1}{k} G_k(u) F_{n-k-1}(u), \tag{14}$$

and with the formula 2, we have

$$\mathsf{H}F(z,u) = \sum_{n\geq 0} F_n(u)(1 + u + u^2 + \cdots + u^n) \frac{z^n}{n!} = \sum_{n\geq 0} G_n(u) \frac{z^n}{n!}, \tag{15}$$

combine 14 and 15, we have

$$\begin{aligned} \frac{\partial F(z,u)}{\partial z} &= \sum_{n\geq 1} F_n(u) \frac{z^{n-1}}{(n-1)!} = \sum_{n\geq 0} F_{n+1}(u) \frac{z^n}{n!} \\ &= \sum_{n\geq 0} (\sum_{k\geq 0} \binom{n}{k} G_k(u) F_{n-k}(u)) \frac{z^n}{n!} \\ &= \sum_{n\geq 0} (\sum_{k\geq 0} \frac{G_k(u)}{k!} z^k \frac{F_{n-k}(u)}{(n-k)!} z^{n-k}) \\ &= F(z,u)\mathsf{H}F(z,u). \end{aligned} \tag{16}$$

We encode the almost full hash table's total displacement information in the differential equation 16. And now we are ready to find the relationship between full table and the almost full table. Let $C(z,u) = \sum_{n,k} C_{n,k} u^k \frac{z^n}{n!} = \sum_{n\geq 0} \left(\sum_{k\geq 0} C_{n,k} u^k\right) \frac{z^n}{n!} = \sum_{n\geq 0} C_n(u) \frac{z^n}{n!}$ is the bivariate generating function of the full table. To get a full table with $n+1$ elements from an almost full table, the last element has exact one slot to enter and its displacement might be $0, 1, \ldots, n$, With the same skill of getting formula 16 we have:

$$\begin{aligned} & C_{n+1}(u) = F_n(u)(1 + u + u^2 + \cdots + u^n) \\ \Longrightarrow & \frac{\partial C(z,u)}{\partial z} = \sum_{n\geq 1} C_n(u) \frac{z^{n-1}}{(n-1)!} = \sum_{n\geq 0} C_{n+1}(u) \frac{z^n}{n!} \\ \Longrightarrow & \frac{\partial C(z,u)}{\partial z} = \sum_{n\geq 0} F_n(u)(1 + u + u^2 + \cdots + u^n) \frac{z^n}{n!} \end{aligned}$$

Thus we get the relationship between C and F:

$$\frac{\partial C(z,u)}{\partial z} = \mathsf{H} F(z,u) \tag{17}$$

Now let's put all important equations together and do more deduction:

$$\begin{aligned} & \frac{\partial F(z,u)}{\partial z} = F(z,u) \frac{\partial C(z,u)}{\partial z} \\ \Longrightarrow & \frac{\partial C(z,u)}{\partial z} = \frac{\frac{\partial F(z,u)}{\partial z}}{F(z,u)} \\ \Longrightarrow & C(z,u) = \log F(z,u), F(z,u) = e^{C(z,u)} \end{aligned} \tag{18}$$

With equation 16 we can analyze almost full table, then together with 18 we can analyze full table.

4.3 Moments analysis

Now we come to the core part of the analysis of the linear probing algorithm. Analytical Combinatorics method is so nice that even Knuth were happy to receive the email [6].

Let's first play with the generating function for the basic enumeraing problem. The enumerating generating function for almost full table is $\mathsf{U} F(z,u)$, we denote it as $f_0(z)$. Note $\mathsf{UH} F(z,u) = \frac{\partial}{\partial z}(zF(z,1))$**, this can be shown by the following steps:

$$\begin{aligned} & \mathsf{H} F(z,u) = \sum_{n\geq 0} F_n(u)(1 + u + u^2 + \cdots + u^n) \frac{z^n}{n!} \\ \Longrightarrow & \mathsf{UH} F(z,u) = \sum_{n\geq 0} F_n(1)(n+1) \frac{z^n}{n!} = \left(\sum_{n\geq 0} F_n(1) \frac{z^n}{n!}\right) + \left(\sum_{n\geq 0} F_n(1) n \frac{z^n}{n!}\right) \\ & = F(z,1) + \sum_{n\geq 1} F_n(1) n \frac{z^n}{n!} = F(z,1) + z \sum_{n\geq 1} F_n(1) \frac{z^{n-1}}{(n-1)!} \\ & = F(z,1) + z \sum_{n\geq 0} F_{n+1}(1) \frac{z^n}{n!} = \frac{\partial}{\partial z}(zF(z,1)) \end{aligned}$$

**Seems there is a typo in the original paper [2].

And based on 16 and the above formula, we have:

$$\begin{aligned}
&\frac{\partial \log F(z,u)}{\partial z} = \mathsf{H}F(z,u), \text{ then take } \mathsf{U} \text{ operation on both side} \\
\Longrightarrow\ & \mathsf{U}\frac{\partial \log F(z,u)}{\partial z} = \mathsf{UH}F(z,u) = \frac{\partial}{\partial z}(zF(z,1)) \\
\Longrightarrow\ & (\log f_0(z))' = (zf_0(z))' \\
\Longrightarrow\ & f_0(z) = e^{zf_0(z)} \\
\Longrightarrow\ & zf_0(z) = ze^{zf_0(z)}
\end{aligned} \tag{19}$$

Recall the tree function's definition is $T(z) = ze^{T(z)}$, we find that $f_0(z) = \frac{1}{z}T(z)$. The number of almost full tables with n elements is $n![z^n]f_0(z) = n![z^n]\frac{1}{z}T(z) = (n+1)^{n-1}$. This matches what we deduct using traditional method. For full table, $C(z,1) = \log F(z,1) = \log f_0(z) = T(z)$, so the number of full tables is n^{n-1} almost matches the traditional method.

Next, let's try to find the moments of total displacement, this is easy to compute with bivariate generating function. We denote $f_r(z)$ to be the following formula:

$$f_r(z) := \frac{\partial^r}{\partial u^r}F(z,u)\Big|_{u=1}, \tag{20}$$

and the rth factorial moment of total displacement is given by:

$$\mathbb{E}(d^{\underline{r}}) = \frac{[z^n]f_r(z)}{[z^n]f_0(z)} \qquad \text{where } d \equiv d_{n+1,n}{}^{\dagger} \tag{21}$$

In the enumerating part we apply $\mathsf{U}\partial_u^0$ to both side of 19, what if here we use the same skill that apply $\mathsf{U}\partial_u^r$ with $r = 1, 2$ to both side of 19? First we need to understand the compound operator $\mathsf{U}\partial_u\mathsf{H}$, since the bivariate generating function is a linear combination of $z^n u^k$, let's apply $\mathsf{U}\partial_u\mathsf{H}$ to it:

$$\begin{aligned}
\mathsf{U}\partial_u\mathsf{H}(z^n q^k) &= \mathsf{U}\partial_u(1+u+u^2+\cdots+u^n)z^n u^k \\
&= \mathsf{U}\big((1+2u+\cdots+nu^{n-1})u^k + (1+u+u^2+\cdots+u^n)ku^{k-1}\big)z^n \\
&= \big((1+2+3+\cdots+n)+(n+1)k\big)z^n \\
&= \frac{n(n+1)}{2}z^n + (n+1)kz^n \\
&= \mathsf{U}\frac{1}{2}z\frac{\partial^2(z(z^n q^k))}{\partial z^2} + \mathsf{U}\frac{\partial}{\partial z}\big(z\frac{\partial z^n u^k}{\partial u}\big) \\
&= \frac{1}{2}\mathsf{UZ}\partial_z^2\mathsf{Z}(z^n u^k) + \mathsf{U}\partial_z\mathsf{Z}\partial_u(z^n u^k)
\end{aligned} \tag{22}$$

Thus we know the compound operator can be written in a new form[‡]:

$$\mathsf{U}\partial_u\mathsf{H} = \frac{1}{2}\mathsf{UZ}\partial_z^2\mathsf{Z} + \mathsf{U}\partial_z\mathsf{Z}\partial_u \tag{23}$$

[†]The original paper [2] here is $d_{n,n-1}$, I think it is typo.

[‡]Seems the original paper [2] makes a mistake in its equation(14), missing a U.

Apply $\mathsf{U}\partial_u$ to equation's both sides:

$$\begin{aligned}
&\mathsf{U}\frac{\partial}{\partial u}\frac{\partial F(z,u)}{\partial z} = \mathsf{U}\big(\frac{\partial F(z,u)}{\partial u}\mathsf{H}F(z,u) + F(z,u)\frac{\partial HF(z,u)}{\partial u}\big)\\
\Longrightarrow\ &\frac{\partial}{\partial z}\mathsf{U}\frac{\partial F}{\partial u} = \mathsf{U}\frac{\partial F}{\partial u}\mathsf{U}\mathsf{H}F + \mathsf{U}F\mathsf{U}\partial_u\mathsf{H}F\\
\Longrightarrow\ &f_1' = f_1(zf_0)' + f_0(\frac{1}{2}z(zf_0)'' + (zf_1)')\\
\Longrightarrow\ &f_1' - f_1(zf_0)' - f_0(f_1 + zf_1') = \frac{1}{2}zf_0(zf_0)''\\
\Longrightarrow\ &(1-zf_0)f_1' - (f_0 + (zf_0)')f_1 = \frac{1}{2}zf_0(zf_0)''\\
\Longrightarrow\ &(1-T)f_1' - \frac{1}{z}(T + zT')f_1 = \frac{1}{2}zf_0(zf_0)''\\
\Longrightarrow\ &(1-T)f_1' - \frac{1}{z}(T + \frac{T}{1-T})f_1 = \frac{1}{2}zf_0(zf_0)''\\
\Longrightarrow\ &(1-T)f_1' - \frac{T(2-T)}{z(1-T)}f_1 = \frac{1}{2}TT''
\end{aligned} \tag{24}$$

The job now is to solve the ODE(ordinary differential equation):

$$(1-T)f_1' - \frac{T(2-T)}{z(1-T)}f_1 = \frac{1}{2}TT'' \tag{25}$$

First step is to solve the homogeneous ODF:

$$\begin{aligned}
&(1-T)f_1' - \frac{T(2-T)}{z(1-T)}f_1 = 0\\
\Longrightarrow\ &\frac{f_1'}{f_1} = \frac{T(2-T)}{z(1-T)^2}\\
\Longrightarrow\ &\log f_1 = \int \frac{T(2-T)}{z(1-T)^2}dz, \text{ and } dz = (1-T)e^{-T}dT\\
\Longrightarrow\ &\log f_1 = \int \frac{T(2-T)}{ze^T(1-T)}dT\\
\Longrightarrow\ &\log f_1 = \int 1 + \frac{1}{1-T}dT = T + \log\frac{1}{1-T}\\
\Longrightarrow\ &f_1 = \frac{e^T}{1-T}
\end{aligned} \tag{26}$$

Next step is to find the solution of the homogeneous ODF, before this, let's simplify TT'':

$$\begin{aligned}
&T = ze^T \implies T' = e^T + zT'e^T = e^T + TT'\\
\Longrightarrow\ &T'' = T'e^T + T'T' + TT''\\
\Longrightarrow\ &T'' = \frac{e^T + T'}{1-T}T' = \frac{e^T}{1-T}\frac{1}{1-T}(e^T + \frac{e^T}{1-T})\\
\Longrightarrow\ &T'' = \frac{2-T}{(1-T)^3}e^{2T}
\end{aligned} \tag{27}$$

And now we are ready for inhomogeneous ODF's solution, it is:

$$\begin{aligned}
&\frac{e^T}{1-T}\int_0^z \frac{1}{2}TT''\frac{1}{1-T}\frac{1-T}{e^T}dz \\
=&\frac{e^T}{2(1-T)}\int_0^z \frac{T(2-T)}{(1-T)^3}e^{2T}\frac{1}{e^T}\frac{1-T}{e^T}dT \\
=&\frac{e^T}{2(1-T)}\int_0^z \frac{T(2-T)}{(1-T)^2}dT \\
=&\frac{e^T}{2(1-T)}(\frac{1}{1-T}-T)
\end{aligned} \tag{28}$$

The complete solution is:

$$\frac{e^T}{2(1-T)}(C+\frac{1}{1-T}-T) \tag{29}$$

Use boundry condition and find out $C=-1$, so we have the result for $f_1(z)$:

$$zf_1(z)=\frac{1}{2}\frac{(T(z))^3}{(1-T(z))^2} \tag{30}$$

With $C(z,u)=\log F(z,u)$ we have:

$$\begin{aligned}
&\mathsf{U}\frac{\partial C(z,u)}{\partial u}=\mathsf{U}(\frac{1}{F(z,u)}\frac{\partial F(z,u)}{\partial u}) \\
\Longrightarrow\ &\mathsf{U}\frac{\partial C(z,u)}{\partial u}=\frac{f_1(z)}{f_0(z)}=\frac{(T(z))^2}{2(1-T(z))^2} \\
&\text{with equation 11 and 12} \\
\Longrightarrow\ &\mathsf{U}\frac{\partial C(z,u)}{\partial u}=\sum_{n\geq 1}\frac{1}{2}(Q(n)-1)\frac{n^n}{n!}z^n
\end{aligned} \tag{31}$$

Result in 31 is the total displacement of full table, there are n^{n-1} full tables with n elements, so we know

$$\mathbb{E}(d_{n,n})=\frac{n}{2}(Q(n)-1)=\frac{\sqrt{2\pi}}{4}n^{\frac{3}{2}}-\frac{2}{3}n+\frac{\sqrt{2\pi}}{48}n^{\frac{1}{2}}-\frac{2}{135}+O(\frac{1}{n}) \tag{32}$$

The above results shows that full hash table performs very bad.

5 Sparse Table

5.1 Simple enumerating

A sparse hash table with m slots and n elements, with $m-n$ slots empty. As in the previous section, we leave the last slot empty to make life easy. This will not impact the analysis of the algorithm (but make life much easier) because of circular symmetry[§]. The probability of the last slot empty is $\frac{m-n}{m}$. The number of all possible sparse hash tables is $\frac{m-n}{m}m^n=(m-n)m^{n-1}$.

[§]To be honest, this is not that straightfoward.

5.2 Generating Function

Sparse tables are those the number of elements n is strictly smaller than the number of slots m and there are $m-n$ empty slots. **Its combinatoric structure is sequence of almost full tables.** This idea is the most beautiful part of the analysis of linear probing hashing. A typical example is shown in figure 2, in this figure, there is a sparse hash table with 10 slots and 6 elements, empty slots do not contain numbers. Each almost full table is in the same color, there are 5 almost full tables.

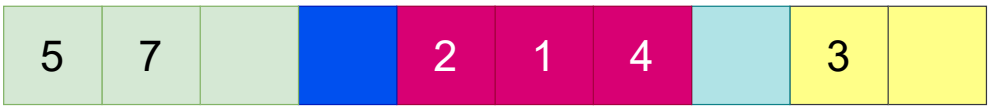

Figure 2: Sparse hash table as sequence of almost full tables

The EBGF is: $H_m(z,u) = \sum_{n\geq 0}\sum_{k\geq 0} H_{m,n}u^k\frac{z^n}{n!}$. $H_{m,n}(u) = \sum_{k\geq 0} H_{m,n}u^k$ encodes the cost (total displacement) distribution information for a sparse hash table with m slots and n elements (with the last slot empty). There are $m-n$ empty slots so that it is a $(m-n)-$SEQ of almost full hash tables, with symbolic method, we have:

$$H_{m,n}(u) = n![z^n]F(z,u)^{m-n} \tag{33}$$

5.3 Moments analysis

Let's first play with the generating function 33 for enumerating problem. The number of sparse hash table with m slots and n elements is:

$$\begin{aligned} H_{m,n}(1) &= n![z^n]F(z,1)^{m-n} \\ &= n![z^n]f_0(z)^{m-n} \\ &= n![z^n]\frac{1}{z^{m-n}}(zf_0(z))^{m-n} \\ &= n![z^n]\frac{1}{z^{m-n}}T(z)^{m-n} \\ &= n![z^m]T(z)^{m-n} \\ &= (m-n)m^{n-1} \end{aligned} \tag{34}$$

This still matches what we get using the traditional method.
For the accumulated cost, we have:

$$\begin{aligned} \mathsf{U}\frac{\partial H_{m,n}(u)}{\partial u} &= n![z^n]\mathsf{U}(m-n)F(z,u)^{m-n-1}\frac{\partial F(z,u)}{\partial u} \\ &= n![z^n](m-n)f_0(z)^{m-n-1}f_1(z) \\ &= n![z^n](m-n)\frac{1}{z^{m-n}}(zf_0(z))^{m-n-1}(zf_1(z)) \\ &= n![z^m](m-n)T(z)^{m-n-1}\frac{1}{2}\frac{T(z)^3}{(1-T(z))^2} \\ &= n![z^m]\frac{m-n}{2}T(z)^{m-n+2}\frac{1}{(1-T(z))^2} \end{aligned} \tag{35}$$

The average total displacement (first order moment) is:

$$\frac{1}{H_{m,n}(1)}\mathsf{U}\frac{\partial H_{m,n}(u)}{\partial u} = \frac{n!}{m^{n-1}}[z^m]\frac{1}{2}\frac{T(z)^{m-n+2}}{(1-T(z))^2} \tag{36}$$

To know the asymptotic result, we have to solve the problem: what is the coefficient of a relational function of the tree function? We need mathematical tool **Lagrange–Bürmann inversion formula** [7].

Theorem 2 (**Lagrange–Bürmann inversion formula**). *Given $f(z) = \frac{z}{\phi(z)}$, $\phi(z)$ is analytical function with $\phi(0) \neq 0$. $f(0) = 0$. $g(z)$ is the inverse function of $f(z)$ such that $f(g(z)) = z$. Then to find the coefficient of arbitrary function $H(g(z))$, we have:*

$$[z^n]H(g(z)) = [u^n]H(u)\phi(u)^{n-1}(\phi(u) - u\phi'(u)). \tag{37}$$

Let's play with theorem 2 for some cases that we have already known at the start of this article.

$$\begin{aligned}
&\text{Given } g(z) = T(z), \text{ then } f(z) = \frac{z}{e^z} \text{ is the inverse function of } T(z), \phi(z) = e^z \\
&H(T(z)) = \frac{T(z)}{(1-T(z))^2} \implies H(u) = \frac{u}{(1-u)^2}, \text{ apply theorem 2,} \\
&[z^n]H(T(z)) = [z^n]\frac{T(z)}{(1-T(z))^2} = [u^n]\frac{u}{(1-u)^2}e^{nu}(1-u) = [u^n]\frac{ue^{nu}}{1-u} \implies \\
&[z^n]H(T(z)) = [u^{n-1}]\frac{e^{nu}}{1-u} = [u^{n-1}]\sum_{k\geq 0} Q_0(n,k)n^k\frac{u^k}{k!} = \frac{n^n}{n!}Q_0(n,n-1)
\end{aligned} \tag{38}$$

It matches what we have found in equation 11. Thus to finish the analysis of average total displacement for sparse hash table we just need to apply theorem 2 to equation 36.

$$\begin{aligned}
&g(z) = T(z), \phi(z) = e^z, H(u) = \frac{u^{m-n+2}}{(1-u)^2} \\
\implies &[z^m]H(T(z)) = [u^m]\frac{u^{m-n+2}}{(1-u)^2}(1-u)e^{mu} = [u^m]\frac{u^{m-n+2}}{1-u}e^{mu} \\
\implies &[z^m]H(T(z)) = [u^{n-2}]\sum_{k\geq 0} Q_0(m,k)m^k\frac{u^k}{k!} = Q_0(m,n-2)\frac{m^{n-2}}{(n-2)!} \\
\implies &\mathbb{E}(d_{m,n}) = \frac{n!}{2m^{n-1}}Q_0(m,n-2)\frac{m^{n-2}}{(n-2)!} = \frac{n}{2}\frac{n-1}{m}Q_0(m,n-2) \\
\implies &\mathbb{E}(d_{m,n}) = \frac{n}{2}(Q_0(m,n-1)-1)
\end{aligned} \tag{39}$$

Let $\alpha = \frac{n}{m}, n = \alpha m$ be the load of the hash table, then we can get the asymptotic results:

$$\begin{aligned}
\mathbb{E}(d_{m,n}) &= \frac{n}{2}\big(\frac{1}{1-\alpha} - \frac{1}{(1-\alpha)^3}\frac{\alpha}{n} + O(n^{-2}) - 1\big) \\
&= \frac{\alpha}{2(1-\alpha)}n - \frac{\alpha}{2(1-\alpha)^3} + O(n^{-1})
\end{aligned} \tag{40}$$

From the aboev result we understand why linear probing hashing works well when the load is not large.

6 The end and start of the tour

I finish the deduction of average searching cost of hash table using linear probing for both full table and sparse table. Many important parts of the original paper [2] is not covered by this article, e.g. high order moments analysis, limit distribution analysis. Two years ago I perused the HyperLogLog paper of the same author, and then the tour ended; a week ago I started for this paper, now the tour ends. It is very happy to finish a tour which at the first glance looks crazy for myself.

References

[1] F. Bergeron, F. Bergeron, G. Labelle, and P. Leroux. *Combinatorial species and tree-like structures*. Number 67. Cambridge University Press, 1998.

[2] P. Flajolet, P. Poblete, and A. Viola. On the analysis of linear probing hashing. *Algorithmica*, 22(4):490–515, 1998.

[3] D. Knuth731 Knuth. E.," the art of computer programming, vol. 3: Sorting and searching,", 1973.

[4] R. Sedgewick and P. Flajolet. *An introduction to the analysis of algorithms*. Pearson Education India, 2013.

[5] R. Sedgewick and K. Wayne. Algorithms (4th edn), 2011.

[6] A. Viola. 50 years of linear probing hashing, 2013.

[7] Wikipedia. Lagrange inversion theorem — Wikipedia, the free encyclopedia. http://en.wikipedia.org/w/index.php?title=Lagrange%20inversion%20theorem&oldid=1060554415, 2022. [Online; accessed 07-June-2022].