PRAMANA © Indian Academy of Sciences Vol. 64, No. 6
— journal of June 2005
physics pp. 1175–1189

Understanding search trees via statistical physics

SATYA N MAJUMDAR[1,2], DAVID S DEAN[2] and P L KRAPIVSKY[3]
[1]Laboratoire de Physique Théorique et Modèles Statistiques, Université Paris-Sud, Bât 100, 91405 Orsay Cedex, France
[2]Laboratoire de Physique Theorique (UMR C5152 du CNRS), Université Paul Sabatier, 31062 Toulouse Cedex, France
[3]Center for Polymer Studies and Department of Physics, Boston University, Boston, Massachusetts 02215, USA
E-mail: majumdar@ipno.in2p3.fr; satya@irsamc.ups-tlse.fr

Abstract. We study the random m-ary search tree model (where m stands for the number of branches of the search tree), an important problem for data storage in computer science, using a variety of statistical physics techniques that allow us to obtain exact asymptotic results. In particular, we show that the probability distributions of extreme observables associated with a random search tree such as the height and the balanced height of a tree have a travelling front structure. In addition, the variance of the number of nodes needed to store a data string of a given size N is shown to undergo a striking phase transition at a critical value of the branching ratio $m_{\mathrm{c}} = 26$. We identified the mechanism of this phase transition and showed that it is generic and occurs in various other problems as well. New results are obtained when each element of the data string is a D-dimensional vector. We show that this problem also has a phase transition at a critical dimension, $D_{\mathrm{c}} = \pi/\sin^{-1}\left(1/\sqrt{8}\right) = 8.69363\ldots$.

Keywords. Search trees; fragmentation; travelling fronts; phase transition.

PACS Nos 02.50.-r; 89.20.Ff; 89.75.Hc

1. Introduction

'Search trees' are the objects of key interest in an important area of computer science called 'sorting and searching' [1] which deals with the basic question: How does one store the incoming data to a computer in an efficient way so that one spends the minimum time in searching a given data element if required later? Amongst various search algorithms, the tree based sorting and search algorithms turn out to be the most efficient ones. One of the simplest such algorithms is the so-called binary search algorithm (BSA) which can be understood by the following simple example. Consider a data string consisting of N elements which are labelled by the N integers $\{1, 2, \ldots, N\}$. These could be the months of the year or the names of people etc. Let us assume that these data appear in a particular order, say $\{6, 4, 5, 8, 9, 1, 2, 10, 3, 7\}$ for $N = 10$ integers. These data are first stored on a binary tree following the

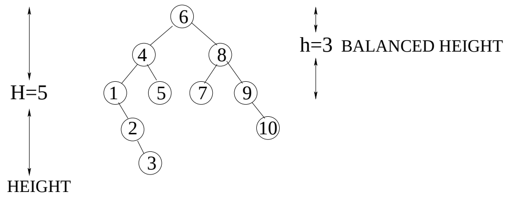

Figure 1. The binary search tree associated with the data string $\{6, 4, 5, 8, 9, 1, 2, 10, 3, 7\}$.

simple dynamical rule: the first element 6 is stored at the root of the tree (see figure 1). The next element in the string is 4. We compare it with 6 at the root and since $4 < 6$, we store 4 in the left daughter node of the root. Had it been bigger than the root 6, we would have stored it in the right daughter node. The next element in the string is 5. We again start from the root, see that $5 < 6$, so we go to the left branch. There we encounter 4 and we find $5 > 4$, so we store 5 in the right daughter node of 4. This process is continued till all the $N = 10$ elements are assigned their nodes and we get a unique binary search tree (BST) (see figure 1) for this particular data string $\{6, 4, 5, 8, 9, 1, 2, 10, 3, 7\}$.

Once the data are stored on the tree, it takes very little time to search a required element. For example, suppose we are looking for the element 7. We start from the root and comparing with 6 at the root, we know that 7 must be on the right branch since $7 > 6$. We then go down one level and next compare 7 with 8 (see figure 1) and since $7 < 8$, we look in the left subtree below 8 and immediately find 7. Thus, by construction, we eliminate searching one half of the subtrees at any level. This makes the search process very efficient. In fact, typical search time to find an element is $t_{\text{search}} = D$ where D is the depth of the element in the tree. Since, roughly speaking, $2^D \sim N$, one gets $t_{\text{search}} \sim O(\log N)$, which is far better than linear search that takes $t_{\text{search}} \sim O(N)$.

An immediate generalization of a BST is an m-ary search tree where the tree has m branches. The BST corresponds to $m = 2$. An m-ary search tree is constructed in the following way. Each node of the tree can now hold at most $(m-1)$ elements. One first collects the first $(m - 1)$ elements of the data string and stores them together in the root of the tree in an ordered sequence $x_1 < x_2 \cdots < x_{m-1}$ (see figure 2 for $m = 3$). Next when the m-th element arrives, one compares it first with x_1. If $x_m < x_1$, the new element x_m is assigned to the leftmost daughter node of the root. If $x_1 < x_m < x_2$, x_m goes to the daughter node in the second branch and so on. Each subsequent incoming element is assigned to either of the m branches according to the above rule. As an example, the same data string $\{6, 4, 5, 8, 9, 1, 2, 10, 3, 7\}$ of size $N = 10$ is stored on a $m = 3$ tree in figure 2. Note that for $m > 2$, some of the nodes of the tree are saturated to their capacity, i.e., are fully occupied with $(m-1)$ elements, while some others are only partially occupied.

Once an m-ary tree is constructed, one can define a number of observables associated with the tree which provides information about the structure of the tree. The knowledge of how these observables depend on the data size N is of central

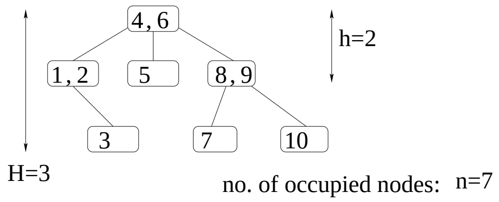

Figure 2. The $m = 3$ search tree associated with the data string $\{6, 4, 5, 8, 9, 1, 2, 10, 3, 7\}$.

importance in 'sorting and searching'. Amongst many observables, we focus here on three central objects:

1. The height H_N of the tree which is defined as the distance of the farthest node from the root. For example, in figure 2, we have $H = 3$. The height H_N measures the maximum possible time to search an element, i.e., it is a measure of the worst case scenario.
2. The balanced height h_N of the tree, defined as the maximum depth up to which the tree is balanced, i.e., all the nodes up to that level are at least partially occupied. In the example of figure 2 we have $h = 2$. Balancing a tree is important for optimizing search algorithms and hence h_N is an important observable.
3. The number of non-empty nodes n_N of the tree which tells us how many nodes typically one needs to store a data of size N. For example, in figure 2, one has $n = 7$. Note that for the binary case $m = 2$, one has trivially $n_N = N$ since each node can contain only one element. However, for $m > 2$, n_N becomes nontrivial since some of the nodes may only be partially filled.

Usually the data arrive at a computer in random order. To study this situation, one considers the simplest model called the 'random m-ary search tree model' (RmST) where one assumes that the incoming data string can arrive in any of the $N!$ possible order or sequence, each with equal probability. For each of these sequences, one has an m-ary tree and the associated observables H_N, h_N and n_N. As the sequence changes, the corresponding tree changes and hence these observables also take on different values. For example, in figure 3, we show two sequences, their corresponding $m = 3$ trees and the values of the three observables. The central question of importance is: given that all the $N!$ sequences occur with equal probability, what are the statistics of H_N, h_N and n_N? For example, what are the averages, variances or even the full probability distributions of these observables?

The statistics of H_N and h_N have been studied by computer scientists over the past two decades and many nontrivial results have been found [2–5]. For example, the average height and the average balanced height of a random m-ary search tree have the following asymptotic behaviors for large N:

$$\begin{aligned}\langle H_N\rangle &\approx a_m \log N + b_m \log(\log N) + \cdots ,\\ \langle h_N\rangle &\approx c_m \log N + d_m \log(\log N) + \cdots .\end{aligned} \tag{1}$$

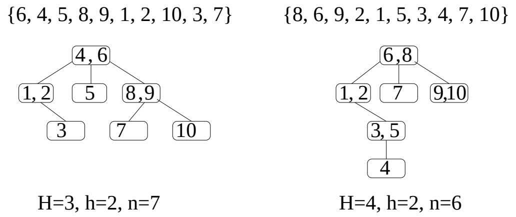

Figure 3. The $m = 3$ search trees associated respectively with the data strings $\{6, 4, 5, 8, 9, 1, 2, 10, 3, 7\}$ and $\{8,6,9,2,1,5,3,4,7,10\}$.

While the leading $\log(N)$ behavior was proved by Devroye [4] who also computed the coefficients a_m and c_m, the subleading double logarithmic behavior was conjectured only recently by Hattori and Ochiai [6], who found $b_2 \approx -1.9$ numerically. Also, the variance and even the higher moments of H_N and h_N were found to be independent of N for large N [7,8].

The study of the statistics of n_N, on the other hand, is relatively recent [9–11]. Chern and Hwang [10] recently found that while the average $\mu_N = \langle n_N \rangle \sim N$ for large N (as one should expect), the variance $\nu(N) = \langle (n_N - \mu(N))^2 \rangle$ undergoes a striking phase transition as a function of m. They found that

$$\begin{aligned} \nu(N) &\sim N \qquad && \text{for} \quad m \leq 26 \\ &\sim N^{2\theta(m)} \qquad && \text{for} \quad m > 26, \end{aligned} \tag{2}$$

where the exponent $\theta(m) > 1/2$ depends on m for $m > 26$.

The various important results mentioned above were derived by computer scientists using sophisticated probabilistic methods which, though rigorous, are often not simple. As physicists, one would like to understand and derive these results in a physically transparent way. Moreover, as it often happens, a physical approach has the advantage that it can make links with other problems and also the generalization often becomes easier. In a series of recent papers [12–15], we were able to build up a statistical physical approach to the RmST problem which not only allowed us to rederive many asymptotically exact results (known previously only via rigorous probabilistic methods) in a physically transparent way, but also led to many new results, generalizations and links to other problems. For example, we were able to generalize our results to other search trees such as the 'digital search trees' (DST) (which has links to the Lempel–Ziv data compression algorithm) and found an exact mapping between the DST and the problem of the directed diffusion limited aggregation (DLA) problem on the Bethe lattice [16]. The latter problem was first studied numerically by Bradley and Strenski [17] and remained unsolved for many years. Our approach provides an exact asymptotic result for the DLA problem [16].

Our strategy was to first map the RmST problem to a random fragmentation problem which was more amenable to statistical physical analysis. The main new discovery was that the distributions of the height H_N and the balanced height h_N, which are 'extreme' variables, have a 'travelling front' structure. The 'travelling fronts' appear in many physics and biology problems and have been well-studied over the past few decades [18]. The techniques developed in analysing travelling fronts were then useful to derive many asymptotically exact results for the RmST problem. Subsequently, we found that in many problems where one is interested in finding the statistics of extreme variables, there is often a 'travelling front' structure [19, 20].

For the number of non-empty nodes n_N, which is not an extreme variable, a different statistical physics approach (equivalent to a backward Fokker–Planck method) was used which allowed us to understand the mechanism of the phase transition, the significance of the critical number 26 and calculate the exponent $\theta(m)$ exactly [15]. We were also able to show that this phase transition is rather generic and occurs in other problems as well. Our approach allowed us to generalize to the case when the data string consists of N D-dimensional vectors. For example, we found that there is again a phase transition at a critical dimension $D_{\rm c} = \pi/\sin^{-1}\left(1/\sqrt{8}\right) = 8.69363\ldots$. In the next few sections we outline our approach and state the main results.

2. Mapping to a fragmentation process

Our strategy is to first map the problem of RmST to a random fragmentation problem [13, 15], which in some sense, is more familiar to physicists. This fragmentation procedure can then be viewed as a dynamical process and one can write down its evolution equation fairly easily. This mapping is best understood in terms of an example. Let us take our favorite data string $\{6, 4, 5, 8, 9, 1, 2, 10, 3, 7\}$ and store it on an $m = 3$ tree as in figure 2 and also shown in the left half of figure 4. In the fragmentation problem, one starts with a stick (or interval) of length $N = 10$. Once the first two elements 4 and 6 are stored in the root of the tree, the remaining elements will belong either to the interval $[1-3]$, $[5]$, or $[7-10]$, which are subsequently completely disconnected from each other. Thus storing the first two elements is equivalent, in the fragmentation problem, to breaking the original interval $[1-N]$ of length N into 3 smaller intervals $[1-3]$, $[5]$, and $[7-10]$. The two break points 4 and 6 are chosen uniformly from the N points $\{1, 2, 3, \ldots, N\}$ in the RmST problem (shown by arrows in figure 4). Next, when the element 5

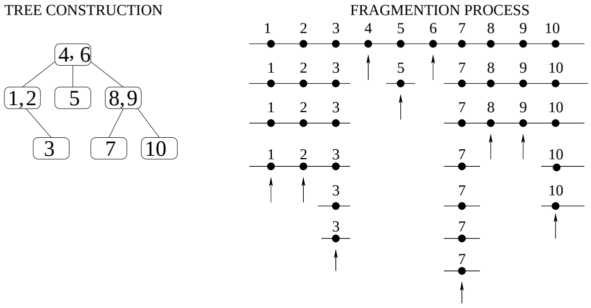

Figure 4. The $m = 3$ search tree associated with the data string $\{6, 4, 5, 8, 9, 1, 2, 10, 3, 7\}$ and the corresponding fragmentation process.

arrives in the tree, it corresponds to breaking the interval containing the element 5 randomly into 3 parts (this breaking is not shown explicitly in figure 4). The process is then repeated for other elements. Note that in the fragmentation problem, an interval breaks iff there is an element (shown by black dots) inside the interval. Thus there is a threshold phenomenon: if the length of a stick is too small so that it does not have an element (black dot) in it, one does not fragment it any more. We denote this threshold length by N_0 (in our example, $N_0 = 1$). It just sets the unit of length and its actual value is not important for the asymptotic large N behaviors. Those intervals which still have black dots in them (and thus have lengths $> N_0$) are thus 'alive' and will fragment subsequently, but those whose lengths are $< N_0$ are 'dead'. Thus, when all the N elements are stored in the tree, all the intervals in the fragmentation problem become 'dead'.

Note that, in the fragmentation problem, at each step (shown by different levels on the right in figure 4) there is only one 'splitting event'. Each time an interval splits, it corresponds to storing in a node on the tree. Thus, completing a tree is equivalent to ending one 'history' of the fragmentation process (at the end of which all intervals are 'dead'). Evidently, the number of non-empty nodes n_N in the tree is exactly same as the total number of 'splitting events' in the history of the fragmentation process (for example, in figure 4 the number of nodes on the tree and the number of splitting events are both 7). Let l_is denote the lengths of intervals in the fragmentation problem at a given stage. One can then set up a dictionary between the two problems [13, 15] and it is easy to see that

1. Height H_N: $\text{Prob}[H_N < n] = \text{Prob}[l_1 < 1,\ l_2 < 1, \ldots$ after n steps of fragmentation.]
2. Balanced height h_N: $\text{Prob}[h_N > n] = \text{Prob}[l_1 > 1,\ l_2 > 1, \ldots$ after n steps of fragmentation.]
3. Number of non-empty nodes n_N: $\text{Prob}[n_N = n]=$ Prob[there are a total of n 'splitting events' till the end of the fragmentation process.]

3. Analysis of the fragmentation problem

Once this dictionary is set up, one can forget about the original tree problem and focus on the fragmentation problem. For simplicity, we will also assume that the lengths of sticks in the fragmentation problem are continuous variables. This is because the original discrete problem and the continuous problem will have the same asymptotic properties for large N, but the continuous problem is easier to handle. Thus, in the continuous problem, we start with a stick of length N where N is large. We break it randomly into m fragments of lengths r_1N, r_2N, $\ldots$, r_mN where the fractions r_is are random numbers between $[0, 1]$ that satisfy the length conservation condition, $\sum_{i=1}^{m} r_i = 1$. At this point, we will consider a general problem where the fractions r_is are drawn from a normalized joint distribution $\eta(r_1, r_2, \ldots, r_m)$. The RmST problem would correspond to a specific choice of this joint distribution. Note that in the RmST problem, all the $N!$ permutations of the original sequence occur equally likely. This means that the first $(m-1)$ elements are random, each drawn independently and uniformly from $[1-N]$. In the fragmentation language,

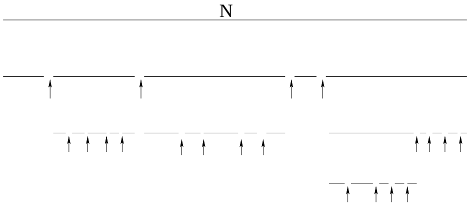

Figure 5. The fragmentation process with continuous lengths for $m = 5$. The arrows denote the break points.

this means that each of the fractions r_1, r_2, ..., r_{m-1} is chosen from a uniform distribution between 0 and 1 and then one sets, $r_m = 1-(r_1+r_2+\cdots+r_{m-1})$. This leads to the normalized joint distribution $\eta(r_1, r_2, \ldots, r_m) = (m-1)!\delta(\sum_i^m r_i - 1)$ [13]. One of the advantages of our method is that it allows us to obtain exact results for arbitrary joint distribution of the fraction r_is, not necessarily only for the uniform case. The RmST problem just corresponds to a special case.

After the first splitting event, we examine the lengths of each of the m fragments. If the length of a fragment is already less than $N_0 = 1$, we proclaim it 'dead' and it does not split any further. Those fragments with lengths $> N_0 = 1$ are 'alive' and each of those 'alive' fragments is further split into m pieces by drawing, for each piece independently, a set of fractions r_is from the identical joint distribution $\eta(\{r_i\}) = \eta(r_1, r_2, \ldots, r_m)$. This process is then repeated till all the intervals become 'dead', i.e., their lengths become $< N_0 = 1$. A pictorial representation is given in figure 5 with $m = 5$.

For subsequent analysis, it is useful to define the 'marginal' distribution $\eta(r_i)$ of any one of the fractions as

$$\eta(r_i) = \int \eta(\{r_i\})\, \mathrm{d}r_1 \ldots \mathrm{d}r_{i-1}\mathrm{d}r_{i+1} \ldots \mathrm{d}r_m. \tag{3}$$

For simplicity, we will assume isotropy, i.e., $\eta(r_i) = \eta(r)$ is independent of the index i and is thus the same for each fragment. For example, for the RmST problem, one easily gets [13]

$$\eta(r) = (m-1)(1-r)^{m-2} \tag{4}$$

for $0 \leq r \leq 1$. Note that for binary trees $m = 2$, where one breaks a stick into two pieces, one gets $\eta(r) = 1$ for $0 \leq r \leq 1$, the usual uniform distribution for the break point.

3.1 *The height and the balanced height*

Let us denote the cumulative height distribution Prob$[H_N < n]$ by $P(n, N)$. Using the dictionary outlined before, we have $P(n, N)$=Prob$[l_1 < 1,\ l_2 < 1, \ldots$ after n steps of fragmentation] where l_is are the lengths of the intervals. It is then easy to set up a recursion satisfied by $P(n, N)$ for the fragmentation process. Consider the first splitting where we have m new intervals of lengths $r_1N, r_2N, \ldots, r_mN$.

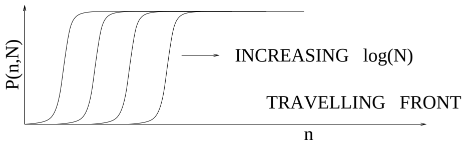

Figure 6. The travelling front structure of the solution of eq. (6).

Each of these new pieces will have subsequent histories of evolution completely independent of each other. Hence, it follows that

$$P(n,N) = \int \left[\prod_{i=1}^{m} P(n-1, r_i N)\right] \eta\left(\{r_i\}\right) \mathrm{d}r_1 \mathrm{d}r_2 \ldots \mathrm{d}r_m, \tag{5}$$

satisfying the condition, $P(n,1) = 1$ for all $n \geq 1$ (this follows from the fact that if the initial length is 1, after the first splitting all the lengths will be < 1). It is further useful to make a change of variables, $t = \log(N)$ and $\epsilon_i = -\log(r_i)$. The joint distribution of ϵ_is are given by $\tilde{\eta}\left(\{\epsilon_i\}\right) \prod_i \mathrm{d}\epsilon_i = \eta\left(\{r_i\}\right) \prod_i \mathrm{d}r_i$. Then eq. (5) reduces to

$$P(n,t) = \int \left[\prod_{i=1}^{m} P(n-1, t-\epsilon_i)\right] \tilde{\eta}\left(\{\epsilon_i\}\right) \mathrm{d}\epsilon_1 \mathrm{d}\epsilon_2 \ldots \mathrm{d}\epsilon_m. \tag{6}$$

Equation (6) is nonlinear and hence is difficult to solve exactly. However, if one plots the numerical solution of eq. (6), one finds a travelling front structure as shown in figure 6.

This means that the solution at late 'times' t has the structure, $P(n,t) \sim f(n - n_f(t))$, where $n_f(t)$ is the position of the front at 'time' t. Note that the front retains its shape as t increases which indicates that the width of the front remains $O(1)$ even as $t \to \infty$. The front position advances with a uniform velocity, i.e., $n_f(t) \approx vt$, to leading order for large t where the velocity v is yet to be determined. We substitute $P(n,t) = 1 - F(n - vt)$ in eq. (6) and then focus near the large n tail where F is small and hence one can linearize the equation to get

$$F(x) = m \int_0^{\infty} F(x - 1 + v\epsilon)\tilde{\eta}(\epsilon)\mathrm{d}\epsilon, \tag{7}$$

where $\tilde{\eta}(\epsilon)\mathrm{d}\epsilon = \eta(r)\mathrm{d}r$ is the effective induced distribution associated with any one of the fractions. This linear equation clearly admits an exponential solution $F(x) = \mathrm{e}^{-\lambda x}$ provided λ is related to v via the dispersion relation

$$1 = m\mathrm{e}^{\lambda} \int_0^{\infty} \mathrm{e}^{-\lambda v \epsilon}\tilde{\eta}(\epsilon)\mathrm{d}\epsilon. \tag{8}$$

Thus, in principle, one can have a whole family of possible velocities $v(\lambda)$ parametrized by λ. However, in practice, the front has a unique velocity. So, how does one select this unique velocity from a continuous one parameter family of possible velocities? It turns out that the solution $v(\lambda)$ of eq. (8) is a nonmonotonic function of λ with a single minimum at $\lambda = \lambda^*$ that depends on the distribution $\tilde{\eta}(\epsilon)$. According to the velocity selection principle developed in the travelling front literature [18, 20], the front always chooses this minimum velocity $v(\lambda^*)$ as long as the initial condition is sharp enough. Thus the leading front position is given by $n_f(t) \approx v(\lambda^*)t$ where $v(\lambda^*)$ is obtained by minimizing $v(\lambda)$ in eq. (8) with respect to λ. Moreover, it turns out that the leading front position has an associated slow logarithmic correction [18]

$$n_f(t) = v(\lambda^*)t - \frac{3}{2\lambda^*}\log(t) + \cdots . \tag{9}$$

Note that since $\mathrm{Prob}[H_N < n] = P(n, N)$, the expected height $\langle H_N \rangle = \sum_n [1 - P(n, N)] \approx n_f(t)$ where $t = \log(N)$. This follows from the fact that the front rises sharply from 0 for $n < n_f(t)$ to 1 for $n > n_f(t)$. Thus, the summation $\sum_n [1 - P(n, N)]$ can be replaced by the front location $n_f(t)$. Using eq. (9), we then get

$$\langle H_N \rangle = v(\lambda^*)\log N - \frac{3}{2\lambda^*}\log\left(\log(N)\right) + \cdots . \tag{10}$$

This then provides a physical derivation of the result in eq. (1) where we identify the constant $a_m = v(\lambda^*)$ with the velocity of the front and the constant $b_m = -3/2\lambda^*$ as the prefactor of the correction term. Note that our result is more general than the RmST (which is just a special case where the break points are chosen uniformly). Our derivation also provides a proof for the double logarithmic form of the correction term previously only conjectured by Hattori and Ochiai [6].

For the RmST problem, we have $\eta(r)$ from eq. (4). This gives, $\tilde{\eta}(\epsilon) = (m - 1)(1 - \mathrm{e}^{-\epsilon})^{m-2}\mathrm{e}^{-\epsilon}$. Substituting this in eq. (8), we get the dispersion relation

$$m(m-1)\mathrm{e}^{\lambda}B(\lambda v + 1, m - 1) = 1, \tag{11}$$

where $B(m, n)$ is the standard beta function. For example, for the binary case $m = 2$, one gets from eq. (11), $v(\lambda) = (2\mathrm{e}^{\lambda} - 1)/\lambda$ which has a minimum at $\lambda^* = 0.76804\ldots$ with $v(\lambda^*) = 4.31107\ldots$. One then gets for $m = 2$ an exact result,

$$\langle H_N \rangle = 4.31107\ldots\log N - 1.95303\ldots\log\left(\log(N)\right) + \cdots . \tag{12}$$

Similarly, one can derive the exact asymptotic behavior for all m and for arbitrary fraction distribution $\eta(r)$ [13]. Note that for the binary case $m = 2$, the same double logarithmic correction term was also found by Reed [21] using rigorous probabilistic methods, but our results are more general.

For the balanced height h_N, the analysis is similar. The cumulative probability $Q(n, N) = \mathrm{Prob}[h_N > n]$ satisfies exactly the same recursion relation as in eq. (5), except the initial condition is different [13]. One has, $Q(n, 1) = 1$ for $n \leq 1$ and $Q(n, 1) = 0$ for $n > 1$. Again, the solution has a travelling front structure, except now it has a $[1-0]$ front as opposed to the $[0-1]$ front in the height case. Proceeding along the same path, one obtains the asymptotic front position and hence the average balanced height

$$\langle h_N\rangle = v(\lambda^*)\log N + \frac{3}{2\lambda^*}\log\left(\log(N)\right) + \cdots, \tag{13}$$

where $v(\lambda^*)$ is determined by maximizing $v(\lambda)$ obtained from the dispersion relation

$$1 = m\mathrm{e}^{-\lambda}\int_0^\infty \mathrm{e}^{+\lambda v\epsilon}\tilde{\eta}(\epsilon)\mathrm{d}\epsilon. \tag{14}$$

Note that this dispersion relation is the same as in eq. (8) provided one changes the sign of λ. This reflects the so-called 'duality' between the height and the balanced height [13]. For the $m = 2$ binary case, we get from eq. (14), $v(\lambda) = (1 - 2\mathrm{e}^{-\lambda})/\lambda$ which has a maximum at $\lambda^* = 1.67835\ldots$ and $v(\lambda^*) = 0.373365\ldots$. This gives [13]

$$\langle h_N\rangle = 0.373365\ldots\log N + 0.89374\ldots\log\left(\log(N)\right) + \cdots. \tag{15}$$

Note that the sign of the correction term is different in eqs (12) and (15). Similarly, one can derive exact asymptotic results for all m as well as for any arbitrary distribution $\eta(r)$.

3.2 *Number of non-empty nodes*

We now turn to the statistics of the number of non-empty nodes n_N required to store a data string of size N. Once again, the fragmentation representation turns out to be useful. One can easily write down a recursion relation for n_N by noting that n_N is just the total number of splitting events in the fragmentation process till it stops, given that it started with an initial stick of length N. After the first splitting one has m pieces of lengths r_1N, $r_2N, \ldots, r_mN$ whose subsequent histories are completely independent of each other. Note that an interval splits if its length is $>N_0$ where N_0 is the threshold length. Evidently, if the starting length $N < N_0$, $n_N = 0$ since there would not be any splitting. However, if $N > N_0$, one can write a recursion [15]

$$n_N \equiv n_{r_1N} + n_{r_2N} + \cdots + n_{r_mN} + 1, \tag{16}$$

where the fractions r_is are again random numbers satisfying $\sum_{i=1}^m r_i = 1$ that are drawn from a joint distribution $\eta\left(\{r_i\}\right)$. The term 1 on the right-hand side of eq. (16) just counts the first splitting and the rest of the terms count the total number of subsequent splitting events arising from each of the m pieces generated after the first splitting. The $\equiv$ symbol represents 'equivalence in law', i.e., the left- and the right-hand side of the $\equiv$ symbol have the same probability distribution.

Taking average on both sides of eq. (16), one finds that the average number of nodes or the 'splitting events' $\mu(N) = \langle n_N\rangle$ satisfies an integral equation [15]

$$\mu(N) = m\int_{N_0/N}^1 \mu(rN)\eta(r)\mathrm{d}r + 1. \tag{17}$$

This integral equation can be solved exactly [15]. One finds that, $\mu(N) = g(N/N_0)$ where the scaling function $g(z)$ is given by

$$g(z) = \alpha_0 + \alpha_1 z + \sum_{k=2}^{\infty} \alpha_k z^{\lambda_k}, \tag{18}$$

where λ_ks are the zeros of the equation

$$m \int_0^1 r^{\lambda} \eta(r) \mathrm{d}r = 1, \tag{19}$$

and thus have the real part $\mathrm{Re}(\lambda_k) < 1$. The leading behavior of the average for large N is given by the linear term and one gets $\mu(N) \sim \alpha_1 N/N_0$ where

$$\alpha_1 = -\frac{1}{m \int_0^1 r \log(r) \eta(r) \mathrm{d}r}. \tag{20}$$

For the RmST problem, we have $\eta(r) = (m-1)(1-r)^{m-2}$ which gives $\alpha_1 = 1/\sum_{k=2}^{m} 1/k$.

For the variance $\nu(N) = \langle (n_N - \mu(N))^2 \rangle$, one can similarly write down a recursion relation [15] starting from eq. (16),

$$\nu(N) = m \int_{N_0/N}^{1} \nu(rN) \eta(r) \mathrm{d}r + J, \tag{21}$$

where J represents a 'source' term that depends on the form of the first moment $\mu(N)$. More precisely, if $S = \sum_{i=1}^{m} \mu(r_i N)$, then $J = \langle (S - \langle S \rangle)^2 \rangle$. The significant fact about this problem is that the equation for the second moment 'closes' in the sense that it involves only second and first moments, but not higher moments. It does not have the usual hierarchy problem that one often encounters in statistical mechanics problem. This fact makes this problem analytically tractable. This source term J also turns out to be responsible for driving the 'phase transition' in the variance. This is a new mechanism of phase transition that one has not encountered before in other problems.

Using the exact solution for the first moment $\mu(N)$ from eq. (18), one can evaluate the source term J which turns out to be only a function of $z = N/N_0$ and for large z one gets

$$J(z) \approx \beta_1 z^{2\lambda_2} + \beta_2 z^{2\lambda_2^*} + \beta_3 z^{\lambda_2 + \lambda_2^*} + \cdots, \tag{22}$$

where λ_2 (and its complex conjugate λ_2^*) are the nearest zeros of the equation, $m \int_0^1 r^{\lambda} \eta(r) \mathrm{d}r = 1$ to the left of the line $\mathrm{Re}(\lambda) = 1$ in the complex λ plane. Substituting this asymptotic behavior of $J(z)$ in eq. (21) and solving the integral equation, one finds that $\nu(N) = Y(N/N_0)$ where the asymptotic behavior of $Y(z)$ for large z depends on the value of $\mathrm{Re}(2\lambda_2)$. One finds that as $z \to \infty$, $Y(z) \sim z$ (as in the case of the first moment) provided $\mathrm{Re}(2\lambda_2) < 1$. In this case, the source term J turns out to be insignificant and gives rise only to subleading correction terms. However, if $\mathrm{Re}(2\lambda_2) > 1$, the source term $J(z)$ becomes significant and controls the asymptotic behavior of $Y(z)$ and one gets, $Y(z) \sim z^{2\theta}$ where $\theta = \mathrm{Re}(\lambda_2)$.

Note that the root λ_2 is a function of m. As one tunes m, λ_2 changes but always stays to the left of the line $\mathrm{Re}(\lambda) = 1$ in the complex λ plane. However, for small

m, $\mathrm{Re}(2\lambda_2) < 1$, i.e., λ_2 stays to the left of the line $\mathrm{Re}(\lambda) = 1/2$. Then as m exceeds a critical value m_c, λ_2 crosses the line $\mathrm{Re}(\lambda) = 1/2$ from its left to its right and $\mathrm{Re}(2\lambda_2) > 1$, leading to a phase transition in the large N behaviour of $\nu(N)$. Thus the critical value of m_c is determined from the condition, $\mathrm{Re}(\lambda_2) = 1/2$. For the RmST problem, substituting $\eta(r) = (m-1)(1-r)^{m-2}$ in eq. (19) one gets, $m(m-1)B(m-1, \lambda+1) = 0$. One then obtains λ_2 using the Mathematica. Setting $\mathrm{Re}(\lambda_2) = 1/2$ determines the critical value, $m_\mathrm{c} = 26.0561\ldots$. Note that, once we have written down the moment equations, m can be treated as a continuous parameter, even though in actual search trees m is always an integer. We thus get a very general result,

$$\begin{aligned}\nu(N) &\sim N \qquad &&\text{for} \quad m \le m_\mathrm{c} \\ &\sim N^{2\theta(m)} \qquad &&\text{for} \quad m > m_\mathrm{c},\end{aligned} \tag{23}$$

for arbitrary breaking distribution $\eta(r)$ where m_c is determined from $\mathrm{Re}(\lambda_2) = 1/2$ and $\theta(m) = \mathrm{Re}(\lambda_2)$ where λ_2 is determined from eq. (19). For the RmST case in particular, we get $m_\mathrm{c} = 26.0561\ldots$.

Thus, we have identified a simple mechanism (driven by the source term) of a rather striking and nontrivial phase transition in a generic fragmentation problem [15]. There is a physical meaning associated with this phase transition. For $m < m_\mathrm{c}$, the fluctuation (variance) in the number of splitting events scales as N for large N and the central limit theorem holds. In fact, one finds that the full distribution of n_N is Gaussian for $m < m_\mathrm{c}$. However, for $m > m_\mathrm{c}$, rare events occasionally give rise to huge fluctuations. In the language of the fragmentation problem, note that the effective distribution of the fraction $\eta(r) = (m-1)(1-r)^{m-2}$ gets highly localized around $r = 0$ for large m. This means that for large m, most of the m fragments have very tiny lengths (which thus become 'dead') except one which has a huge length (due to the length conservation condition, $\sum_{i=1}^{m} r_i = 1$). Thus this large piece will persist for a long time and one will get a huge number of splitting events. This qualitative argument, of course, does not explain why there is a sharp phase transition. For that, one has to carry out explicit calculations as done here.

4. Generalization to vector data string

So far, we have considered the storing of a data string of size N on a tree where each element of the data is a scalar. A natural generalization of this is when the data consist of a string of N D-dimensional vectors. For example, suppose we have the following data of two-dimensional vectors: $\{(6,4), (4,3), (5,2), (8,7), \ldots\}$. How do we store these data on a tree? The corresponding tree is known as a quad-tree in the computer science literature [22]. To store these data, one imagines a $N \times N$ square. The first key $(6,4)$ is stored at the coordinate $(6,4)$ of this square and it forms the root of the tree. This root has now four branches corresponding to the four quadrants around the point $(6,4)$. Note immediately the analogy to a corresponding fragmentation problem. The storing of the first vector corresponds to fragmenting the original $N \times N$ square into 4 rectangles which join each other

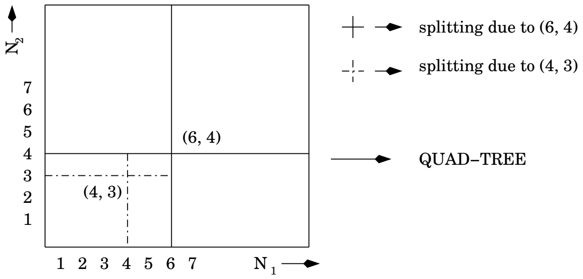

Figure 7. The storing of $(6,4)$ and $(4,3)$ on a quad tree → square fragmentation process.

as $(6,4)$ (see figure 7). This is the generalization of breaking a one-dimensional stick in the scalar case. Since both the components 6 and 4 are chosen independently and randomly from the set $\{1,2,3,\ldots,N\}$, this becomes a random fragmentation problem where the side lengths of any one of the 4 rectangles are chosen uniformly from the interval $[0-N]$.

The next element $(4,3)$ arrives and storing $(4,3)$ is equivalent to the fragmentation of the rectangle containing the new point $(4,3)$ into 4 further smaller rectangles. This process continues till all the data are stored, i.e., when the areas of all the rectangles become smaller than some threshold value $A_0 = 1$. One immediately sees the generalization to the case where each data element is a D-dimensional tuple. In the corresponding fragmentation problem, one starts with a D-dimensional cuboid of side lengths N and the arrival of each data correspond to fragmenting a cuboid into 2^D number of smaller cuboids. Note that $D = 1$ corresponds to the binary search tree of the scalar data, discussed before.

Following similar routes as in the m-ary search tree case, we were able to determine the exact asymptotic properties of the height H_N, the balanced height h_N and the number of non-empty nodes n_N of a D-dimensional quad tree. We just mention our main results here without providing details since they are similar to the earlier cases. For the extreme variables such as the height H_N and the balanced height h_N, we again find a travelling front structure whose analysis provides us with the following exact asymptotics for large N,

$$\langle H_N\rangle = 4.31107\ldots \log N - \frac{1.95303\ldots}{D}\log(D\log N) + \cdots,$$

$$\langle h_N\rangle \approx 0.373365\ldots \log N + \frac{0.89374\ldots}{D}\log(D\log N) + \cdots. \quad (24)$$

Surprisingly, the leading behavior (especially the coefficients of $\log(N)$ terms) turns out to be independent of the dimension D. Besides, due to the existence of a travelling front structure, one immediately finds that all the higher moments including the variance of H_N and h_N are bounded $\sim O(1)$ for large N.

For the number of nodes n_N, we again find a phase transition [15] driven by the same mechanism mentioned earlier. We find that while the average number

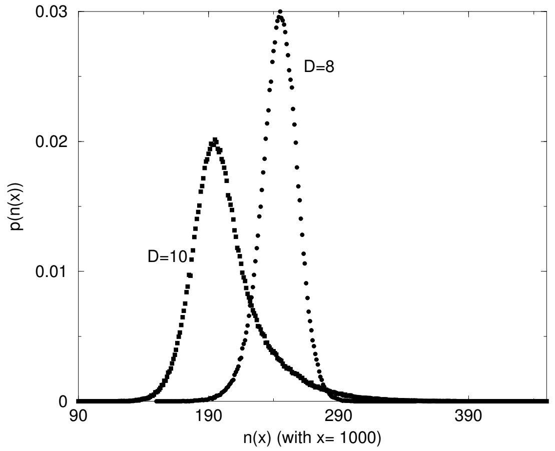

Figure 8. The distribution of the number of splittings of a cuboid with sidelength $N = x = 1000$ for $D = 8$ (filled circles) and for $D = 10$ (filled squares). The distribution is Gaussian for $D = 8$, but has a non-Gaussian skewness for $D = 10$. Note that the theoretically predicted critical dimension is $D_{\rm c} = 8.69363\ldots$. The histogram was formed by numerically splitting 5×10^5 samples in each case.

of non-empty nodes $\mu(N) = \langle n_N\rangle \approx 2V/D$ for large N where $V = N^D$, the variance $\nu(N) = \langle(n_N - \mu(N))^2\rangle$ undergoes a phase transition at a critical value of $D_{\rm c} = \pi/\sin^{-1}\left(1/\sqrt{8}\right) = 8.69363\ldots$,

$$\begin{aligned} \nu(N) &\sim V && \text{for} \quad D \le D_{\rm c} \\ &\sim V^{2\theta(D)} && \text{for} \quad D > D_{\rm c}, \end{aligned} \tag{25}$$

where $V = N^D$ and we computed the critical exponent $\theta(D) \ge 1/2$ exactly [15],

$$\theta(D) = 2\cos\left(\frac{2\pi}{D}\right) - 1 \tag{26}$$

which increases monotonically with D for $D > D_{\rm c}$. Furthermore, we computed numerically the full distribution of n_N for different values of D and found that while the distribution is Gaussian for $D < D_{\rm c}$ (a fact that can also be proved analytically), it becomes non-Gaussian for $D > D_{\rm c}$ (see figure 8). As before, once we write down the moment equations, D can be treated as a continuous parameter though in actual vector data D represent the dimension of a vector element and therefore D is always an integer.

5. Conclusion

In this paper, we have demonstrated how a variety of techniques developed in statistical physics can be successfully used to understand the statistical properties of various search trees, in particular for the random m-ary search tree problem. Search trees are the basic objects in data storage and retrieval. Hence we expect that our results will have important consequences in the 'sorting and searching' area of computer science. Our approach, perhaps not rigorous in the strict mathematical sense, has the advantage that it provides a physically transparent derivation of asymptotic results and can be readily generalized to study different types of search trees. For example, the travelling front method has subsequently been used to study the so-called 'digital search tree' that are used in the Lempel–Ziv data compression algorithm [16]. Besides, our approach has the beauty that it makes links between seemingly different problems and provides us with new results such as those for the vector data. We hope that the techniques discussed in this paper would be useful in future for studying other problems in computer science.

References

[1] D E Knuth, The art of computer programming, sorting and searching, 2nd ed. (Addison-Wesley, Reading, MA, 1998) vol. 3

[2] J M Robson, *Austr. Comput. J.* **11**, 151 (1979)

[3] P Flajolet and A Odlyzko, *J. Comput. System. Sci.* **25**, 171 (1982)

[4] L Devroye, *J. Assoc. Comput. Mach.* **33**, 489 (1986); *Acta Inform.* **24**, 277 (1987)

[5] H M Mahmoud, *Evolution of random search trees* (Wiley, New York, 1992)

[6] T Hattori and H Ochiai (unpublished).

[7] J M Robson, *Theor. Comput. Sci.* **276**, 435 (2002)

[8] M Drmota, *J. Assoc. Comput. Mach.* **50**, 333 (2003)

[9] H M Mahmoud and B Pittel, *J. Algorithms* **10**, 52 (1989)

[10] H-H Chern and H-K Hwang, *Random Struct. Algorithms* **19**, 316 (2001)

[11] B Chauvin and N Pouyanne, *Random Struct. Algorithms* **24**, 133 (2004)

[12] P L Krapivsky and S N Majumdar, *Phys. Rev. Lett.* **85**, 5492 (2000)

[13] S N Majumdar and P L Krapivsky, *Phys. Rev.* **E65**, 036127 (2002)

[14] E Ben-Naim, P L Krapivsky and S N Majumdar, *Phys. Rev.* **E64**, 035101(R) (2001)

[15] D S Dean and S N Majumdar, *J. Phys.* **A35**, L501 (2002)

[16] S N Majumdar, *Phys. Rev.* **E68**, 026103 (2003)

[17] R M Bradley and P N Strenski, *Phys. Rev.* **B31**, 4319 (1985)

[18] For a recent review see W van Saarloos, *Phys. Rep.* **386**, 29 (2003)

[19] S N Majumdar and P L Krapivsky, *Phys. Rev.* **E62**, 7735 (2000); *Phys. Rev.* **E63**, 045101 (R) (2001)
D S Dean and S N Majumdar, *Phys. Rev.* **E64**, 046 121 (2001)

[20] For a brief review, see S N Majumdar and P L Krapivsky, *Physica* **A318**, 161 (2003)

[21] B Reed, *J. Assoc. Comput. Mach.* **50**, 306 (2003)

[22] R A Finkel and J L Bentley, *Acta Inform.* **4**, 1 (1974)
P Flajolet, G Gonnet, C Puech and J M Robson, *Algorithmica* **10**, 473 (1993)