Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Institute of Theoretical Computer Science
Mohsen Ghaffari, Angelika Steger, David Steurer, Emo Welzl, Peter Widmayer

Solution 1: From Ancestors to Parents

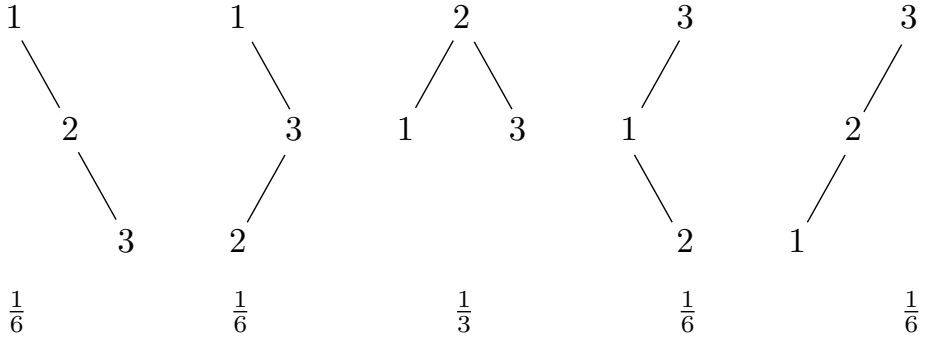

Figure 1: The 5 different randomized search trees on 3 nodes and the probability of their occurrence.

(a) We let $i = 2$ and $j = 1$. Clearly $\mathbf{E}[P_2^1(2)] = \frac{1}{2}$. From Figure 1 we observe that $\mathbf{E}[P_2^1(3)] = \frac{1}{6} \cdot 1 + \frac{1}{6} \cdot 1 = \frac{1}{3} \neq \mathbf{E}[P_2^1(2)]$. We choose further $i' = 2$ and $j' = 3$ and compute using Figure 1 that

$$\mathbf{E}[P_2^3(3)] = \frac{1}{6} \cdot 1 + \frac{1}{6} \cdot 1 = \frac{1}{3}, \text{ and}$$
$$\mathbf{E}[P_3^2(3)] = \frac{1}{6} \cdot 1 + \frac{1}{3} \cdot 1 = \frac{1}{2} \neq \mathbf{E}[P_2^3(3)].$$

(b) In the partial fraction decomposition we want to find constants A, B and C so that

$$\frac{1}{k(k-1)(k-2)} \equiv \frac{A}{k} + \frac{B}{k-1} + \frac{C}{k-2}.$$

Here $\equiv$ means that the equation should hold for all values of k for which the left side is defined. By multiplying both sides of the equation with $k(k-1)(k-2)$ we the equation

$$1 \equiv A(k-1)(k-2) + Bk(k-2) + Ck(k-1)$$
$$\Leftrightarrow 1 \equiv (A+B+C)k^2 + (-3A-2B-C)k + 2A$$

The left hand side is equal to $0k^2 + 0k + 1$ and since this equation has to hold for all possible values of k we get the system of equations

$$\begin{cases} 0 = A + B + C \\ 0 = -3A - 2B - C \\ 1 = 2A \end{cases}$$

which has the solution $(A, B, C) = (\frac{1}{2}, -1, \frac{1}{2})$. We can now compute the sum. For convenience define the harmonic number $H_0 := 0$. Then we get that

$$\begin{aligned}
\sum_{k=i}^{n} \frac{1}{k(k-1)(k-2)} &= \sum_{k=i}^{n} \left(\frac{1}{2} \cdot \frac{1}{k} - \frac{1}{k-1} + \frac{1}{2} \cdot \frac{1}{k-2} \right) \\
&= \frac{1}{2}(H_n - H_{i-1}) - (H_{n-1} - H_{i-2}) + \frac{1}{2}(H_{n-2} - H_{i-3}) \\
&= \left(\frac{1}{2} H_n - H_{n-1} + \frac{1}{2} H_{n-2} \right) - \left(\frac{1}{2} H_{i-1} - H_{i-2} + \frac{1}{2} H_{i-3} \right) \\
&= \frac{1}{2n} - \frac{1}{2(n-1)} - \frac{1}{2(i-1)} + \frac{1}{2(i-2)} \\
&= \frac{1}{2(i-1)(i-2)} - \frac{1}{2n(n-1)}.
\end{aligned}$$

(c) Notice that $Q_{1,n} = 0$ for every $n \in \mathbf{N}$. For $2 \leq i \leq n$ we compute $Q_{i,n}$ directly by conditioning on the rank of the root and we get that

$$\begin{aligned}
Q_{i,n} &= \sum_{k=1}^{n} \Pr(P_i^1(n) = 1 \mid \mathrm{rk}(\mathrm{root}) = k) \cdot \Pr(\mathrm{rk}(\mathrm{root}) = k) \\
&= \frac{1}{n} \sum_{k=1}^{n} \Pr(P_i^1(n) = 1 \mid \mathrm{rk}(\mathrm{root}) = k) \\
&= \frac{1}{n(n-1)} + \frac{1}{n} \sum_{k=2}^{n} \Pr(P_i^1(n) = 1 \mid \mathrm{rk}(\mathrm{root}) = k) \\
&= \frac{1}{n(n-1)} + \frac{1}{n} \sum_{k=i+1}^{n} \Pr(P_i^1(n) = 1 \mid \mathrm{rk}(\mathrm{root}) = k) \\
&= \frac{1}{n(n-1)} + \frac{1}{n} \sum_{k=i+1}^{n} Q_{i,k-1} \\
&= \frac{1}{n(n-1)} + \frac{1}{n} \sum_{k=i}^{n-1} Q_{i,k}.
\end{aligned}$$

The third equality above holds because in the case that 1 is the root, we have to choose i next and there are $n-1$ choices. The fourth equality follows because choosing any of the elements $\{2, \ldots, i\}$ as the root will either force 1 and i in different subtrees or put i before 1 into the tree and therefore the corresponding probabilities are 0. The last two equalities are just plugging in the definition and renaming the indices.

For $i = n$ the previous calculations give that $Q_{n,n} = \frac{1}{n(n-1)}$. For any i with $2 \leq i \leq n-1$ we compute in the usual way:

$$\begin{aligned}
& nQ_{i,n} - (n-1)Q_{i,n-1} = \frac{1}{(n-1)} - \frac{1}{(n-2)} + Q_{i,n-1} \\
\Leftrightarrow \quad & Q_{i,n} = Q_{i,n-1} - \frac{1}{n(n-1)(n-2)} \\
\Rightarrow \quad & Q_{i,n} = Q_{i,i} - \sum_{k=i+1}^{n} \frac{1}{k(k-1)(k-2)} = \frac{1}{i(i-1)} - \sum_{k=i+1}^{n} \frac{1}{k(k-1)(k-2)}.
\end{aligned}$$

Using part (b) we know the value of the sum and we get that

$$\begin{aligned} Q_{i,n} &= \frac{1}{i(i-1)} - \frac{1}{2i(i-1)} + \frac{1}{2n(n-1)} \\ &= \frac{1}{2i(i-1)} + \frac{1}{2n(n-1)} \end{aligned}$$

All in all we have that

$$\Pr(P_i^1(n) = 1) = Q_{i,n} = \begin{cases} 0 & \text{if } i = 1 \\ \frac{1}{2i(i-1)} + \frac{1}{2n(n-1)} & \text{if } i \in [n] \setminus \{1\} \end{cases} .$$

(d) This question makes sense because node 1 can only have one child. Using part (d) we can compute the probability that node 1 has a child

$$\begin{aligned} \Pr(\text{node 1 has a child}) &= \sum_{i=2}^{n} Q_{i,n} = \sum_{i=2}^{n} \frac{1}{2i(i-1)} + \sum_{i=2}^{n} \frac{1}{2n(n-1)} \\ &= \frac{1}{2} \sum_{i=2}^{n} \left(\frac{1}{i-1} - \frac{1}{i} \right) + \frac{1}{2n} \\ &= \frac{1}{2}(H_{n-1} - H_n + 1) + \frac{1}{2n} \\ &= \frac{1}{2}. \end{aligned}$$

There is also a simple argument for computing this probability by considering the permutations that lead into node 1 having a child (consider the relative order of 1 and 2). Now we can compute the expectation that was asked for as follows.

$$\begin{aligned} \mathbb{E}[\text{Child of } 1 \mid 1 \text{ has a child}] &= \sum_{i=2}^{n} i \cdot \Pr(P_i^1(n) = 1 \mid 1 \text{ has a child}) \\ &= \sum_{i=2}^{n} i \cdot \frac{\Pr(P_i^1(n) = 1 \wedge 1 \text{ has a child})}{\Pr(1 \text{ has a child})} \\ &= \sum_{i=2}^{n} i \cdot \frac{\Pr(P_i^1(n) = 1)}{\Pr(1 \text{ has a child})} \\ &= 2 \sum_{i=2}^{n} \frac{1}{2(i-1)} + 2 \sum_{i=2}^{n} \frac{i}{2n(n-1)} \\ &= H_{n-1} + \frac{\frac{1}{2}n(n+1) - 1}{n(n-1)} \\ &= H_{n-1} + \frac{n^2 + n - 2 + n - n}{2n(n-1)} \\ &= H_{n-1} + \frac{2(n-1) + n(n-1)}{2n(n-1)} \\ &= H_n + \frac{1}{2}. \end{aligned}$$

Solution 2: Sample and Factor

(a) $\Pr(j = i) = \Pr(c_n = c_{n-1} = \ldots = c_{i+1} = 0 \text{ and } c_i = 1) = \frac{n-1}{n} \cdot \frac{n-2}{n-1} \cdots \frac{i}{i+1} \cdot \frac{1}{i} = \frac{1}{n}$.

(b) Based on part (a) we can interpret the uniform generation of the sequence $(s_i)_{i=1}^k$ via the biased coins (although in the algorithm the sampling is still really done as in (a) and we use the coins only for the analysis). To generate s_1 in step 1 toss the coins $c_n, c_{n-1}, \ldots$ independently of each other until reaching some j so that $c_j = H$ and then set $s_1 = j$. To generate s_2 we would then (re)toss the coins $c_j, c_{j-1}, \ldots$ independently of each other and the previous tosses until reaching j' so that $c_{j'} = 1$ and then set $s_2 = j'$. It is possible that $j' = j$. We continue similarly and stop when reaching k so that $s_k = 1$.

Notice now that because in step 2 of the algorithm we only take those s_i's that are prime, we only care of the coin tosses c_p where p is prime. For the product of all the prime s_i's to be equal to t, when tossing the coin c_p we have to get α_p heads in a row followed by a one tails. Therefore

$$\Pr(r = t) = \prod_{p \le n} \left(\frac{1}{p}\right)^{\alpha_p} \frac{p-1}{p} = \frac{M_n}{t}$$

where the product is over all primes at most n.

(c) Fix some $t \in [n]$. The probability that $r = t$ after step 2 is equal to $\frac{M_n}{t}$ by part (b). Therefore in step 3 the probability that $r = t$ and that we output t is $\frac{M_n}{t} \cdot \frac{t}{n} = \frac{M_n}{n}$ which is independent of t. Therefore the output distribution is uniform. Because the probability is the same and independent for every number the probability that we output some number is $n \cdot \frac{M_n}{n} = M_n$.

(d) The expected number of primality tests is the expectation of the random variable

$$Y := \sum_{\ell=1}^{\infty} \sum_{i=1}^{n} X_{i,\ell}.$$

(e) In one iteration of the steps 1,2, and 3 we will test the primality of any number $i \in \{1, \ldots, n\}$ with probability $\frac{1}{i}$ because we have seen with the coin interpretation that this is the probability that at least one s_i is equal to i. Let $(I_\ell)_{\ell=1}^{\infty}$ be indicator variables for the event "there is an ℓ'th iteration through the steps 1,2 and 3" in a run of the algorithm SampleWithFactors(n). The expected number of primality tests is then

$$\begin{aligned}
\mathbf{E}[Y] &= \mathbf{E}\left[\sum_{\ell=1}^{\infty} \sum_{i=1}^{n} X_{i,\ell}\right] \\
&= \sum_{\ell=1}^{\infty} \sum_{i=1}^{n} \mathbf{E}[X_{i,\ell}] \quad \text{(Formally we are using the monotone convergence theorem here)} \\
&= \sum_{\ell=1}^{\infty} \sum_{i=1}^{n} \Pr(X_{i,\ell} = 1) \\
&= \sum_{\ell=1}^{\infty} \sum_{i=1}^{n} \Pr(X_{i,\ell} = 1 \mid I_\ell = 1)\Pr(I_\ell = 1) + \Pr(X_{i,\ell} = 1 \mid I_\ell = 0)\Pr(I_\ell = 0) \\
&= \sum_{\ell=1}^{\infty} \sum_{i=1}^{n} \Pr(X_{i,\ell} = 1 \mid I_\ell = 1)\Pr(I_\ell = 1)
\end{aligned}$$

$$= \sum_{\ell=1}^{\infty} \sum_{i=1}^{n} \frac{1}{i} \cdot (1 - M_n)^{\ell-1} = \frac{H_n}{M_n} = \Theta(\log^2 n).$$

Solution 3: Intersection patterns of polygons

We will use trapezoidal decomposition to solve the exercise. Let $E(Q)$ denote the edges in the polygon Q. We start by constructing a trapezoidal decomposition for the edges $E(Q)$. In the lecture notes on constructing a trapezoidal decomposition we assumed that the edges are in general position but in $E(Q)$ there are edges that share endpoints. As we saw in the exercises, with minor tweaks we can use essentially the same randomized incremental construction from Section 2.4 of the lecture notes to construct the trapezoidal decomposition for $E(Q)$ in expected $O(m \log m)$ time.

We can mark for every trapezoid whether it lies inside or outside the polygon Q. To do this orient the edges $E(Q)$ in counter-clockwise order along the boundary of Q in time $O(m)$. Then a trapezoid T is inside Q if the edge $e \in E(Q)$ that bounds T from above is oriented from right to left. If there is no edge above or if the edge is oriented from left to right, then T is outside. This marking takes linear time in m since the trapezoidal decomposition has size $O(m)$. Recall that we can maintain through the incremental construction for every trapezoid T a list of all trapezoids that share a vertical border with T. Let us denote this list by $L(T)$. Because no two vertices share an x-coordinate, this list has actually length at most 2 but we will argue without this additional information.

We are now ready for our algorithm. Because of the general position assumption on the vertices of the polygons, no point of P can be on the boundary of a trapezoid. Consider some point of P, say p_1. Do a point location in the trapezoidal decomposition to find the trapezoid that contains p_1. This takes $O(\log m)$ time in expectation. Starting from p_1 we walk along ∂P in counterclockwise order and record the sequence of trapezoids that intersect the boundary of P. More concretely, for $i = 1, \ldots, n$ assume we have already located the trapezoid T_i that contains p_i and we want to walk along the edge $e_i := \{p_i, p_{i+1}\}$ and record all trapezoids that intersect that edge (we define $p_{n+1} := p_1$). For doing this first check if T_i also contains p_{i+1} and if yes, we don't need to check other trapezoids since all trapezoids are convex. If T_i does not contain p_{i+1} we first check if the line segment $\overline{p_i p_{i+1}}$ crosses one of the at most two segments from $E(Q)$ that bound T_i. If yes, we stop and output the intersection case (d). If no, we linearly check through the list $L(T_i)$ and find the unique neighbor of T_i that is intersected by the line segment $\overline{p_i p_{i+1}}$. This takes time comparable to the length of $L(T_i)$. We then continue the same process with the newly found trapezoid until reaching the trapezoid that contains p_{i+1}.

At first it might seem that the above process can have very bad complexity but we claim that it takes time $O(m+n)$. The key insight is that during our walk on ∂P we can enter (and exit) any trapezoid T at most 4 times. Indeed, because P is convex, the boundary ∂P can intersect any of the at most 4 segments bounding T at most 2 times. Therefore we search through the neighborhood list of any trapezoid at most a constant number of times (because we made sure to always check first whether the currently processed edge exits the trapezoid). The combined length of all the neighborhood lists is at most 2 times the size of the trapezoidal decomposition which is $O(m)$. Therefore the walking takes time $O(m+n)$.

If the input is (d) then we indeed detect this at some intersection of the edges in P and Q and the above procedure already outputted the correct answer. Otherwise we get a sequence S of trapezoids intersected by ∂P. Recall that we augmented the trapezoids with the information of whether they are inside or outside Q. Then either (i) all trapezoids in S are outside of Q or (ii) all trapezoids in S are inside of Q since the last option of having both types of trapezoids

corresponds to intersection case (d). If (i) happens we know that we are either in intersection case (a) or (b). We can distinguish between them by taking a vertex of Q and checking whether it is inside or outside of P. This checking can be done in time $O(n)$ as we have seen in Section 2.1 of the lecture (the sorting of vertices of a convex polygon by x-coordinate can be done in linear time with a sweep, given the counter-clockwise order). If (ii) happens we are in the intersection case (c).

Therefore our algorithm found the correct intersection pattern and can output the right answer. The total time used is expected $O(m \log m + m + n) = O(m \log m + n)$ as required.