UNIVERSITÀ DI PISA

Second hands-on: Depth of a node in a Random Search Tree

Algorithm Design (2021/2022)

Gabriele Pappalardo
Email: g.pappalardo4@studenti.unipi.it
Department of Computer Science

March 2022

1 Introduction

A **random binary search tree** for a set S can be defined as follows: if S is empty, then the null tree is a random search tree; otherwise, choose uniformly at random a key $k \in S$: the random search tree is obtained by picking k as *root*, and the random search trees on $L = \{x \in S : x < k\}$ and $R = \{x \in S : x > k\}$ become, respectively, the left and right subtrees of the root k.

Consider the *Randomized Quick Sort* (RQS) discussed in class and analyzed with indicator variables, and observe that the random selection of the pivots follows the above process, thus producing a random search tree of n nodes.

- Using a variation of the analysis with indicator variables X_{ij}, prove that the expected depth of a node (i.e. the random variable representing the distance of the node from the root) is nearly $2 \log n$.
- Prove that the expected size of its subtree is nearly $2 \log n$ too, observing that it is a simple variation of the previous analysis.
- Prove that the probability that the depth of a node exceeds $c2 \log n$ is small for any given constant $c > 2$.

2 Solution

The hands-on's solution relies on the analysis we did during the lectures with randomized version of Quick-Sort. Let us recall for a moment how the algorithm works.

```
def randomizedPartition(A: [int], p: int, r: int) → int:
    i = random.randint(p, r)
    swap(A[r], A[i])
    return partition(A, p, r)

def randomizedQuickSort(A: [int], p: int, r: int):
    if p < r:
        q = randomizedPartition(A, p, r)
        randomizedQuickSort(A, p, q - 1)
        randomizedQuickSort(A, q + 1, r)
```

Listing 1: ‘Randomized QuickSort’

At the first sight of the code it seems that an RBST and the RQS do have nothing in common. Actually, the way how the RBST is build match 1-to-1 to the recursion tree created by the RQS.

When we select a random key $k \in S$ during the building of a RBST (the root of our tree), it corresponds to select a pivot in RQS. And the partition call allows us to create two subtrees (L, R) that will have elements less/greater than pivot, and then we call the tree build construction algorithm onto the subtrees.

2.1 Expected depth of a node

We define the elements of a random binary search tree with $\forall i \in [1, n].x_i \in S$. Let X_{ij} an indicator random variable defined as follows:

$$X_{ij} = \begin{cases} 1 & x_i \,\text{is an ancestor of}\, x_j \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

The depth of a node can be expressed as the sum of all the indicator random variables. Let D_i be the depth of a node x_i in a random binary search tree, defined as $D_i = \sum_{j=1}^{n} X_{ij}$. We can express the expected depth as:

$$E[D_i] = E\left[\sum_{j=1}^{n} X_{ij}\right] = \sum_{j=1}^{n} E[X_{ij}] = \sum_{j=1}^{n} \Pr(\{X_{ij} = 1\}) \tag{2}$$

The probability of $X_{ij} = 1$ can be computed using the analysis we did during lectures with RQS. Since the RQS maps over the RBST we can say that $X_{ij} = 1$ when x_i is a pivot (select as root for a tree) or x_j is a pivot.

The expected depth of a node can be computed using this fact:

$$E[D_i] = \sum_{j=0}^{n} \Pr(\{X_{ij} = 1\}) =$$
$$\sum_{j=0}^{i} \frac{1}{i-j+1} + \sum_{j=i+1}^{n} \frac{1}{j-i+1} =$$
$$\sum_{k=1}^{i+1} \frac{1}{k} + \sum_{k=2}^{n-i+1} \frac{1}{k} \le \log n + \log n = 2\log n$$

2.2 Expected size of a subtree

The analysis for the expected size of a subtree is similar to the previous one. We change the meaning of our indicator random variable as:

$$X_{ij} = \begin{cases} 1 & x_i \,\text{is an descendant of}\, x_j \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

2.3 Probability for the depth of a node

Knowing that the expected depth of a node $E[D_i] \le 2\log n$, we can prove that probability that the depth of a node exceeds $c2\log n$ is small for any given constant $c \ge 2$.

$$\mathbf{Pr}(\{D_i \ge c2\log n\}) \le e^{-\frac{\lambda^2}{2\mu+\lambda}} \tag{4}$$

Knowing that $\mu = E[D_i] \le 2\log n$ we can derive λ:

$$\mu + \lambda = c2\log n$$
$$E[D_i] + \lambda = c2\log n$$
$$\lambda = c2\log n - E[D_i] \leq c2\log n - 2\log n = \log n(2c-2)$$

Therefore, the Chernoff Bound becomes:

$$\begin{aligned}
\mathbf{Pr}(\{D_i \geq c2\log n\}) &\leq e^{-\frac{(\log n(2c-2))^2}{2(2\log n)+\log n(2c-2)}} \\
&= e^{-\frac{(\log n)^2(2c-2)^2}{2(2\log n)+\log n(2c-2)}} \\
&= e^{-\frac{(\log n)^2(2c-2)^2}{\log n(2c+2)}} \\
&= e^{-\frac{\log n(2c-2)^2}{(2c+2)}} \\
&= n^{-\frac{(2c-2)^2}{(2c+2)}} \\
&= n^{-\frac{4c^2+4-8c}{2c+2}} \\
&= n^{-\frac{2c^2+2-4c}{c+1}} \\
&= \frac{1}{n^{\frac{2c^2+2-4c}{c+1}}}
\end{aligned}$$

At the end, the probability is small for any constant $c > 2$.