Information Retrieval: Assignment #3

Alhajssssras Algdairy
alhajras.algdairy@gmail.com, 4963555, aa382

University of Freiburg — May 17, 2021

Exercise 1

In order to prove this we need to make an example. We will introduce two random lists A and B, where A has k elements and B has n elements, where $k \leq n$

$$A = \{1, 2, 4, 8\} \tag{1}$$

$$B = \{0, \underline{1}, \underline{2}, 3, \underline{4}, 5, 6, 7, \underline{8}, 9, 10, 11, 12, 13\} \tag{2}$$

By applying the galloping-search algorithm to allocate $A[i]$ inside B, we note the follow.

When	di	$=$	1,	we	get	$1.O(1)$	$=$	1,	therefore	$log(1)$	$=$	0
When	di	$=$	2,	we	get	$2.O(1)$	$=$	2,	therefore	$log(2)$	$=$	1
When	di	$=$	4,	we	get	$3.O(1)$	$=$	3,	therefore	$log(4)$	$=$	2
When	di	$=$	8,	we	get	$4.O(1)$	$=$	4,	therefore	$log(8)$	$=$	3

As we can see there is a direct relationship between the complexity and the log of the di, therefore the big O, has the next relationship for each step of the comparison $O(di) = log(di) + 1$, this is only for one step, if we want to sum it up for each element in A, we get the follow.

$$O(k + \sum_{i=1}^{k} log(di)) \tag{3}$$

Exercise 2

We first introduce the Lagrange multipliers equation.

$$L(x_1, \dots, x_n, \lambda) := f(x_1, \dots, x_n) - \lambda g(x_1, \dots, x_n) \tag{4}$$

In this problem our input functions are:

$$Maximize = \sum_{i=1}^{k} log(di) \tag{5}$$

$$Constraint = \sum_{i=1}^{k} di \leq n \tag{6}$$

By substituting both equations 5 and 6 in 4 we get:

$$\begin{aligned}
L &= \sum_{i=1}^{k} log(di) - \lambda(\sum_{i=1}^{k} di - n) \\
L &= (log(d_1) + log(d_2)..log(d_k)) - \lambda((d_1 + d_2..d_k) - n) \\
\frac{\partial L}{\partial d_1} &= \frac{1}{d_1} - \lambda = 0, \lambda = \frac{1}{d_1} \\
\frac{\partial L}{\partial d_2} &= \frac{1}{d_2} - \lambda = 0, \lambda = \frac{1}{d_2} \\
\frac{\partial L}{\partial d_k} &= \frac{1}{d_k} - \lambda = 0, \lambda = \frac{1}{d_k} \\
L &= \sum_{i=1}^{k} log(di) - \lambda(\sum_{i=1}^{k} di - n) \\
\frac{1}{d_1} &= \frac{1}{d_2} = \frac{1}{d_k} \\
d_k &= \frac{n}{x}
\end{aligned} \tag{7}$$

The only condition where it makes the previous terms equal independent of x, is when all terms $d_i = \frac{n}{x}$ are evenly equal. This can be done when we take the list B and divide it into evenly steps where all d_i are equals.

Therefore, we get $d_i = \frac{n}{k}$, to distribute the smaller list inside the bigger list symmetrically.

From the Exercise 1, we have.

$$\begin{aligned}
&O(k + \sum_{i=1}^{k} log(di)) \\
@d_i = \frac{n}{k}, &O(k + k.log(\frac{n}{k})) \\
&O(k(1 + log(\frac{n}{k})))
\end{aligned} \tag{8}$$

The simpler bounds is not correct, when we think of an edge case, such as when the $d_i = 1$ for all elements, this means we get hit after each jump in the algorithm, without exponentially increasing. This case gives k number of jumps, however, when we use the algorithm with out the constant k we get:

$\sum_{i=1}^{k} log(d_i) = \sum_{i=1}^{k} log(1) = 0$

Hence adding the k fix this issue.

Exercise 3

In order to prove this we need to illustrate the next example. Considering a list with $n = 100$ elements and a smaller list with only $k = 2$, we know that $d_i = \frac{100}{2} = 50$, now with such a big deviation between the two elements. The Zipper search algorithm will start searching for the first element which will be found in the middle of the list at $B[50]$ this will take 50 comparisons which is eqaul to $\frac{n}{2}$, same procedure will be done to the second element, it will take the same steps, therefore the number of comparisons we have in total $\frac{n}{2} + \frac{n}{2} = n$, therefore it will have always O(n) which is a linear time. This case can be noticed for any arbitrary list as long as we have the d_i evenly distributed.

Considering the galloping search when we look at its complexity which is logarithmic and non linear as the Zipper in this case. This can be seen in $equation 8$. Therefore $O(log(n))$ is always better then $O(n)$