ECS 122A Homework 2

Hardy Jones
999397426
Professor Gysel
Fall 2014

1. (a) **function** SET-INTERSECTION(X, Y)
    $X' \leftarrow SORT(X)$
    $Y' \leftarrow SORT(Y)$
    $i \leftarrow 0$
    $j \leftarrow 0$
    $intersection \leftarrow$ EMPTY-SET
    while $i < LENGTH(X')$ AND $j < LENGTH(Y')$ **do**
        if $X'[i] < Y'[j]$ **then**
            $i \leftarrow i + 1$
        else if $X'[i] = Y'[j]$ **then**
            $intersection \leftarrow INSERT(intersection, X'[i])$
        else if $X'[i] > Y'[j]$ **then**
            $j \leftarrow j + 1$
        end if
    end while
    return intersection
end function

Assuming we have a sort function that runs in $O(n \lg n)$ time and an insertion function that runs in $O(1)$ time, this algorithm should take $O(n \lg n)$ time.

We first sort the two sets in $O(n \lg n)$ time.

When we're iterating over the two arrays, we take a maximum of $O(n)$ time as each iteration runs in $O(1)$ time. This is less than $O(n \lg n)$ time, so our upper bound has not changed.

Thus, we have an intersection algorithm that runs in $O(n \lg n)$ time.

(b) **function** SET-INTERSECTION(X, Y)
    $intersection \leftarrow$ EMPTY-SET
    $table \leftarrow$ HASH-TABLE
    for all elements in Y **do**
        hash each element into *table*
    end for
    for all elements in X **do**
        if SEARCH(table, element) **then**
            $intersection \leftarrow$ INSERT(intersection, element)
        end if
    end for
    return intersection

end function

We need a table of size k as this is the number of elements in the set Y. We should not need to worry about collision resolution as the set ensures that each element is distinct in Y, so no two elements should hash to the same location.

Assuming we have a hashing function that runs in $O(1)$ time and an insert function that runs in $O(1)$ time, this algorithm should run $O(n+k)$ time.

We need to iterate each element of $X = O(n)$ time, and each element of $Y = O(k)$. This combines to $O(n+k)$ time.

2. Using the IRV defined in Theorem 11.2 $X_{ij} = I\{h(k_i) = h(k_j)\}$.

   We have $Pr\{h(k_i) = h(k_j)\} = \frac{1}{m}$, so $\mathrm{E}[X_{ij}] = \frac{1}{m}$.

   So we have that the expected number of collisions is

$$\begin{aligned} E\left[\sum_{i=1}^{n}\sum_{j=1}^{n} X_{ij}\right] &= \sum_{i=1}^{n} E\left[\sum_{j=1}^{n} X_{ij}\right] \\ &= \sum_{i=1}^{n}\sum_{j=1}^{n} E[X_{ij}] \\ &= \sum_{i=1}^{n}\sum_{j=1}^{n} \frac{1}{m} \\ &= \sum_{i=1}^{n} \frac{n}{m} \\ &= \frac{n^2}{m} \end{aligned}$$

   So the expected number of collisions is $\frac{n^2}{m}$.

3. We can give a counter example with the following values:

i	1	2	3
p_i	5	22	36
$\frac{p_i}{i}$	5	11	12

   If we wanted to maximize the profit on a rod that is 4 inches long, the new algorithm would choose to cut the rod into two pieces: one of length 3 inches, and one of length 1 inch.

   This gives us $36 + 5 = 41$.

   However, the optimal solution is to cut the 4 inch rod into two equal length pieces 2 inches long.

   This gives us $22 + 22 = 44$.

   Since $44 > 41$, the new algorithm does not provide an optimal solution.