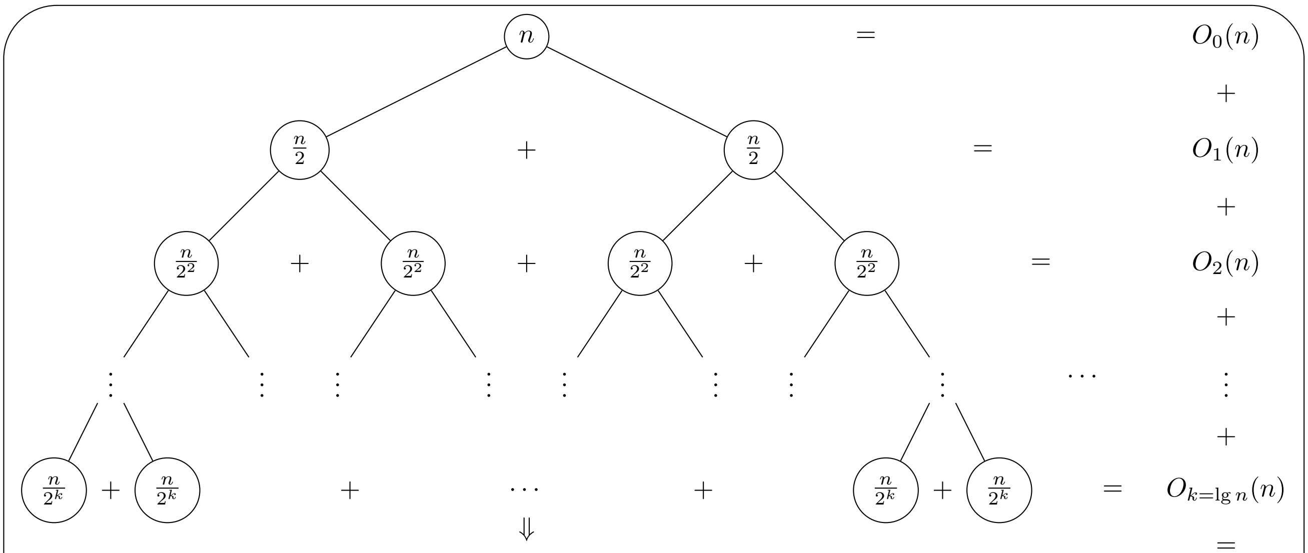
CSC263H1 Assignment 3

Jiatao Xiang, Xu Wang, Huakun Shen

February 14th, 2019

Question 1



$$O\left(\sum_{i=0}^{k} 2^i \cdot \frac{n}{2^i}\right) = O\left(\sum_{i=0}^{k} n\right) = O(k \cdot n) \Leftrightarrow O(n \cdot \lg n)$$

Question 2

Question 3

a. Our data structure is based on **hash table**.

Idea: Put every element of set **B** into a hash table. Then hash every element of set **A** to a slot in the hash table, and check whether the element of **A** is in this slot. If it is not in the slot, then it means that the element of set **A** is not in set **B**.

Pseudo Code:

Suppose $\boldsymbol{\alpha = 10}$, that is, each slot contains a linked list with size of at most approximately 10 elements.
Suppose we have a hash table $\boldsymbol{T}$ with a size of $\frac{n}{\alpha}$.
Suppose we have a hashing function ***h(x)*** that would return the index of one of the slots of **T** given $\boldsymbol{x}$ as a input. Also assume **SUHA**.

```
def h(x, num_slot):
  return x % num_slot
```

```
for element in B:
    linked_list = T[h(element, len(T))]
    linked_list.append(element)

result = []
for element in A:
    linked_list = T[h(element, len(T))]
    for item in linked_list:
        if element == item:
            break
```

b. Assumptions:

- The linked list in each slot of hash table has approximately a length of $\alpha = 10$
- Hash table **T** has a length of $\frac{len(A)}{\alpha}$
- SUHA for hash function ***h(x, num_slot)***

Explanation: Part I: put **B** into **T**

1. Hashing function costs constant time
2. There are n elements in **B**, performing hashing function ***h*** for each element of **B** costs $\mathcal{O}(n \times 1)$ of time.

Part II: match element of **A** to **T**

1. Each linked list in each slot of **T** has a size of at most α (by **SUHA**), which is constant. In the worst case, we have to traverse through every linked list, which costs $\mathcal{O}(\alpha) = \mathcal{O}(1)$ of time.
2. There are n elements in **A**, performing step 1 for each of them costs $\mathcal{O}(\alpha n) = \mathcal{O}(n)$ of time.

c.