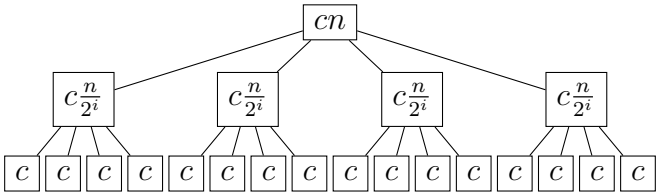
Problem Set 2

Name: Your Name

Collaborators: Name1, Name2

Problem 2-1.

(a) $T(n) = 4T(\frac{n}{2}) + O(n)$



$T(n) = \sum_{i=0}^{logn} 4^i \frac{n}{2^i} = n\sum_{i=0}^{logn} 2^i = n(2n-1) = O(n^2)$
In another hand, there are $4^{log_2 n} = n^2$ leaves. And $T(1) = \Theta(1)$, we deduce that $T(n) = \Omega(n^2)$. As consequence, $T(n) = \Theta(n^2)$

(b)

$$T(n) = 3T(\frac{n}{\sqrt{2}}) + O(n^4) = \sum_{i=0}^{\log_{\sqrt{2}} n} 3^i(\frac{n^4}{4^i}) \tag{1}$$

$$= 4n^4(1 - (\frac{3}{4})^{log_{\sqrt{2}} n+1}) = O(n^4) \tag{2}$$

(c)

$$T(n) = 2T(\frac{n}{2}) + 5nlogn = \sum_{i=0}^{logn} 2^i \frac{5nlogn}{2^i} \tag{3}$$

$$= 5nlogn(logn + 1) = O(nlog^2 n) \tag{4}$$

Problem 2-2.

(a) The problem requires the algorithm to be in-place, which excludes the merge sort. $D.set\_at(i, x)$ costs $\Theta(nlogn)$ and each swap operation calls $D.set\_at(i, x)$ twice, which is definitely inefficient. So we want to choose **the algorithm which needs fewer swaps**. And the answer for that **is selection sort**, which needs $\Theta(n)$ swaps at worst. In this case, $T(n) = O(n(n + nlogn)) = O(n^2logn)$. In contrast, insertion sort will perform $\Theta(n^2)$ times swaps in worst case, which leads $T(n) = O(n^3logn)$

(b) In this case, comparison is an expensive operation. As a result, we want to choose **the algorithm which needs fewer comparisons**. And the answer for that is **merge sort**. For selection sort and insertion sort, they need n^2 times comparisons. For merge sort, it needs $nlog_2n$ times comparisons.

(c) In this case, the array is basiclly sorted because even for $n = 10^9, loglogn \simeq 5$. In addition, swaps are adjacent. So the answer is absolutely insertion sort. In this case, $T(n) = \Theta(n)$, which is linear time.

Problem 2-3. Pass

Problem 2-4.

•Description: There are no more people to join in, so we can use a sorted-array that stores unique interger ID. Then, each person has three data structures: a time-list, a hash table and a dynamic-array. The time-list stores the unique ID by FIFO. And the key of hash table is unique ID, the value of hash table is the message set which is also ordered by FIFO. The dynamic-array records the index of people who receive the message.

•Time analysis: So $build(V)$ is $O(nlogn)$ for a sorted-array.

$send(v, m)$ is divided into two procedures:

1.Find the people according to ID, which costs $O(logn)$. And append the index into the back of dynamic-array, which costs $O(1)$

2.For people v: append the ID into the time-list and ID, message into the hash table, which are $O(1)$

$recent(k)$ according to time-list, return the message. Here maybe needs extra space to record how many times of a ID appears.

$ban(v)$: Find the people according to ID, which costs $O(logn)$. And use the index in dynamic-array to find the n_v people. Delete messages in hash table.

Solution: See the pdf, which maintains only two data structures. In fact, i misunderstood the recent function here. The real meaning of recent is to return the k messages been sent, not been received.

Problem 2-5.

(a)

(b)

(c) Submit your implementation to `alg.mit.edu`.