Problem Set 1

0pt

Name: Your Name

Collaborators: Name1, Name2

Problem 1-1.

(a) $f_1 = log(n^n) = n \cdot log(n)$ so $f_1 = \Theta(n \cdot log(n))$
$f_2 = (log(n))^n$ so $f_2 = \Theta((log(n))^n)$
$f_3 = log(n^{6006}) = 6006 \cdot log(n)$ so $f_3 = \Theta(log(n))$
$f_4 = (log(n))^{6006}$ so $f_4 = \Theta((log(n))^{6006})$
$f_5 = log(log(6006n))$ so $f_5 = \Theta(log(log(n)))$
Obviously, $f_2 > f_4 > f_3 > f_5$. For $f_1, \frac{f_1}{f_2} = \frac{n \cdot log(n)}{(log(n))^n} = \frac{n}{(log(n))^{n-1}}$ when n is large enough, $\frac{n}{(log(n))^{n-1}} = \frac{n}{(n-1) \cdot (log(n))^{n-2}} = ... = 0$ so $f_1 < f_2$. $\frac{f_1}{f_4} = \frac{n \cdot log(n)}{(log(n))^{6006}} = \frac{n}{(log(n))^{6005}} = \infty$ so $f_1 > f_4$.
so $f_2 > f_1 > f_4 > f_3 > f_5$. The answer is $(f_5, f_3, f_4, f_1, f_2)$

(b) $(\{f_1, f_2\}, f_5, \{f_3, f_4\})$

(c) $(\{f_2, f_5\}, f_4, f_1, f_3)$

(d) $(f_5, f_4, f_2, f_3, f_1)$

Problem 1-2.

(a) To reverse k items in D starting from index i, we can repeatedly delete each item at position i and re-insert it at position $i + k - 1$. Since each delete and insert costs $O(\log n)$ and we perform k such operations, the total running time is $O(k \log n)$.

(b) To move the k items in D starting from index i to be in front of the item at index j, we first delete the k items one by one from index i (each time the next item to be moved remains at position i). Then we re-insert them in order at position j (adjusting j to $j - k$ if $j > i$ after deletion). Both the deletion and insertion phases cost $O(k \log n)$, so the total running time is $O(k \log n)$.

Problem 1-3.

Data structure. Represent the binder as the concatenation $L \mid M \mid R$:

- L: pages strictly to the left of bookmark A (a deque implemented by a circular dynamic array).
- M: pages strictly between bookmarks A and B (deque).
- R: pages strictly to the right of bookmark B (deque).

Bookmarks are the boundaries: A sits between L and M; B sits between M and R. Let $\ell = |L|$, $m = |M|$, $r = |R|$, and $n = \ell + m + r$. Index i maps in $O(1)$ to

$$\text{page}(i) = \{ \ L \quad [i]0 \leq i < \ell, M[i-\ell]\ell \leq i < \ell + m, R[i-\ell-m]\ell + m \leq i < n.$$

Each deque supports amortized $O(1)$ push/pop at both ends and $O(1)$ random access; total space is $O(n)$.

Operations and running times.

- build(X): set $L \leftarrow [\,]$, $R \leftarrow [\,]$, $M \leftarrow X$ in order. Time: $O(|X|)$ worst-case.
- place_ mark(i, m) ($m \in \{A, B\}$ between page i and $i+1$): If $m = A$, adjust the boundary so that $|L| = i+1$ by moving items between L and M using pop/push at the appropriate ends; if $m = B$, make $|L| + |M| = i+1$ by moving between M and R. Time: $O(n)$ worst-case.
- read_ page(i): use the mapping above to read from L, M, or R. Time: $O(1)$ worst-case.
- shift_ mark(m, d) with $d \in \{-1, 1\}$: If $m = A$: $d = +1$ moves pop_ front$(M) \rightarrow$ push_ back(L); $d = -1$ moves pop_ back$(L) \rightarrow$ push_ front(M). If $m = B$: $d = +1$ moves pop_ front$(R) \rightarrow$ push_ back(M); $d = -1$ moves pop_ back$(M) \rightarrow$ push_ front(R). Time: $O(1)$ amortized.
- move_ page(m): If $m = A$, move the page just after A to just before B: pop_ front$(M) \rightarrow$ push_ back(M). If $m = B$, move the page just after B to just before A: pop_ front$(R) \rightarrow$ push_ back(L). Time: $O(1)$ amortized.

Why amortized $O(1)$? All edits touch only deque ends. Circular dynamic arrays resize by constant factors, so each element is moved only $O(1)$ times over its lifetime; rare $O(n)$ resizes are spread over $\Theta(n)$ cheap operations, yielding $O(1)$ amortized per end-operation.

operation	time
build(X)	$O(\|X\|)$ (worst)
place_mark(i,m)	$O(n)$ (worst)
read_page(i)	$O(1)$ (worst)
shift_mark(m,d)	$O(1)$ (amortized)
move_page(m)	$O(1)$ (amortized)

(a)

(b)

(c)

(d) Submit your implementation to alg.mit.edu.