David Bozarth
CES 512
Spring 2005

8.5

dfn k-sorted : (n-element A)
for each $i = 1, 2, \ldots, n-k$

ⓐ $$\frac{\sum_{j=i}^{i+k-1} A[j]}{k} \le \frac{\sum_{j=i+1}^{i+k} A[j]}{k}$$

ⓐ What does it mean for an array to be 1-sorted?
<u>Sorted</u>.

ⓑ Give a permutation of $1, 2, \ldots, 10$ that is 2-sorted, but not 1-sorted.

1, 3, 2, 4, 5, 6, 7, 8, 9, 10.

ⓒ Prove that an n-element array is k-sorted <u>iff</u> $A[i] \le A[i+k] \ \forall \ i = 1, 2, \ldots, n-k$.

($\Rightarrow$) Assume A has n elements and is k-sorted.
For each $i = 1, 2, \ldots, n-k$,

$$\frac{1}{k}\left(A[i] + A[i+1] + \cdots + A[i+k-1]\right) \le \frac{1}{k}\left(A[i+1] + A[i+2] + \cdots + A[i+k]\right)$$

Since the definition implies $k > 0$, then algebra yields

$$A[i] \le A[i+k] \quad (\Rightarrow) \checkmark$$

($\Leftarrow$) Assume $A[i] \le A[i+k]$ for each $i = 1, 2, \ldots, n-k$.

By reversing the algebra in ($\Rightarrow$), we see that A must be k-sorted.

The proof is complete. #

D. Bozarth

8-5.d

Give an algorithm that k-sorts an n-element array in $\Theta(n \lg(\frac{n}{k}))$ time.

<u>KSORT</u> (start, n, k, A) returns void

: if n < 1 then RETURN

for i = start to n

if A[i+k] < A[i] then

SWAP (i, i+k, A)

KSORT (i - 2*k, i-1, k, A)

: RETURN

The for-loop executes n times.

Each execution of the recursive call represents a decision tree with height bounded from above by $\lg(\frac{n}{k})$.

Therefore the algorithm k-sorts in $\Theta(n \lg(\frac{n}{k}))$.

Preliminary

D. Bozarth
CES 512
Spring 2005

Ex 6.5-8

there was an exam question like this also.

Merge k sorted lists into one sorted list A in time $\Theta(n \lg k)$, where n is total # elements.

: Let H be a min-heap, empty. Let A be empty.
Let $\{d_1, d_2, \ldots d_k\}$ be the set of lists.

while some list is not empty

for i = 1 to k
  if d_k not empty then
    a = (least element of d_k)

    insert (a, H)

for i = 1 to n

A[i] = extract Min (H)

: RETURN

D. Bozarth

$\boxed{\text{Ex. 6.5-8 cont'd}}$

Running time:

Insert: $\sum_{i=1}^{n/k} (\lg(i\,k)) = \sum_{i=1}^{n/k} \lg i + \sum_{i=1}^{n/k} \lg k$

Adding groups of k elements each to a heap.
The number of groups is bounded by $\lceil \frac{n}{k} \rceil$.
Inserting to the ith heap takes time $\lg i$.

$$\leq \int_1^{n/k+1} \lg t \, dt + \frac{n}{k} \lg k$$

$$= C + \frac{n}{k} \lg k \Rightarrow \underline{\Theta(n \lg k)}$$

Extract Min:

$$\sum_{i=1}^{n} \lg i \leq \int_1^{n+1} \lg t \, dt = C$$

So the running time depends on Insert

$$\Rightarrow \boxed{\Theta(n \lg k)}$$

D. Bozarth

8-5.e

Show that a k-sorted array of length n can be sorted in $\Theta(n \lg k)$ time.

Suppose $m = 0, 1, 2, \ldots, \frac{n}{k} - 1$.

Then the elements $i + mk$ for $i = 1, 2, \ldots, k$ represent k sorted lists, with total number of elements n.

By 6.5-8, these k lists can be collated in $\Theta(n \lg k)$ time.

#

8.5 f

Show that with constant k, to k-sort an n-element array requires $\Omega(n \lg n)$ time.

* Consider a two-step process of k-sorting the array, which can be done in time $\Theta(n \lg \frac{n}{k})$; then collating the k-sorted array, which can be done in time $\Theta(n \lg k)$.
* This is a comparison sort, which takes $\Omega(n \lg n)$. Since k is constant, the running time of the k-sort is $\Theta(n \lg n - n \lg k) = \Theta(n \lg n - n\alpha)$ $= \Theta(n \lg n)$

  The running time of the 2nd step is $\Theta(n\alpha) = \Theta(n)$.
* Thus the second step is insignificant, and the running time of the first step is $\Omega(n \lg n)$. #