CSL 630 Lecture 5, Aug 7

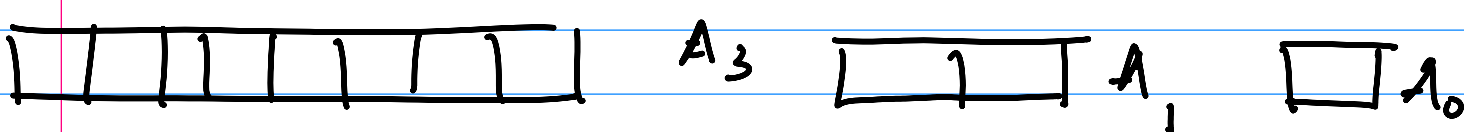
Array based data structure that supports

1. search
2. insertion

For any given n, we represent the n elements using a set of arrays A_i where $|A_i| = 2^i$ such that $\sum |A_i| = n$

For eg. $n = 11$ then we have

$A_3 \quad A_1, \quad A_0$



The elements in A_i are sorted (but have no relation with A_j $j \neq i$

Given any n, we can determine A_i's from the binary rep of n, similar to Binomial heaps

$n = 12 \qquad A_3 \qquad A_2$

Search: for a given key x, we will do binary searches in each of the arrays

Time : in A_i we take : $O(i)$

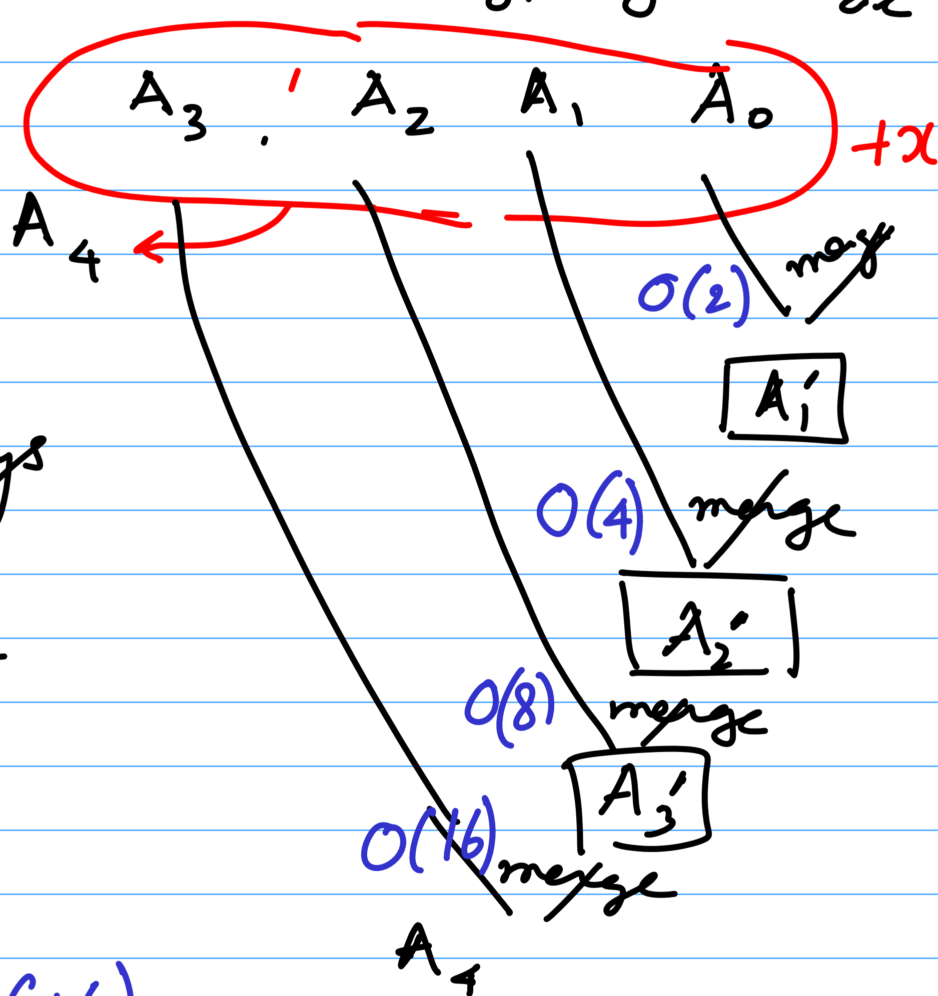
Maxm. time : $\sum_{i=1}^{\log n} i \le O(\log^2 n)$

Insertion : $n \to n+1$

$A_{i_1} \; A_{i_2} \cdots A_{i_k}$ $\qquad$ $A_{j_1} \; A_{j_2} \cdots A_{j_\ell}$

$n = 15$

$n = 16$



Time to merge two sorted arrays S_1, S_2 with $|S_1| = n_1$, $|S_2| = n_2$

$= O(n_1 + n_2)$

$\le c \cdot (n_1 + n_2)$

$= O(16)$

In general, if we have done this over j stages, then cost will be $O(2^{j+1})$

"Worst case" : $n = 2^j$
(all arrays were replaced)

Total cost is $O(n)$

How often will this happen?

How often will A_i be rebuilt?

Once every 2^i insertions

Cost of rebuilding A_i : $O(2^i)$

For inserting n elements, # rebuilds $= \frac{n}{2^i}$

Total cost of inserting n elements in all the arrays : $\sum_{i=0}^{\log n} \frac{n}{2^i} \times O(2^i)$

$\leq O(n \cdot \log n)$

Average cost : $O(\log n)$

Stacks : ① insertion, push $O(1)$

② deletion pop $O(1)$

③ Empty stack $O(\#\text{elements})$
pops all elements

Consider a sequence of push, pops and Empty Stack operations : n of them

What is the total cost ?

$\Rightarrow O(n^2)$

Consider a function

$$\phi : D \longrightarrow \mathbb{Z}$$ potential function

data structure

Amortised cost of an operation O, relevant to D = actual cost + change in potential

Eg: Suppose for the stack, we define

$$\phi(S) = (\#\text{ elements in } S)^2$$

Amortised cost of pop : 1 (actual cost) $+ 10^2 - 11^2$
if S had 10 elements

Total Amortised cost of a sequence of operations $O_1, O_2, O_3 \ldots O_n$

$$\sum_i \Big(T(O_i) + [\phi_{i+1} - \phi_i] \Big)$$

actual cost

change in potential after i operat.

$$= \sum_i T(O_i) + \phi_{n+1} - \cancel{\phi_n} + \cancel{\phi_n} - \cancel{\phi_{n-1}} + \cdots \phi_0$$

initial potential

$$+ \phi_{n+1} - \phi_0$$

final pot.

initial pot

Total actual cost : $T(n)$

" amortised " : $A(n)$

$$T(n) = A(n) + \phi_0 - \phi_{n+1}$$

If $\phi_{n+1} - \phi_0 \geq 0$ then

$$T(n) \leq A(n)$$

Example: For the case of stacks, let us define

$\phi(S)$ = # of elements in stack

$\phi(\text{empty stack}) = 0$ $\qquad \phi_k - \phi_0 \geq 0$

$A(\text{push}) = 1 + 1 = 2$

$A(\text{pop}) = 1 + (-1) = 0$

$A(\text{empty stack opn}) = K - (K) = 0$

The maxm amortised cost of any opn = 2, so total amortised cost of n operations $\leq 2n$

h.w.: Try to come up with an appropriate potential function for the array based search data structure

<u>Problem</u>: Given n strings over some finite alphabet Σ, we want to arrange them in lexicographic order.

String $s_1, s_2, \ldots s_n$
. have lengths $l_1, l_2, \ldots l_n$

$$\sum_{i=1}^{n} l_i = N$$

Spl case: all l_i's are equal
Run radix sort