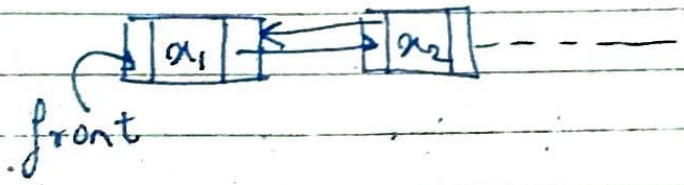
SOLO

Assignment-3
Theory (CS202)

Anukool Dwivedi
B19071
Grp-13

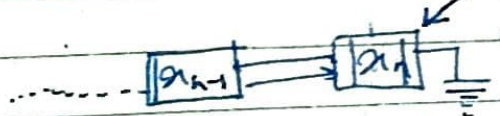
(1) The doubly linked list data structure is used here.

(i) We will maintain a front pointer.



For insertion;
Create a new node, named t.
t → data = key;
t → next = front;
~~front = t~~ front → prev = t;
front = t;

(ii) We will maintain a end pointer.



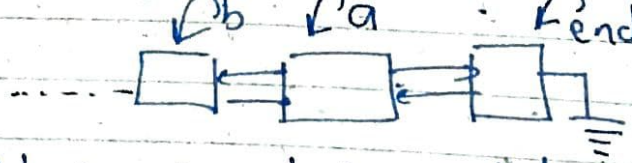
For insertion at end;
create a new node, named t;
t → data = key;
t → next = NULL;
t → prev = end;
end → next = t;
end = t;

(iii) We will maintain a middle pointer.
if insertion at beginning or deletion at end happen, then
middle = middle → prev.
if insertion at end, or deletion at beginning happen, then
middle = middle → next.
In this way, middle will adjust its place.

(iv) Now, comes the insertion at median. Let us consider that element is inserted after ~~previous m~~ middle
then, create node t;
~~t =~~ t → data = key;
t → next = middle → next;
t → prev = middle;
middle → next = t;
Now, middle = middle → next;

~~Similarly if inserted~~

(iv)



Now, we need to delete a;
So; a = end → prev;
b = a → prev;
b → next = end;
end → prev = b;
delete a;

② If we divide the input into groups of 7, then after partitioning the input array with median-of-medians, say m, as the pivot element, we can get a lower bound on the number of elements that are greater than m, and a lower bound on the number of elements smaller than m as follows. So, the number of elements greater than m ~~at~~ is at least :-

$$4 \times \left[\left\lceil \frac{1}{2} \left\lceil \frac{n}{7} \right\rceil \right\rceil - 2 \right] \geq \frac{2n}{7} - 8.$$

Similarly the elements that are smaller than m is also at least $\boxed{\frac{2n}{7} - 8}$.

Now, the recurrence relation for the runtime becomes :-

$$T(n) \leq T\left(\left\lceil \frac{n}{7} \right\rceil\right) + T\left(\frac{5n}{7} + 8\right) + O(n) \quad \text{---(1)}$$

We want to verify if the above recurrence has the soln that $\boxed{T(n) = c \cdot n}$ for some constant $c > 0$.

Assume, $T(n) \leq cn$ for some $c > 0$;

$$T(n) \leq T\left(\left\lceil \frac{n}{7} \right\rceil\right) + T\left(\frac{5n}{7} + 8\right) + a \cdot n$$

$$\leq c\left\lceil \frac{n}{7} \right\rceil + c\left(\frac{5n}{7} + 8\right) + a \cdot n$$

$$\leq \frac{cn}{7} + c\left(\frac{5n}{7} + 8\right) + an$$

The RHS is at most cn iff :-

$$\Rightarrow \frac{cn}{7} + c\left(\frac{5n}{7} + 8\right) + an \leq cn$$

$$\Rightarrow 8c + \frac{5nc}{7} + an \leq cn$$

$$\Rightarrow 8c + an \leq \frac{cn}{7}$$

$$\Rightarrow an \leq c\left(\frac{n}{7} - 8\right)$$

$$\Rightarrow c \geq \frac{a \cdot n}{\frac{n}{7} - 8} = \boxed{\frac{7an}{n - 56}}$$

So, we must have $n > 56$, then a const c exists such that $c > \frac{7a}{1 - \frac{56}{n}}$

Now, if we chose $n > 112$;

we have $1 - \frac{56}{n} > \frac{1}{2}$ and $c > 14a$.

So, the conditions for c can be satisfied, the SELECT algorithm still runs in linear time.

③ Now, here we have groups of size 3. Now, for this, the recurrence relation will be :-

$$T(n) = T\left(\lceil \frac{n}{3} \rceil\right) + T\left(\frac{4n}{6}\right) + O(n)$$

$$\geq T\left(\frac{n}{3}\right) + T\left(\frac{2n}{3}\right) + O(n)$$

Now, we will show it is $\geq cn \log(n)$.

$$T(n) \geq c\left(\frac{n}{3}\right)\log\left(\frac{n}{3}\right) + c\left(\frac{2n}{3}\right)\log\left(\frac{2n}{3}\right) + O(n)$$

$$T(n) \geq \boxed{cn\log(n) + O(n)}$$

∴ we have that it grows more quickly than **linear**.

④ In this problem, we have to find the $\sqrt{n}$ th smallest element in linear time.
Also, we have proved that min heap could be build from given n elements in $O(n)$ time.
(Proved in assignment 2).

So, we will call the extractMin() $\sqrt{n}$ times, and then we get $\sqrt{n}$ th smallest element.

The time complexity will be

$$O(n + \sqrt{n} \log n) \approx \boxed{O(n)}$$

as $\lim_{n \to \infty} \frac{\sqrt{n} \log n}{n} = \boxed{\lim_{n \to \infty} \frac{\log n}{\sqrt{n}} \Rightarrow 0}$

So; $O(n + \sqrt{n} \log n) \leq O(n + n)$

$\Rightarrow O(2n) \approx \boxed{O(n)}$

Hence, we could get the $\sqrt{n}$ smallest element $O(n)$ time.

(5) We have a black box which finds median in O(n) time.
The algorithm is as follows:-

(i) Find median of an array of n elements. Time taken O(n).
(ii) Then divide the array into 2 parts (or sets) A and B.
A is on left side of median.
B is on right side of median.
All the elements of A are less than median and all the elements of B are greater than median.
(iii) Then, the above step takes n-1 comparisons.
(iv) Recursively call on both A and B.
(v) In this way, the array is sorted.

$$T(n) = 2T\left(\frac{n}{2}\right) + O(n)$$

And, solution to this is $O(n \log n)$.