Building Cartesian trees in linear time	
Given an arry A = (ATI],, ATU])	
Cartestan tree of A	
1. binary tree with hodes (i, Ati])	
2. Search free according to i	
3. min-heap according to ATi]	
	• • • • • • • • • • • • •
5261134	
Algorithm	
1. Add dummy - so to the felt, to the night of the array	
2. Turn averagents a linked list	
()-> (Ati)-> (Atz)-> (Atm)-> (->)	
3. As long as there is a local maximum "in list link (X).	· · · · · · · · · · · · · · · · · · ·
$link(\times)$.	
$X \mapsto (acl MM: X > M)$	
×.72	
linh (x): make x child of Max (M1Z)	
(INC (X). IN C X COULD UP II () II-	× dropped out of Revited list
· if y > 2, make × right child of y	eviled list
$\cdots \mathbb{Q} \mathbb{E} \cdots$	operation tales O(1)
	operat (1) to be time
· il Z7Y, muche × left child of 2	
$ \rightarrow $	Lafter M-1 lul oprictions

after min link operations lat: Or OF O hat: Or OF O how to find load index?
(2) (where the first item is is contention fried is constant left not item is is constant left not item is while is not at the end - if lad wire at i, the live (i) - else is it is not - left of is no lod max - every iteration makes progress's - either cut one node (reduces litt site by 1) example:
t i start at left wat iten i while i not at the end - if lace wix at i, then link (i) - else it i. went - left of i no loce max - every iteration mules progress: - either cut out one node (reduces lott site by 1) example:
t i start at left wat iten i while i not at the end - if lace wix at i, then link (i) - else it i. went - left of i no loce max - every iteration mules progress: - either cut out one node (reduces lott site by 1) example:
t i start at left wat iten i while i not at the end - if lace wix at i, then link (i) - else it i. went - left of i no loce max - every iteration mules progress: - either cut out one node (reduces lott site by 1) example:
- if local max at i, then link(i) - else i ← i. next - left of i no lock max - every iteration makes progress: - either cut out one node (reduces list site by 1) example:
- if load max at i, then link (i) - else it i. next - left of i no load max - every iteration more progress: - either cut out one node (reduces list site by 1) example:
-left of i no lock man - every iteration modes progress: - either cut out one node (reduces list size by 1) example:
- every iteration more progress: - either cut out one node (reduces list size by 1) example:
LI I I I I I I I I I I I I I I I I I I
$-p \rightarrow 5 \rightarrow 2 \rightarrow 6 \rightarrow 1 \rightarrow 5 \rightarrow 4 \rightarrow -2^{n}$ $\uparrow \qquad \uparrow \qquad$
$-p \rightarrow 2 \rightarrow 6 \rightarrow 1 \rightarrow 3 \rightarrow 6 \rightarrow -\infty$
$\begin{array}{c} 5 \\ 2 \\ -\gamma \end{array} \xrightarrow{2} \\ \gamma \end{array}$
$-\gamma \rightarrow 1 \rightarrow 3 \rightarrow 4 \rightarrow -\gamma$
2 3 5 6

Correctness	· · · · · · · · · · · · · · · · · · ·	•
1. bin 2. sen 3. mi	ch tree (=) Cartena Nec	0 0 0 0 0 0
(1)	locilmax X. Y. Y. Y.	0 0 0
	x becomes left child of y	0
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~	need to show: y will not get another left child. This is true, because z <y, (as="" a="" as="" be="" can="" is="" local="" long="" max="" never="" so="" th="" there),<="" y="" z=""><th>0</th></y,>	0
• • • • • •	so it will not be linked to y. When y is gone, then it cannot get left children anyway.	•
		۰
So a .noc	e can only get one left child. By symmetric argument, only one right child.	0
So a .noc	e can only get one left child. By symmetric argument, only one right child.	0
	e can only get one left child. By symmetric argument, only one right child.	0 0 0
Z. Mai	ntain invariant: if x is left of y in the list, then all nodes in subtree(x) have smaller indices than all nodes in subtree(y).	0 0 0
(Ž.) Mai Thi A si	ntain invariant: if x is left of y in the list, then all nodes in subtree(x) have smaller indices than all nodes in subtree(y). s is true initially. ngle link preserves this, so always true.	0 0 0 0
(Ž.) Mai Thi A si	ntain invariant: if x is left of y in the list, then all nodes in subtree(x) have smaller indices than all nodes in subtree(y).	• • • •
(Ž.) Mai Thi A si	ntain invariant: if x is left of y in the list, then all nodes in subtree(x) have smaller indices than all nodes in subtree(y). s is true initially. ngle link preserves this, so always true.	0 0 0 0 0 0
(Ž.) Mai Thi A si	ntain invariant: if x is left of y in the list, then all nodes in subtree(x) have smaller indices than all nodes in subtree(y). s is true initially. ngle link preserves this, so always true.	0 0 0 0 0 0 0 0 0 0 0 0 0
(Ž.) Mai Thi A si	ntain invariant: if x is left of y in the list, then all nodes in subtree(x) have smaller indices than all nodes in subtree(y). s is true initially. ngle link preserves this, so always true.	
(Ž.) Mai Thi A si	ntain invariant: if x is left of y in the list, then all nodes in subtree(x) have smaller indices than all nodes in subtree(y). s is true initially. ngle link preserves this, so always true.	
(Ž.) Mai Thi A si	ntain invariant: if x is left of y in the list, then all nodes in subtree(x) have smaller indices than all nodes in subtree(y). s is true initially. ngle link preserves this, so always true.	
(Ž.) Mai Thi A si	ntain invariant: if x is left of y in the list, then all nodes in subtree(x) have smaller indices than all nodes in subtree(y). s is true initially. ngle link preserves this, so always true.	
(Ž.) Mai Thi A si	ntain invariant: if x is left of y in the list, then all nodes in subtree(x) have smaller indices than all nodes in subtree(y). s is true initially. ngle link preserves this, so always true.	
(Ž.) Mai Thi A si	ntain invariant: if x is left of y in the list, then all nodes in subtree(x) have smaller indices than all nodes in subtree(y). s is true initially. ngle link preserves this, so always true.	
(Ž.) Mai Thi A si	ntain invariant: if x is left of y in the list, then all nodes in subtree(x) have smaller indices than all nodes in subtree(y). s is true initially. ngle link preserves this, so always true.	
(Ž.) Mai Thi A si	ntain invariant: if x is left of y in the list, then all nodes in subtree(x) have smaller indices than all nodes in subtree(y). s is true initially. ngle link preserves this, so always true.	
(Ž.) Mai Thi A si	ntain invariant: if x is left of y in the list, then all nodes in subtree(x) have smaller indices than all nodes in subtree(y). s is true initially. ngle link preserves this, so always true.	
(Ž.) Mai Thi A si	ntain invariant: if x is left of y in the list, then all nodes in subtree(x) have smaller indices than all nodes in subtree(y). s is true initially. ngle link preserves this, so always true.	
(Ž.) Mai Thi A si	ntain invariant: if x is left of y in the list, then all nodes in subtree(x) have smaller indices than all nodes in subtree(y). s is true initially. ngle link preserves this, so always true.	
(Ž.) Mai Thi A si	ntain invariant: if x is left of y in the list, then all nodes in subtree(x) have smaller indices than all nodes in subtree(y). s is true initially. ngle link preserves this, so always true.	
(Ž.) Mai Thi A si	ntain invariant: if x is left of y in the list, then all nodes in subtree(x) have smaller indices than all nodes in subtree(y). s is true initially. ngle link preserves this, so always true.	
(Ž.) Mai Thi A si	ntain invariant: if x is left of y in the list, then all nodes in subtree(x) have smaller indices than all nodes in subtree(y). s is true initially. ngle link preserves this, so always true.	
(Ž.) Mai Thi A si	ntain invariant: if x is left of y in the list, then all nodes in subtree(x) have smaller indices than all nodes in subtree(y). s is true initially. ngle link preserves this, so always true.	
(Ž.) Mai Thi A si	ntain invariant: if x is left of y in the list, then all nodes in subtree(x) nave smaller indices than all nodes in subtree(y). s is true initially. ngle link preserves this, so always true.	