①

②

i: Insert in a singly linked list:

برای محاسبه O_{worst} باید در نظر بگیریم که ما متغیر tail را نداریم ← آدرس آخرین خانه

و حالا می‌خواهیم یک عنصر جدید را به آخر لیست اضافه کنیم ← باید دوتا عملیات را انجام دهیم: اول به اندازه $O(n)$ در آرایه search می‌کنیم تا آن position (index) که می‌خواهیم عنصر جدید را در آن قرار دهیم پیدا کنیم، و سپس با $O(1)$ نیز مقدار مورد نظر را insert می‌کنیم ← $O_{worst}(n)$

$$O_{amortized} = \frac{Cost(n\ operations)}{n}$$

$$= \frac{Cost(normal\ operations) + Cost(Expansive\ operations)}{n}$$

$$= \frac{O(1) + O(n)}{n} = \frac{O(n)}{n} = O(1)$$

Subject Year: Month: Day:

ii: Delete in a singly linked_list.

برای این حالت نیز بهترین حالت O(1) می شود اگر به tail یا head دقیقا روی همان position ی باشد که می خواهیم از آن حذف کنیم اما در غیر اینصورت لیست الزام است index مورد نظر را پیدا کنیم.

$O_{worst}(n)$ ⟵ $O(n)$

$O_{amortized} = \frac{O(1)+O(n)}{n} = O(1)$

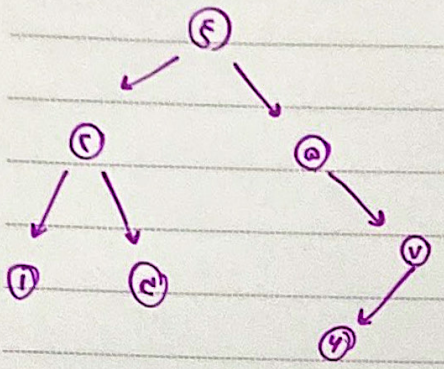
iii. Insert in a sorted arrays

در یک آرایه مرتب شده عملیات search را با استفاده از Binary Search انجام می دهیم و سپس عملیات insert را با استفاده از shift دادن عناصر انجام می دهیم

و در واقع در یک آرایه sort نشده عملیات insert کردن سریعتر هست چرا که لازم نیست به position عضوی که می خواهد جایگذاری شود توجه کنیم

در بدترین حالت اگر تمام عناصر لازم باشد shift پیدا کند ⟵ $O(n)$ worst

Subject Year: Month: Day:



(٤)

a) In order ⇒ ١, ٢, ٣, ٤, ٥, ٦, ٧

b) Pre order ⇒ ٤, ٢, ١, ٣, ٥, ٧, ٦

c) Post order ⇒ ١, ٣, ٢, ٦, ٧, ٥, ٤

Pre order ⇒ ① key
② left
③ right

Post order ⇒ ① left
② right
③ key

In order ⇒ ① left
② key
③ right

(٤)

without right or left child ⇒ -1

Post order ⇒ -1 -1 -1 ٨ -1 -1 ٦ ٣ -1 -1 -1 ٧ ٥ ٢ ١

right key r key
right key
key