تلخيص موضوع Search Algorithms هياكل البيانات

مهم جدا

متى استخدم Binary ومتى استخدم Sequential ؟ الـBinary مع البيانات الكثيرة والـSequential مع البيانات القليلة

Search Alg.

- Binary Search
- Sequential Search (Linear Search)

Sequential Search (linear search) :-

راح يكون عن طريق المقارنة.

لنفرض ان عندنا array بهذا الشكل :-

0	5	4	9	2	1

وبني نبحث عن 4 بأستخدام Sequential Search

راح يبدأ من اليسار يقارن

① -

0	5	4	9	2	1

↓

4 = 0 ؟ الاجابة لا

② -

0	5	4	9	2	1

↓

4 = 5 ؟ الاجابة لا

③

0	1	2	3	4	5
0	5	4	9	2	1

↓

4 = 4

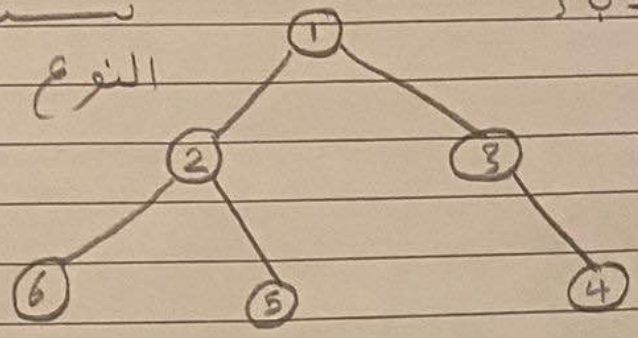
الاجابة نعم 4 = a[2]

١ / تكملة لمحاضرة trees هياكل البيانات

balanced binary tree :-

ان يكون معظم الاباء لهم ابناء

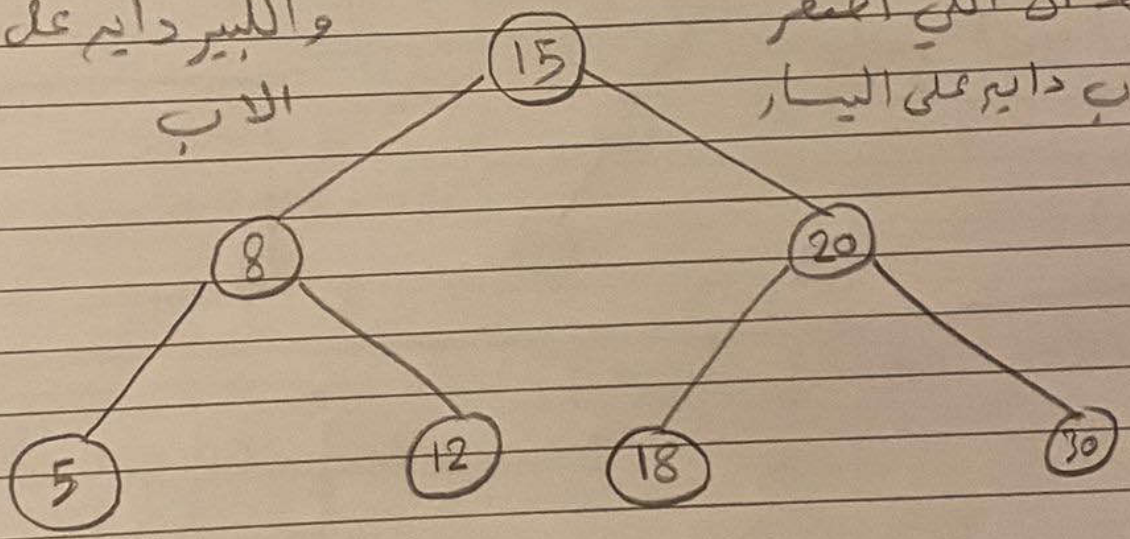
نستخدم مع هذا النوع double linked list



في حال كانت binary search tree لازم تكون مترتبة

نلاحظ ان اللي اصغر من الاب دايم على اليسار

والكبير دايم على يمين الاب



في حال بدنا اكتبها بصيغة array لازم تكون completed binary tree

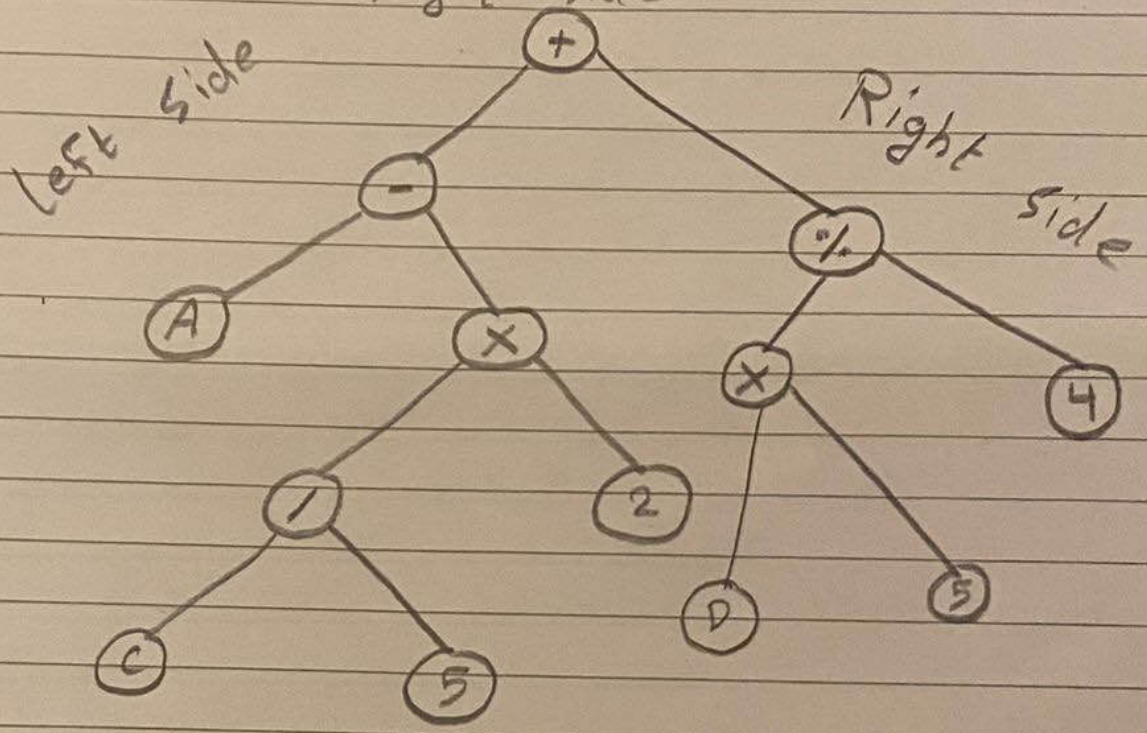
لان الحجز مسبق

trees اكمال لموضوع ال

Prase tree:-

A-(c/5 x2) +(D x5 %4)

left side — Right side



Left side

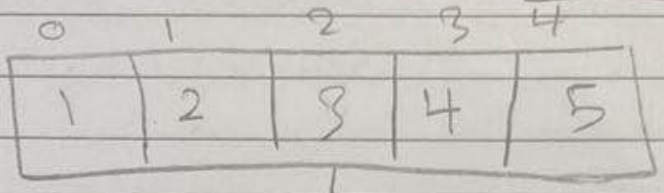
Right side

Search alg. تكملة لموضوع

Binary Search alg. :-

شرط اساسي: اول شي لازم يتحقق ان تكون البيانات مترتبة.

مثال توضيحي لآلية البحث:



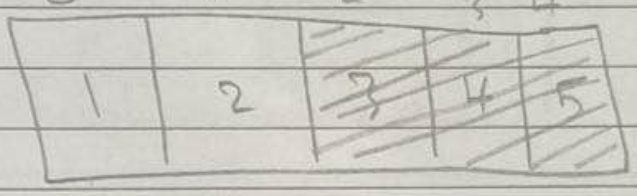
اول مقارنة تبدا من الوسط:

في حالتي بحث عن 2.

2 = 3 ? الاجابة لا

2 < 3

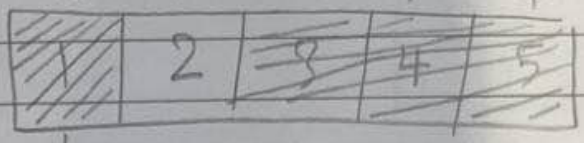
طالما ان 2 اصغر من الوسط فاراح يكنسل الجزء الايمن



الجزء المظلل مكنسل (ملغي)

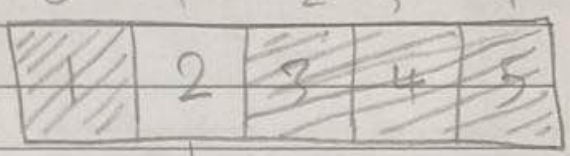
نحتاج نطلع الوسط راح نجمع العدد الاول والثاني ونقسمه على 2

الوسط = (الاخير + الاول) / 2



2 = 1 ? الاجابة لا

2 > 1



2 = 2 ✓

2 = a[1]

$O(n) = \log_2(n)$

تكملة لموضوع search algorithms هياكل البيانات

big O (n) ←
- best case ← افضل وضع
- avrage case ← الوضع المتوسط
- worst case ← اسوء وضع

بالنسبة للـ Sequential Search في افضل وضع best case

$O(n) = 1$ لية !

يعني مثلا عندنا هذي المصفوفة :

0	1	2	3	4
5	9	5	4	2

وحنا بي نبحث عن 5

فا راح يكون من المقارنة الاولى نحصل اللي نبحث عنة

$5 = 5$

$5 = a[0]$ في افضل حال $O(n) = 1$ عشان كذا يكون

بالنسبة لاسوء وضع كم بيكون الـ $O(n)$ ؟

اول شي لازم نعرف ان اسوء وضع ممكن يصير ان نبحث عن شي في الاخير او مو موجود اصلا

مثال توضيحي :

0	1	2	3	4
5	9	6	4	2

بي نبحث عن 2 : راح يقارن من اول واحد الى ان يوصل لـ 2 وهي في الاخير

او مثلا بي نبحث عن 2 : فا الجهاز بيسوي مقارنات الى الاخير ولا راح يحصل 2

فا الـ $O(n) = n$ worst case

حول المصفوفة $n =$

والمتوسط راح يكون $O(n) = n/2$ avrage case

هياكل البيانات

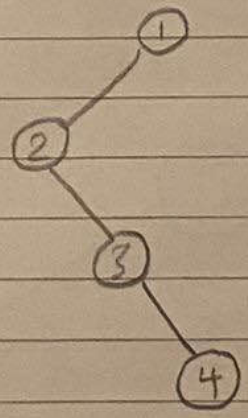
trees تلخيص موضوع الشجره

tree

- Prase tree
- binary tree
  - completed or balanced
    - يوجد شرط لل binary tree
      يكون لكل اب ابناء بالكثير
  - degenirated

degenirated :-

مع هذا النوع نستخدم linked list عادي

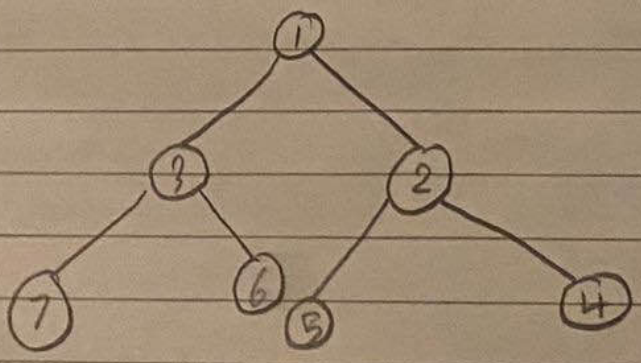


عندما يكون لكل أب ابن واحد فقط

يكون اسم ال tree degenirated

completed binary tree :-

يكون لكل اب ابناء بالقليل



مع هذا النوع نستخدم doube linked list

(أسئلة مقترحة)

Q1 When is not appropriate to use the data structure array? What is the major advantage of the array?

متى يكون استخدام الأراي سيئ؟ وما هي الميزة الرئيسية للـ Array.

is we have huge data. - Array element can be accessed randomly by use index value

إذا كنا نملك بيانات ضخمة
الأراي عندما نستطيع الوصول لها عشوائياً عن طريق index value

Q2 Give two resons why will use linked list over array data structure?

1 linked list is dynamic memory.
لأن اللينكد داينمكية

2 linked list take less time in insertion and deletation.
عمليات الحذف والإضافة
اللينكد تأخذ وقت قصير في

Q3 When linear search algorithms is better than binary search algorithms? Why?

When array can be in any order and not huge data because in sequantial the search element by element.

عندما تكون الأراي بأي ترتيب (غير مرتبة وقليلة)
لأن السكونشال يبحث عنصر بعنصر أو خطوة بخطوة.

مسائل هياكل

Q1 if items in a list are floats taking 8 memory locations each. compare the amount of space required altogether if

A list is kept contiguously in an array 90% full?

Solve:- I will assume my size 1000 50%.

$$\frac{8 \times 90}{100} = 7.2 \times 1000 = 7200 \text{ MI}$$

General size is $8 \times 1000 = 8000$ MI

B list is kept contiguously in an array 50% full?

Solve:- I will assume my size 1000 50%.

$$\frac{8 \times 50}{100} = 4 \times 1000 = 4000 \text{ MI}$$

General size is $8 \times 1000 = 8000$ MI.

C list is kept as a linked list (Pointer Take Two MI)?

In case 90% :- $10 \times 900 = 9000$ MI (linked list)
VS 8000 MI (array)

So 90% the array better than linked.

In case 50% :- $10 \times 500 = 5000$ MI (linked)
VS 8000 MI (array)

So in 50% the linked list better than array

information about linked list

array في cell أو إضافة	linked في Node عملية إضافة
called : insert	called: append

Q wiche best insert in Array or append in linked?

insert is best because The counter in array start from end.

but

append needs To comPirison evry Nodes To find node Null.

- متى تكون append أفضل في حالات استثنائية:
when append from The midle.

- Movement لأن الأوامر يتطلب عملية

end

مقارنة بين binary, sequential وعيوب ومميزات

Benefit (binary)

- more efficient than liner search (sequential)
  أكثر كفاءة من السكونشال .
- Take a lot of Data
  تأخذ الكثير من البيانات

Benefit (sequential)

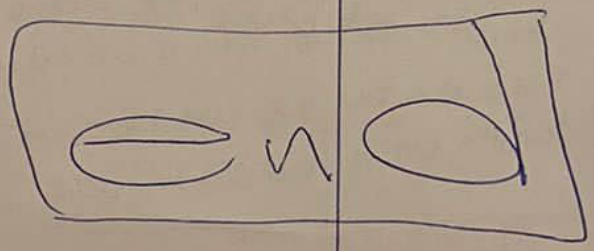
- easy algorithm to understand
  لسهولة الحفظ .
- Array can be in any order
  ممكن أن تكون بأي ترتيب

disadvanteg (binary)

- Requires that array elements be sorted
  لازم تكون البيانات مرتبة .

disadvantage (sequential)

- insicient (slow) is the data is huge.
  كفاءة بطيئة إذا كانت البيانات هائلة .



(أسئلة مقترحة)

Q4 متى يكون الإدخال في الأراي أفضل من الإدخال في اللينكد ؟
When The insert better Than append ?
عندما يكون الإدخال من آخر خلية
When The insert in end array . because The counter start from end cell .
لأن الإدخال في الأراي يبدأ من النهاية

Q5 When The append better Than insert ?
عندما يكون الإدخال من الوسط أو من البداية
when append from ~~[redacted]~~ medel or in head data .

Q6 Why We use Traverse ? and What is Travers ?
عملية المرور من بداية اللينكد إلى نهايتها (Null)
Traverse : The Process of Passing from begining ~~[redacted]~~ linked list To end Node (Null), When We need To know number of Node .
نستخدمها عندما نريد أن نعرف كم عدد النود في اللينكد .

Q7 هل يمكن الإدخال قبل العقد المطلوبة ؟ ولماذا ؟
is it Possible To append before Required ~~[redacted]~~ Node ?
why ? No We can't . because ~~[redacted]~~ The Pointer can't access To The Node in The back .
لا يمكننا
لأن البوينتر يصل العقدة التي أمامه فقط .

Q8 متى نستطيع معرفة موقع النود التي في الخلف
When We can know The location of The Node in The back ? When We use duplelinked list .
عندما نستخدم دبل لينكد ليست .

Q9 متى تسبب عملية الحذف في المفقودة Hole
When does The delet operation cause a Hole ?
عندما يكون الحذف من الأول أو من الوسط .
When The delet in sits on middle array .

Q10 كيف تكون عملية الحذف في اللينكد ليست
How is The deletion Process in linked list ?
نختار نود ثم نجعلها Null ثم نصبح متاحة .
Choose Node Then Put it Null . Now The Null free

Algorithms: is searching mision and disined by disiner or Programer

- binary search
- Sequential search

Sequential search

Sequential search : Search element by element.

(البحث عنصر بعد عنصر (إلى أن تجد العنصر المطلوب).
ويبدأ البحث من أول عنصر.

Comparison : Number of comparison for CPU

عدد المقارنات التي يعرضها CPU (للوصول لعنصرين)

binary searching

Ways t case : When the data is in the last Cell That hold data.

الحالة السيئة

[البحث عندما تكون البيانات في آخر خلية تحمل بيانات]

Best case : When The data is in the first cell

الحالة المفضلة

[عندما تكون البيانات المطلوب البحث عنها في أول خلية]

avarige case : When The data is in the middle cell

الحالة المتوسطة

[عندما تكون البيانات المطلوب البحث عنها في الخلية الوسطى]

binary search

Binry search : looks for Items in a list by using dividing-and-conquer strategy.

يبحث عن العناصر باستخدام استراتيجية تفريق البيانات.

عدد المقارنات = عدد البيانات مرفوع للأس

مثل: 16 مقارنة = 2^4

للحصول على :

Waist case : 5

Best case : $\frac{0+5}{2} = 2.5 \approx \boxed{3}$

avarge case : $3 \div 2 = 1.5 \approx \boxed{2}$

in linked list.

عن العناصر بطيئ .

هي مصفوفة مرتبة من البيا

an ordered set of

y

emory allocation

linked

١ هياكل البيانات

نستخدم ال array اذا كان الحجم معقول اما اذا كان كبير يصير

overflow في حال كونتينر

false overflow

فاذا كنت حجزت مليون مكان

وحطيت ٥٠٠ الف من البيانات

وبقي ٥٠٠ الف فاضية تكون

waste علما

او اتلفناها

اتلفناها

struct 400

complex type

لا نستخدم ال array في الاعداد الكبيرة لان الarray يكون حجز مسبق

dynamic Memory Au

Linked list ↑

Pointer

هو وسيلة للوصول للعناوين

ال array متسلسل

Node

dynamic اعرف انا

Memory Au

اما ال Karrayd اعرف انا الى

/ /

المشكلة اللي ممكن نواجهها اذا ما استخدمنا الbinary search tree

تكون في الـ movement للبيانات

لان بنخزنها في array :-

A	B	D	E	F	G

~~نواجه~~ نواجه مشكلة هنا في ان اذا جينا نرتب ونكتشف ان مافي

مكان للـ C

يصير اذا جينا نخزن C في الـ array بتخزن كذا

A	B	D	E	F	G	C			

نلاحظ ان فيه خطأ في الترتيب ↑

نحتاج نروح للبيانات movment عشان نفضي مكان للـ C في

مكانها الصح

عشان كذا راح نستخدم Data structure اسمها binary search tree

هذا والله اعلم

12/1
2

Data structure → array

Divide and conquer اسلوب بحث/الرتيابجي

binary search نستخدمه في البيانات الكبيره فقط

اما في حال كانت البيانات بسيطة نستخدم ال Linear Search

نحتاج ال binary ~~sh~~ Search في اعلى فرق مقارنة.

لان الفرق البسيط مايغير شي كبير اما في حال ان البيانات كبيره

نستخدمها في البيانات الكبيره عشان يكون الفرق عالي مثل قلنا عندنا مليون من البيانات يصير الفرق بدل ما يسوي مليون مقارنة تصير 20 مقارنة فقط

اذا فيه فرق كبير للبيانات الكبيره اما للبيانات القليلة يكون فرق بسيط

بينما اقدر اختصر وقت ال binary search في الترتيب (Sort)

وفيه Data structure مبنية لل binary search خصيصاً اسمها :

binary Search tree يكون سلس اكثر ومافيه اخطاء

1 Febl 14 هياكل البيانات

big O (n)

worst
best
avreage

في

Worst is the worst thingg coulde happen

Ex: the information is not avelable or will be the last

Leaneer/sequentiol

Searching algerthms

binary search

$\log_{10}(100) = 2$

lower bound = 0

upper bound = عدد البيانات

$\log_2(n) = 8 = 2^3$ ← عدد الخطوات

عدد البيانات

binary

متى احتاج ال linear search ومتى نحتاج ال binary search

يشترط ان تكون البيانات مرتبة

/ /

Traverse

movement Data up = عملية

Hole

تسبب تأخير في الوقت

Comparision

Insert
overwrite
Update

$O(n) \simeq n$ عدد العمليات

big O(n) worst case

complexity of alg

best case
$O(n) \simeq 1$
linear

avrage case
$n/2$

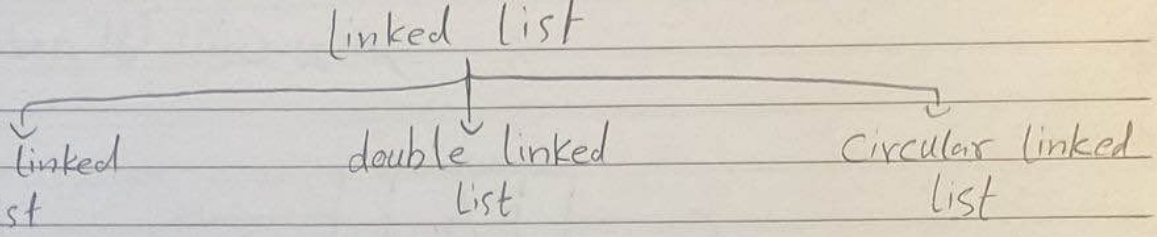
$O(n) \simeq n/2$
avrage case

$O(n) \simeq n$
Worst case

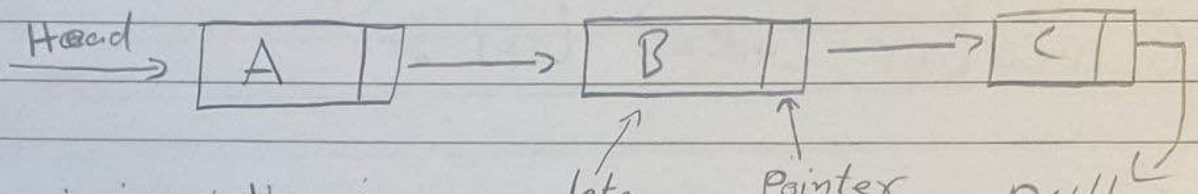
Best case
$O(n) \simeq 1$

→ linear/seqontal search

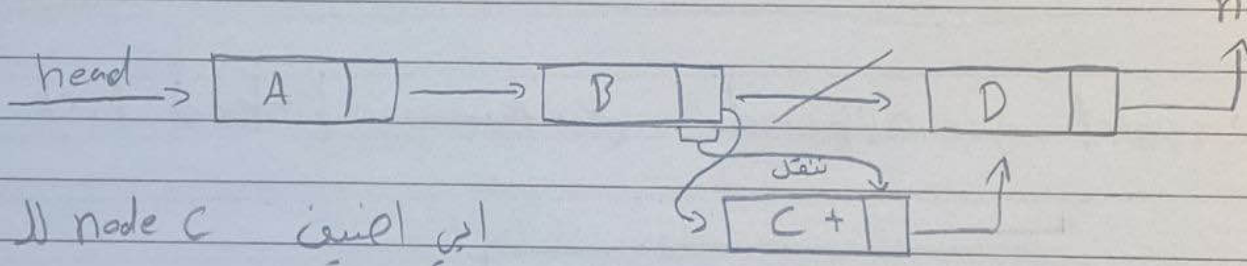
linked list تلخيص موضوع



Singl linked list :-



في حال بي تضيف node
للـ list ، يكون افضل حال بعد node



list الـ node C اني اضيف

افضل حل اضيف C بعد الـ B

لان البوينتر حق الـ B بيأشر على عنوان الـ D

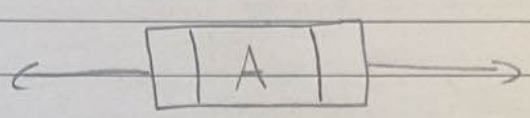
نحفظ البوينتر حق الـ B في الـ C بيصير الـ C يأشر على الـ D

ونحفظ عنوان الـ C في البوينتر حق الـ B

١ اكمله لشرح موضوع ال linked list

Double linked list :-

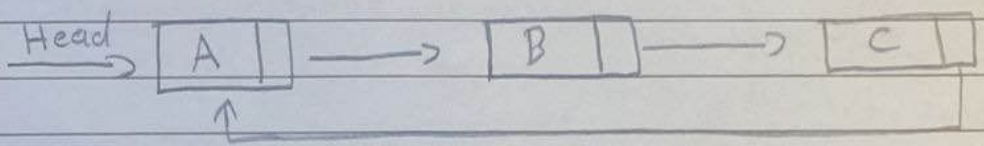
عباره عن node لها اثنين بوينتر



هذا هو ال doubl linked list

Circular linked list :-

هذا الشكل يمثل ال circular linked list



في ال linked list نستخدم append بدل insert

Insert → array
append → linked list

١ / اشرح عن الـ Stake & Queue هياكل البيانات

Stake :-

Stake قاعدة الـ
last in First out

عندنا الـ Function اسمها Is Empty و Is Full نستخدمها مع

Is Empty → Static → Top = -1 (فاضي)
→ dynamic → Top = null (فاضي)

متى نستخدم Is Empty ؟

نستخدمها قبل الحذف "قبل الـ Pop"

ليش قبل الحذف ← عشان ما نواجه مشكلة under Flow

متى نستخدم Is Full ؟

نستخدمها قبل الاضافة "قبل الـ Push"

Is Full → Static

نستخدم Is Full مع الـ Static فقط ، ليش ؟

لأن الـ dynamic ما تحتمل يكون Full

ليش نستخدم Is Full قبل الاضافة ؟

عشان ما نواجه مشكلة overflow

ملحوظة :
Stake
Push → اضافة
Pop → حذف
enqueue → اضافة
dequeue → حذف
Queue

arrays

هذي اسمها Index

0	1	2	3	4
A	B	C	D	E

المربع الواحد اسمه خلية

خلية
Cell

في حال بي نحذف من الـ array :-

عبارة عن اسم او عنوان لـ Index الخلية

مثلا :-

اذا قررنا نحذف x[3]

مثلا لو اقول لك x[3] = ?

Index

x[3] = D

0	1	2	3	4
A	B	C		E

الفراغ ذا اسمه hole

ونحتاج نحرك البيانات اللي على يمين الـ hole عشان نسد الفراغ

①

0	1	2	3	4
A	B	C		E

movement

②

1	2	3	4	5
A	B	C	E	

بعد الـ movement

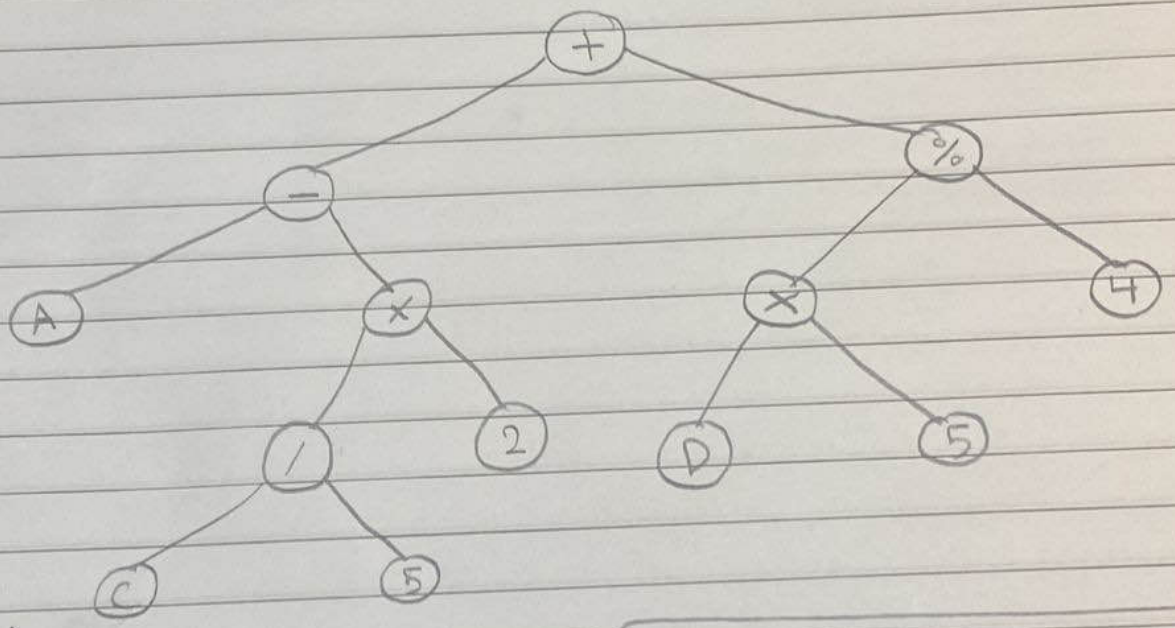
لو حذفنا من الاخير ما راح يسبب لي "فراغ" hole اذا ما يحتاج نسوي شي

اما اذا حذفنا من البداية راح يسبب لي hole ولازم نسد الفراغ وسوي movement

توضيح لموضوع الـ tree

Prase tree :- بيعطينا معادلة زي كدا وممكن يطلب الـ tree Post order, Inorder, Preorder

$$\underbrace{A-(C/5\times 2)}_{\text{left side}} + \underbrace{(D\times 5\ \%\ 4)}_{\text{Right side}}$$



Pre order ← الاب قبل الابناء

Postorder ← الابناء قبل الاب

Pre order: + - A X / C 5 2 % X D 5 4

Post order: A C 5 / 2 X - D 5 X 4 % +

In order: A - C / 5 X 2 + D X 5 % 4