الفصل الاول

المحتوي:

١ :مقدمة

٢ :binary search

٣ :BIG O notation

لو عندك قاموس وحابه تبحثي عن كلمة تبدا بحرف ال "ك"

اما تبداي من اول صفحه وتبحثي لحد ماتوصلي لحرف ال ك وده هياخد وقت اكيد كبير
الطريقة دي اسمها Simple search

او انك عارفه انه الحرف ده في اواخر المعجم وتبداي البحث من اخر كام صفحة
الطريقة دي اسمها binary search

Binary search  simple search

هنفترض اننا عندنا لستة تحتوي علي٥ ارقام وعاوزين نبحث عن رقم٧٠ موجود بداخلها ام لا
[10,20,30,50,70]

Simple search

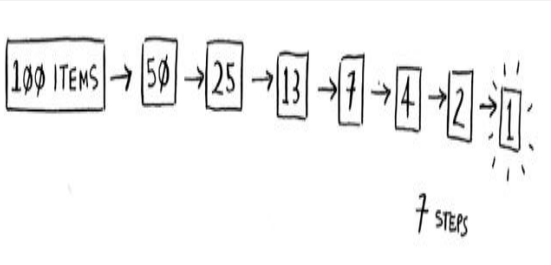
هنعدي علي رقم رقم بالترتيب داخل اللستة ونتاكد هل يساوي ٧٠ ام لا
هنعمل ٥ دورات حتي نصل للاجابة

binary search

هنشتغل بطريقة المتوسط والبحث من خلال موقع الرقم
١:المتوسط هنا ٢ مع العلم اول رقم في اللستة موقعه يساوي "0"
هل رقم ٣٠ يساوي ٧٠
"لا " هل الرقم اكبر ام اصغر ؟
اصغر
اذا هيتم تجاهل رقم ٣٠ وجميع الارقام اللتي تسبقة
واللستة الجديدة بالنسبالنا هتبدا من رقم ٥٠"50,70]"
٢:المتوسط هنا تقريبا رقم ١ يعني موقع رقم 70وده الرقم اللي عاوزينه
عدد الخطوات هنا خطوتين فقط علي عكس الSimple search خمس خطوات

كيفية حساب عدد العمليات اللتي تجري في كلا من الطريقتين
اذا كان لدينا لستة تحتوي علي عدد nمن العناصر

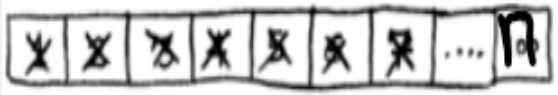
binary search



زي مافي في الصورة بتيم قسمة العدد علي ٢ حتي نصل للرقم عدد الخطوات هنا ٧

عدد الخطوات بيساوي $\log n$

Simple search



في اسوا الحالات ان يكون الرقم في اخر اللستة فان عدد الخطوات يساوي ١٠٠

عدد الخطوات يساوي n

توضيح
اللوغاريتم دايما
للاساس
2

$10^2 = 100 \leftrightarrow \log_{10} 100 = 2$

$10^3 = 1000 \leftrightarrow \log_{10} 1000 = 3$

$2^3 = 8 \leftrightarrow \log_2 8 = 3$

$2^4 = 16 \leftrightarrow \log_2 16 = 4$

$2^5 = 32 \leftrightarrow \log_2 32 = 5$

كود لتوضيح ال binary search

```
def binary_search(list, item):
    low = 0
    high = (len(list)-1
    while low <= high:
        mid = (low + high)//2
        guess = list[mid]
        print("mid ==",mid)
        if guess == item:
            return mid
        if guess > item:
            high = mid - 1
        else:
            low = mid + 1
    return None
my_list = [1, 3, 5, 7, 9]
print (binary_search(my_list, 3)) # => 1
print (binary_search(my_list, -1)) # => None
```

للتاكيد ان اللستة مش فاضية
هنا يتم حساب المتوسط

زي مااتفقنا انه المتوسط عبارة عن موقع الرقم داخل اللستة هنا بنستخدمه لاستخراج القيمة من اللستة

اذا كان الرقم اللي ال indexبتااعه هو المتوسط اكبر من الرقم المطلوب ده معناه انه كل الارقام اللي اكبر من الرقم ده انا مش محتاجها ف اكبر index بالنسبالي دلوقتي هو ال indexاللي قبل المتوسط

اذا كان الرقم اللي ال indexبتااعه هو المتوسط اصغر من الرقم المطلوب ده معناه انه كل الارقام اللي اصغر من الرقم ده انا مش محتاجها ف اصغر index بالنسبالي دلوقتي هو ال indexاللي بعد المتوسط

Big O notation

من خلال ال notationده بنعرف اد ايه ال algorithm سريع

الفكره انه سرعة ال **algorithm** لا تتناسب مع الوقت ولكن بتتتناسب مع عدد العمليات اللي بينفذها

تعالو ناخد مثال :

اذا كان لدينا لستة تحتوي علي٨ رقم وكل رقم يستغرق ١ ثانية اثناء التاكد من انه الرقم المطاول او لا فاذن الوقت المستغرق للتاكد من جميع الارقام بداخل اللستة هو ٨ ثواني

ولكن ذلك اذا كنا نستخدم طريقة ال **simple research** لانه كما اتفقنا ان عدد العمليات في ال **simple search** تساوي عدد الارقام في اللستة

ما هو الزمن المستغرق اذا كانت الطريقة المستخدمة هي **binary search** ؟

بما ان عدد العمليات تساوي **log(n) =log (8)** = ٣

اذا الزمن = ٣* ١ ثانية **= ثلاث ثوان**

١اذا تضاعف عدد الارقام هل سيتضاعف الوقت المستغرق ؟

الاجابة علي حسب الطريقة المستخدمة

Simple search

بما ان ال n=16

اذا الزمن هيساوي ١٦

اذا الزمن تضاعف

Binary search

بما ان ال n=16

اذا الزمن هيساوي

4=log(16)

اذا الزمن لم يتضاعف

لذلك سرعة algorithm تتناسب مع عدد العمليات وليس الزمن

Some common Big O run times