الباب الأول

الجوريزم(خوارزمية):

مجموعة من الخطوات الي بتتبعها لحل مشكلة او انجاز مهمة معينة بتقابلك.

أي كود ممكن نعتبره الجوريزم مهما كان بسيط اهم حاجة في الالجوريزم هو قد ايه بيكون عملي و بيوفرلي وقت و مساحة تخزين.

في الباب ده هنتعلم اول اتنين الجوريزم يستخدموا في البحث و هنفهم الفرق بينهم و نحدد ايه فيهم الأنسب للحالة الي بتقابلنا.
و بعد كده هنتعرف على ال big O الي بنستخدمها عشان نقارن الالجوريزمات المختلفة من حيث السرعة .

البحث:

خلينا نفترض ان عندك كتاب صفحاته مترقمة من 1 ل 100 و عايز تروح للصفحة رقم 100 فا ابسط حاجة ممكن تعملها انك هتقلب فالكتاب صفحة صفحة لحد ما توصل للصفحة 100 طبعا دي طريقة مملة و محدش هيعمله و الأقرب للواقع انك هتفتح الكتاب من الاخر طب لو قلتك انا عايز الصفحة 65 فا التفكير الي هيجي في بالك هو انك هتفتح الكتاب قريب من النص و بعدين تقلب شوية بشوية لحد ما توصل للصفحة 65 الطريقتين دول قريبين جدا من الجورزمين البحث الي هنتكلم عليهم الي هما linear search و binary search.

linear search

ده ابسط الجوريزم بحث و فكرته بكل بساطة انك بتمشي على ال list او ال array بتاعك عنصر عنصر و اشوف هل العنصر الي انا واقف عليه هو العنصر الي انا عايزه خلاص هوقف هنا و ارجع المكان بتاعه طب لا مش ده الي انا عايزه هنقل على الي بعده يعني نفترض ان عندنا array فيه ارقام من 1 ل 100 بالترتيب لو انا عايز رقم 54 فا الي ال linear search هيعمله انه هيمشي على ال array اول عنصر يسأل هل هو 54 طبعا لا ننقل على الي بعده هل هو 54 لا و هكذا لحد ما نوصل للعنصر رقم 54 (او 53 بمعنى اصح عشان ال array ترقيمه بيبدأ من 0) و هيلاقيه فعلا هو الي انا بدور عليه فهيرجعلي المكان بتاعه في ال array و طبعا كل ما الرقم الي بدور عليه يكون ابعد في ال list هاخد خطوات اكتر عشان اوصله.

Binary search

ده الالجوريزم الأقرب للمنطقية و قريب جدا لفكرة انك تفتح الكتاب قريب من النص و بعدين تقلب للصفحة الي انت عايزها الشرط الوحيد الي لازم تاخد بالك منه هو ان ال array الي بتدور فيه يكون مترتب و الا هتشتغل linear عادي طب ال binary search كرته أيه فكرته انك بتقسم ال array نصين يعنيفي حالتنا هتروح عند العنصر رقم 49 الي هو هيكون 50 و هتسأل هل الرقم الي بدور عليه اكبر من ولا اصغر من ال50 طبعا ال54 اكبر فا هتلغي كل العناصر الي اصغر من او بتساوي ال50 من ال array بتاعك هتعتبرها مش موجودة و هنخلي البداية بتاعت ال array العنصر ال50 فا كده ال array الي معانا موجود فيه الأرقام من 51 لحد 100 فا نقسمه نصين تاني هنلاقي النص بتاعنا بقى رقم 75 فنسأل نفس السؤال تاني هل ال75 اكبر من ولا اصغر من 54 طبعا اكبر من فا ساعتها هنعتبر النهاية بتاعتنا هي 75 بدل 100 و

نكرر نفس العملية تاني لحد ما في مرة النص بتاعنا هيبقى هو الرقم الي بندور عليه (ده لو كان فعلا موجود ممكن تدور على عنصر و يكون مش موجود في ال array)في حالة رقم 54 هنعرف نوصله بعد 6 محاولات بس في حين ان مع ال linear search هنحتاج 54 محاولة فا طبعا ال binary search اوفر و اسرع بكتير.

هل ديما ال binary search هو الحل؟

طبعا الإجابة لا بس قبل ما نقول ليه لا اكيد و انت بتقرأ حسيت ان المشكلة بتاعت الأرقام دي تافهة اوي يعني انا لو فاهم ال array شغال ازاي فبمنتهى البساطة لما تقولي فين رقم 54 هقولك انه في المكان رقم 53 بس ده كان مجرد مثال توضيحي مش دايما تعاملك هيكون مع ارقام و مش ديما هتكون عارف ايه محتوى ال list الي انت بتدور فيها يعني ممكن ال array بتاعك يكون كده {1,2,5,10,13,14} فحالة زي دي مش هتقدر تعرف مكان العتصر الي بتدور عليه بشكل مباشر و في حالة زي دي ال binary search هيكون احسن بس لازم تاخد بالك ان لما تستخدم الbinary search العناصر بتاعتك تكون مترتبة لكن في حالة زي كد {1,4,2,5,3,12,40} ال binary search مش هيقدر يساعدك انك توصل للعنصر حتى لو كان موجود فعلا و هتحتاج تستخدم linear search.

أوجه المقارنة	Linear search	Binary search
السرعة	ابطأ	أسرع
نوع الداتا الي بيتعاملوا معاها	أي نوع من الداتا	أي نوع من الداتا
ترتيب ال array	مش مهم يكون مترتب	لازم يكون مترتب

Big O

دلوقتي انت ممكن تكون خدت بالك اني اتكلمت عن السرعة و قلت ان ال binary searchاسرع من ال linear طب ازاي السرعة دي بتتقاس و ازاي ممكن نقول ان الجوريزم معين اسرع من التاني هنا بيجي دور ال big O اسمها كده عشان بتحط O قبل الرقم الي بيمثل سرعة الالجوريزم بتاعك) و هي طريقة بتسهل على المبرمجين التواصل و بتبسط مقارنة الالجريزمات ببعض .

طب ازاي نستخدمها؟ خلينا نقول انك هتستخدم ال linear searchعشان تدور على عنصر جوا arrayفيه 100 عنصر أسوأ سيناريو ممكن يحصل انك تكون عايز العنصر الأخير ساعتها هتحتاج 100 محاولة طب لو ال array في 1000 عنصر هتحتاج 1000 محاولة طب 10000 يبقى 10000 محاولة لو لاحظت هنا عدد المحاولات بيساوي عدد العناصر الي هتعامل معاها يعني لو قلنا ان عدد العناصر الي عندي هو n يبقى كده linear searchبياخد وقت O(n).

طب انت ممكن دلوقتي تقول طب انا ممكن العنصر الي بدور عليه يكون في اول ال array او في نصه فمحتاجش أوصل للاخر تمام ده سيناريو وارد فعلا بس ال big O بتهتم بس بالحالة الاسوء و بتقيس عليها لان الالجوريزم بتاعك هيشتغل على سيناريوهات كتير منها السهلة الي تخلص على طول و منها سيناريوهات معقدة فا انا عايز اعرف اسوء حاجة ممكن تحصل ايه هي و بقيم الالجوريزم على أساسها.

دلوقتي ازاي نحسب الوقت بتاع ال binary search لو جربنا ندور في arrayفيه 4 عناصر اكبر عدد محاولات هنحتاجه هيكون 2 طيب لو 5 عناصر هنحتاج 3 محاولات طب 6 عناصر لسه 3 محاولات لحد ما نوصل 8 بعد 8 هيبقوا 4 محاولات لحد ما نوصل 16 بعد ال16 هيبقوا خمسة ده مش بيفكرك بحاجة؟ اللوغاريتم ال binary search بيحتاج لوغاريتم عدد العناصر ل 2 (دايما الأساس بتاعي بيكون 2 لما بيتكلم عن اللوغاريتم فالكتاب ده) يعني دلوقتي او انا كنت

يدور على عنصر في array حجمه 4 مليار عنصر هحتاج 32 محاولة بس في أسوأ حالة فا كده نقدر نقول ان ال big O لل binary search عبارة عن log(n) .

اشهر الأوقات الي هتقابلك في التعامل مع الالجوريزمات:

الابطأ **O(log(n)),O(n),O(n log(n)),O(n^2),O(n!)** الاسرع

مثال على ملكة حلها بيحتاج O(n!)

Travelling salesman problem