الأكاديمية العربية الدولية
Arab International Academy

إسم المادة: الخوارزميات

إسم المحاضِر: م. خليل المحمد

الأكاديمية العربية الدولية – منصة أعد

مقدمة عن الخوارزميات

الخوارزمية (Algorithm) هي مجموعة من الخطوات المتسلسلة والمحددة التي تُنفذ لحل مشكلة معينة أو تنفيذ مهمة معينة. تعتمد الخوارزمية على مدخلات (Inputs)وتعطي مخرجات (Outputs) بعد تنفيذ هذه الخطوات بشكل محدد ومرتب. يجب أن تكون الخوارزمية واضحة، قابلة للتنفيذ، ولها نهاية محددة.

أهمية الخوارزميات في علوم الحاسوب:

1. **حل المشكلات بكفاءة**: الخوارزميات تساعد في حل المشكلات بطريقة فعّالة ومنظمة، مما يؤدي إلى تحسين أداء البرمجيات وتقليل وقت التنفيذ.
2. **تحسين استخدام الموارد**: من خلال استخدام الخوارزميات المناسبة، يمكن تقليل استخدام الذاكرة والمعالجة، مما يساعد على إدارة الموارد بشكل أفضل.
3. **قابلية التوسع**: الخوارزميات الجيدة تساعد في تصميم أنظمة يمكنها التكيف مع زيادة حجم البيانات أو عدد المستخدمين بسهولة.
4. **أساس البرمجة والتطوير**: الخوارزميات تشكل جوهر البرمجة، حيث تُستخدم لتطوير حلول برمجية للمشكلات المعقدة.
5. **تحليل الأداء**: من خلال فهم الخوارزميات، يمكن تحليل أداء الأنظمة وتحديد نقاط القوة والضعف في التطبيقات البرمجية.
6. **التخصصات المتعددة**: تلعب الخوارزميات دورًا محوريًا في مجالات متعددة مثل الذكاء الاصطناعي، علم البيانات، الشبكات، والتشفير.

مجالات تطبيق الخوارزميات

1. **الذكاء الاصطناعي والتعلم الآلي**: تستخدم الخوارزميات لتدريب النماذج والتنبؤات ومعالجة البيانات الضخمة.
2. **معالجة البيانات الضخمة (Big Data)**: تعتمد الخوارزميات على تحليل كميات ضخمة من البيانات بسرعة وكفاءة.
3. **التشفير وأمن المعلومات**: تستخدم الخوارزميات في تصميم أنظمة التشفير لحماية البيانات وتأمين الاتصالات، مثل خوارزمية AES وخوارزمية RSA.
4. **الشبكات والاتصالات**: الخوارزميات تلعب دورًا رئيسيًا في توجيه البيانات عبر الشبكات وتحسين أداء البروتوكولات.
5. **تحليل الرسوم البيانية (Graph Theory)**: يتم استخدام الخوارزميات لحل المشكلات المتعلقة بالشبكات والرسوم البيانية، مثل خوارزمية Dijkstra لإيجاد أقصر مسار، وخوارزمية Kruskal لتوليد شجرة التغطية الدنيا.
6. **الألعاب والمحاكاة**: في تطوير الألعاب ومحاكاة الأنظمة، تستخدم الخوارزميات لتوليد الذكاء الاصطناعي وتحليل الأداء وتوليد السيناريوهات التفاعلية.
7. **البحث والفرز**: في محركات البحث والتطبيقات المختلفة، يتم استخدام خوارزميات مثل البحث الثنائي والفرز السريع لتحسين سرعة ودقة العمليات.
8. **تحسين الأنظمة**: الخوارزميات تستخدم لتحسين أداء الأنظمة الصناعية واللوجستية من خلال البرمجة الديناميكية والخوارزميات الجشعة لتحقيق الحلول المثلى.
9. **التجارة الإلكترونية وتحليل السوق**: تحليل البيانات وتقديم التوصيات في التطبيقات التجارية والمالية يتم عبر خوارزميات التحليل والتنقيب عن البيانات.

مفاهيم أساسية في الخوارزميات

1. **المدخلات:(Inputs)**
الخوارزمية تبدأ بمدخلات، وهي البيانات أو المعطيات التي تحتاج إلى معالجتها. يجب أن تكون المدخلات محددة وواضحة لتتمكن الخوارزمية من تنفيذ العمليات عليها.

2. **المخرجات:(Outputs)**
المخرجات هي النتائج أو الحلول التي تنتج عن تنفيذ الخوارزمية. يجب أن تكون المخرجات صحيحة ومتناسبة مع المدخلات بعد إتمام جميع الخطوات بنجاح.

3. **التعقيد الزمني:(Time Complexity)**
يشير إلى كمية الوقت الذي تستغرقه الخوارزمية لتنفيذ مهمتها اعتمادًا على حجم المدخلات. يُعبر عنه عادة باستخدام تدوين O الكبير، مثل O(n) أو O (log n)، وهو مقياس لعدد العمليات التي تحتاجها الخوارزمية.

4. **التعقيد المكاني:(Space Complexity)**
يشير إلى كمية الذاكرة التي تحتاجها الخوارزمية أثناء تنفيذها. على غرار التعقيد الزمني، يُستخدم تدوين O الكبير لتقدير مساحة الذاكرة المطلوبة، مثل O(n) أو O (1).

5. **التكرار:(Iteration)**
هو عملية تكرار مجموعة من الخطوات عدة مرات في الخوارزمية حتى تحقيق الهدف أو الوصول إلى شرط معين. يُستخدم عادة في الخوارزميات التي تعتمد على الحلقات.

مفاهيم أساسية في الخوارزميات

6. العودية :(Recursion)

هي عملية استدعاء الخوارزمية لنفسها لحل مشكلة أصغر داخل المشكلة الكبيرة. تُستخدم العودية كثيرًا في الخوارزميات مثل فرز الدمج (Merge Sort)وخوارزميات التحقق من الهياكل الشجرية.

7. الصحة والاكتمال :(Correctness & Completeness)

تشير الصحة إلى أن الخوارزمية تؤدي المهمة المطلوبة بشكل صحيح وتنتج مخرجات دقيقة. الاكتمال يعني أن الخوارزمية تعمل في جميع الحالات الممكنة وتغطي كل السيناريوهات المتوقعة.

8. الكفاءة :(Efficiency)

الكفاءة تقاس بمدى سرعة الخوارزمية واستخدامها الفعال للموارد. تعتمد الكفاءة على التعقيد الزمني والمكاني، ومدى تحسين الخوارزمية لتلك العوامل لتحقيق أداء جيد.

9. القابلية للتوسع :(Scalability)

هي قدرة الخوارزمية على التعامل مع زيادة حجم المدخلات دون تدهور كبير في الأداء. الخوارزميات القابلة للتوسع يمكن أن تنفذ بكفاءة حتى عند معالجة بيانات كبيرة.

10. الاستقرار :(Stability)

يشير إلى سلوك الخوارزمية عند وجود مدخلات مكررة. الخوارزمية المستقرة تحتفظ بترتيب العناصر المتساوية في المدخلات كما هو في المخرجات.

التعقيد الزمني والمكاني

- **التعقيد الزمني:(Time Complexity)**

يشير إلى كمية الوقت الذي تحتاجه الخوارزمية لإكمال تنفيذها بناءً على حجم المدخلات. يتم قياسه بعدد الخطوات أو العمليات التي تقوم بها الخوارزمية مع زيادة حجم المدخلات. يتم التعبير عن التعقيد الزمني باستخدام تدوين O الكبير(Big O Notation) ، مثل:

  - **O (1):** وقت ثابت، لا يتغير مع حجم المدخلات.
  - **O(n):** يتناسب الخطوات بشكل مباشر مع حجم المدخلات.
  - **O (log n):** ينمو ببطء مع زيادة حجم المدخلات.
  - **O(n^2):** ينمو عدد الخطوات بشكل تربيعي مع حجم المدخلات.

- **التعقيد المكاني:(Space Complexity)**

يشير إلى كمية الذاكرة التي تستخدمها الخوارزمية أثناء تنفيذها. يقيس مقدار الذاكرة الإضافية التي تحتاجها الخوارزمية بجانب المدخلات نفسها. مثل التعقيد الزمني، يُعبر عن التعقيد المكاني باستخدام تدوين O الكبير، ويعتمد على حجم المدخلات. التعقيد المكاني يحدد ما إذا كانت الخوارزمية تتطلب ذاكرة إضافية كبيرة أو يمكن تنفيذها باستخدام ذاكرة صغيرة.

  - مثال :في خوارزمية فرز مستقر، إذا كان لدينا عنصران متماثلان في القيمة في المدخلات، فسوف يظل ترتيبهما النسبي نفسه في المخرجات.

العنوان: طريقة كتابة البحث

- **الاستقرار:(Stability)**

الاستقرار في الخوارزميات يعني أن الخوارزمية تحافظ على ترتيب العناصر المتماثلة (ذات القيمة نفسها) كما كانت في المدخلات بعد تنفيذ الخوارزمية. هذا المفهوم مهم في خوارزميات الفرز، حيث يمكن أن يكون للبيانات المتماثلة خصائص إضافية بخلاف القيمة، ويجب الحفاظ على ترتيبها النسبي.

- **التكرار:(Iteration)**

التكرار هو عملية إعادة تنفيذ مجموعة من الخطوات أو العمليات عدة مرات حتى يتم الوصول إلى نتيجة معينة أو شرط محدد. يُستخدم التكرار في الخوارزميات لحل المشكلات المتكررة أو لتقليل حجم البيانات بطريقة تكرارية.

- مثال :حلقة (loop) تكرارية في الخوارزمية قد تستخدم لفرز أو معالجة جميع العناصر في قائمة حتى يتم تحقيق شرط معين، مثل العثور على عنصر معين أو ترتيب القائمة بالكامل.

أنواع الخوارزميات

1- الخوارزميات التتابعية (Sequential Algorithms) : هي نوع من الخوارزميات التي تنفذ خطواتها بشكل متسلسل خطوة بخطوة.

خصائص الخوارزميات التتابعية:

- التنفيذ المتسلسل: يتم تنفيذ الخطوات الواحدة تلو الأخرى.
- عدم التزامن: كل عملية تعتمد على انتهاء العملية التي قبلها.
- البساطة: تعتبر سهلة الفهم والتنفيذ لأنها تعتمد على تسلسل محدد.

2- الخوارزميات العودية (Recursive Algorithms) : هي نوع من الخوارزميات التي تستدعي نفسها لحل مشكلة معينة، حيث يتم تقسيم المشكلة إلى مشاكل أصغر وأكثر بساطة. تعتبر العودية طريقة فعالة لحل المشكلات التي يمكن تقسيمها إلى نفس النوع من المشكلات، مما يجعلها مناسبة لعديد من التطبيقات، مثل معالجة الأشجار والرسوم البيانية.

مفهوم العودية (Recursion) : هو عملية استدعاء الخوارزمية لنفسها لحل جزء من المشكلة. في هذه العملية، يتم تقليل حجم المشكلة في كل استدعاء، حتى الوصول إلى حالة قاعدة (Base Case) يمكن من خلالها تحديد النتيجة النهائية دون الحاجة إلى مزيد من الاستدعاءات.

أنواع الخوارزميات

3- الخوارزميات التكرارية (Iterative Algorithms): هي خوارزميات تستخدم حلقات مثل for و while لتكرار مجموعة من التعليمات حتى تتحقق شروط معينة أو حتى الوصول إلى نتيجة بينما الخوارزميات العودية (Recursive Algorithms) هي خوارزميات تستدعي نفسها لحل مشكلة معينة، حيث يتم تقسيم المشكلة إلى مشكلات أصغر حتى الوصول إلى حالة قاعدة.

العلاقة بين الخوارزميات التكرارية والعودية:

- **أساليب الحل:** كلا النوعين يهدفان إلى حل المشكلات ولكن يستخدم كل منهما أسلوبًا مختلفًا. الخوارزميات التكرارية تستخدم الحلقات، بينما العودية تعتمد على استدعاء الدالة لنفسها.
- **الحالات الأساسية:** في الخوارزميات العودية، يجب تحديد حالة قاعدة لإنهاء الاستدعاءات. بينما في الخوارزميات التكرارية، تُستخدم الشروط في الحلقة لإنهاء التنفيذ.
- **الكفاءة:** في بعض الحالات، يمكن أن تكون الخوارزميات التكرارية أكثر كفاءة من الخوارزميات العودية من حيث استخدام الذاكرة والأداء. حيث أن الخوارزميات العودية قد تؤدي إلى استهلاك كبير للذاكرة بسبب تخزين استدعاءات الدالة في مكدس الاستدعاءات.
- **البساطة:** يمكن أن تكون الخوارزميات العودية أكثر وضوحًا وبساطة في بعض الحالات، خاصة عندما يكون الحل الطبيعي للمشكلة هو التقسيم إلى مشاكل أصغر. ولكن، قد تكون الخوارزميات التكرارية أكثر وضوحًا للمشكلات البسيطة.
- **تحويل الخوارزميات:** يمكن تحويل الخوارزميات العودية إلى تكرارية، حيث يمكن استخدام الحلقات لمحاكاة الاستدعاءات المتكررة. ولكن، يجب أن يكون ذلك مصحوبًا بتحليل الأداء.

أنواع الخوارزميات

4- الخوارزميات العشوائية (Randomized Algorithms): هي نوع من الخوارزميات التي تستخدم العشوائية كجزء من المنطق الخاص بها. تعتمد هذه الخوارزميات على توليد أرقام عشوائية أو اتخاذ قرارات عشوائية خلال تنفيذها، مما يمكن أن يؤثر على النتيجة النهائية أو تسريع عملية الحل.

أنواع الخوارزميات العشوائية:

- **الخوارزميات العشوائية المباشرة:** تعتمد على اتخاذ قرارات عشوائية مباشرة. على سبيل المثال، اختيار عنصر عشوائي من مجموعة للقيام بعملية معينة.
- **الخوارزميات العشوائية المحسوبة:** تستخدم العشوائية لتحديد خطوات معينة في عملية معقدة. مثال على ذلك هو خوارزمية "Quick Sort" التي تختار محورًا عشوائيًا لتقسيم البيانات.
- **الخوارزميات العشوائية التكرارية:** تستخدم العشوائية في عملية تحسين أو تعديل الحلول المتكررة، مثل خوارزميات."Simulated Annealing"

أمثلة على الخوارزميات العشوائية:

- خوارزمية:"Quick Sort"
  تستخدم اختيار محور عشوائي لتقسيم قائمة، مما يحسن الأداء في كثير من الحالات.
- خوارزمية:"Randomized Selection"
  تستخدم لاختيار العنصر k الأكبر من قائمة من العناصر، وتستند إلى اختيار عنصر عشوائي وتقسيم القائمة بناءً على هذا العنصر.

أنواع الخوارزميات

5- الخوارزميات الديناميكية (Dynamic Programming Algorithms): هي أسلوب لحل المشكلات المعقدة من خلال تقسيمها إلى مشاكل أصغر وأكثر بساطة، ثم تخزين نتائج هذه المشاكل لحلها بشكل متكرر دون الحاجة إلى إعادة حساب النتائج. تستخدم هذه الخوارزميات بشكل خاص عندما تتداخل المشاكل، مما يعني أن الحل لمشكلة واحدة يعتمد على الحلول لمشاكل أخرى.

خصائص الخوارزميات الديناميكية:

- **التحليل إلى مشاكل أصغر:** تقوم الخوارزميات الديناميكية بتحليل المشكلة الأصلية إلى مشاكل فرعية أصغر، مما يسهل حلها.
- **التخزين المؤقت(Memoization) :** تستخدم الخوارزميات الديناميكية تقنية التخزين المؤقت لتخزين نتائج المشاكل الفرعية، مما يمنع الحاجة إلى إعادة حسابها. يمكن أن يتم ذلك عبر استخدام هياكل بيانات مثل المصفوفات أو القوائم.
- **حل المشكلات التكرارية:** تتعامل الخوارزميات الديناميكية بشكل جيد مع المشكلات التي يمكن إعادة استخدامها، مما يجعلها فعالة من حيث الوقت.

أمثلة على الخوارزميات الديناميكية:

مشكلة الموزع (Knapsack Problem): تحدد أفضل طريقة لملء حقيبة محددة الوزن بأشياء ذات قيمة محددة. تقوم الخوارزمية بحساب القيم الممكنة لكل وزن فرعي ثم تجمعها لتحقيق أقصى قيمة.

حساب فيبوناتشي: حساب سلسلة فيبوناتشي باستخدام التخزين المؤقت. بدلاً من حساب كل عدد من البداية، يتم تخزين الأعداد المحسوبة مسبقًا لإعادة استخدامها.

مشكلة الأطول تصاعديًا:(Longest Increasing Subsequence) تستخدم لتحديد أطول تسلسل فرعي يتزايد من مصفوفة معينة.

أنواع الخوارزميات

6- خوارزميات البحث(Search Algorithms) : هي مجموعة من الأساليب المستخدمة للعثور على عنصر معين ضمن مجموعة من العناصر. تتنوع خوارزميات البحث في تعقيدها وكفاءتها، وتعتمد الطريقة المثلى على هيكل البيانات وحجمها.

مفهوم البحث الخطي(Linear Search) : البحث الخطي هو أبسط أشكال خوارزميات البحث. يعمل عن طريق فحص كل عنصر في مجموعة البيانات واحدًا تلو الآخر حتى يتم العثور على العنصر المطلوب أو حتى يتم استنفاد جميع العناصر.

خصائص البحث الخطي:

البساطة: يعد البحث الخطي سهل التنفيذ ويستخدم في معظم لغات البرمجة. لا يتطلب أي هيكل بيانات خاص، مما يجعله ملائمًا للبيانات غير المرتبة.

عدم الاعتماد على ترتيب البيانات: يمكن استخدامه مع أي مجموعة من البيانات، سواء كانت مرتبة أو غير مرتبة.

الوقت المستغرق: الزمن المستغرق في البحث الخطي يعتمد على حجم مجموعة البيانات، حيث يكون وقت التنفيذ O(n) في أسوأ الحالات، حيث n هو عدد العناصر في المجموعة.

تعقيد الخوارزميات

التعقيد الزمني: هو قياس عدد العمليات التي تقوم بها الخوارزمية كدالة لحجم المدخلات. يُستخدم عادةً لقياس أداء الخوارزميات في أسوأ الحالات، ومتوسط الحالات، وأفضل الحالات.

O(n) (Time Complexity):هو تعبير يستخدم في تحليل التعقيد الزمني لوصف خوارزمية حيث يزداد الوقت المستغرق لتنفيذ الخوارزمية بشكل خطي مع زيادة حجم المدخلات n

خصائص التعقيد الزمنى :O(n)

1.النمو الخطي: عندما تتزايد قيمة n، يتزايد الوقت المستغرق بشكل خطي. على سبيل المثال، إذا استغرقت الخوارزمية 1 ثانية لمعالجة 10 عناصر، فستستغرق 2 ثانية لمعالجة 20 عنصرًا.

2. تطبيقات :O(n): تتواجد O(n) في العديد من الخوارزميات، مثل البحث الخطي، حيث يتطلب فحص كل عنصر من عناصر المجموعة حتى يتم العثور على العنصر المطلوب.

3. مقارنة مع التعقيدات الأخرى: O(n)أقل كفاءة من O (log n) مثل البحث الثنائي، ولكنه أكثر كفاءة من O(n^2) مثل خوارزمية الفقاعات في أسوأ الحالات.

أمثلة على :O(n)

- البحث الخطي :Linear Search

في هذا النوع من البحث، تحتاج إلى التحقق من كل عنصر في القائمة حتى تجد العنصر المستهدف. إذا كان لديك قائمة تحتوي على nعناصر، فسوف تستغرق الخوارزمية في أسوأ الحالات O(n) من الزمن.

```
def sum_array(arr):
    total = 0
    for number in arr:
        total += number
    return total
```

- حساب مجموع عناصر مصفوفة: في هذا المثال، يتطلب حساب مجموع العناصر O(n) من الزمن، حيث يتم المرور عبر كل عنصر مرة واحدة.

تعقيد الخوارزميات

O (log n) و $O(n^2)$:

1- O(log n) : هو تعبير يستخدم في تحليل تعقيد الخوارزميات للإشارة إلى خوارزمية تزداد فيها الوقت المستغرق بشكل لوغاريتمي مع زيادة حجم المدخلات n. يشير هذا التعقيد إلى أن زمن التنفيذ ينمو بشكل أبطأ بكثير مقارنةً بالنمو الخطي.

```python
def binary_search(arr, target):
    left, right = 0, len(arr) - 1
    while left <= right:
        mid = (left + right) // 2
        if arr[mid] == target:
            return mid  # العنصر موجود
        elif arr[mid] < target:
            left = mid + 1
        else:
            right = mid - 1
    return -1  # العنصر غير موجود
```

خصائص:O(log n)

A. **النمو اللوغاريتمي:** عندما تتزايد قيمةn ، فإن الوقت المستغرق ينمو بشكل بطيء جداً. على سبيل المثال: إذا استغرقت الخوارزمية 1 ثانية لمعالجة 10 عناصر،فإنها قد تستغرق 2 ثانية لمعالجة 100 عنصر، لكن فقط 3 ثواني لمعالجة 1000 عنصر. في هذا المثال، يتم تقسيم قائمة مرتبة إلى نصفين في كل مرة، مما يؤدي إلى تعقيد.O(log n)

B. **تطبيقات:O(log n) :** شائع في خوارزميات البحث التي تقلل عدد العناصر المراد فحصها بشكل متكرر، مثل **البحث الثنائي** (Binary Search)، حيث يتم تقسيم مجموعة البيانات إلى نصفين في كل خطوة.

تعقيد الخوارزميات

2- تعريف:$O(n^2)$: هو تعبير يُستخدم في تحليل تعقيد الخوارزميات للإشارة إلى خوارزمية يزداد فيها الوقت المستغرق بشكل تربيعي مع زيادة حجم المدخلات n.

خصائص:$O(n^2)$

1. **النمو التربيعي:** عندما تتزايد قيمةn ، فإن الوقت المستغرق ينمو بشكل أسرع.
على سبيل المثال، إذا استغرقت الخوارزمية 1 ثانية لمعالجة 10 عناصر، فستستغرق 4 ثوانٍ لمعالجة 20 عنصرًا (لأن $20^2 = 400$).
2. **تطبيقات:$O(n^2)$:** شائع في خوارزميات مثل **خوارزمية الفقاعات** (Bubble Sort) وخوارزمية **الإدراج** (Insertion Sort)، حيث يتطلب الأمر إجراء عمليات مقارنة داخلية تكرارية.

أمثلة على:$O(n^2)$

- **خوارزمية الفقاعات:(Bubble Sort) :** في هذا المثال، يتم تكرار الفحص لجميع العناصر عدة مرات، مما يؤدي إلى تعقيد.$O(n^2)$

```python
def bubble_sort(arr):
    n = len(arr)
    for i in range(n):
        for j in range(0, n-i-1):
            if arr[j] > arr[j+1]:
                arr[j], arr[j+1] = arr[j+1], arr[j]  # تبادل القيم
```

تحليل التعقيد

1- تحليل التعقيد في أسوأ الحالات (Worst-case): تحليل التعقيد في أسوأ الحالات هو أسلوب لتقييم أداء الخوارزمية من خلال قياس الزمن أو الذاكرة المستخدمة في الحالة الأكثر تعقيدًا (أي عندما تكون المدخلات أو الظروف غير مواتية).

أمثلة على تحليل التعقيد في أسوأ الحالات:

A. البحث الخطي:(Linear Search) :

- في أسوأ الحالات، يتعين على الخوارزمية البحث عن العنصر في نهاية المصفوفة أو عدم وجوده.
- التعقيد في أسوأ الحالات$O(n)$:

B. خوارزمية الفقاعات:(Bubble Sort)

- في أسوأ الحالات، يجب أن تقوم الخوارزمية بإجراء عدد كبير من المقارنات، خاصة عندما تكون العناصر مرتبة في ترتيب عكسي.
- التعقيد في أسوأ الحالات$O(n^2)$:

C. البحث الثنائي:(Binary Search)

- في أسوأ الحالات، قد يتطلب الأمر عدة عمليات تقسيم للعثور على العنصر أو تأكيد عدم وجوده.
- التعقيد في أسوأ الحالات $O(\log n)$:

تحليل التعقيد

2- تحليل التعقيد في أفضل الحالات (Best-case Complexity Analysis) : تحليل التعقيد في أفضل الحالات هو أسلوب لتقييم أداء الخوارزمية من خلال قياس الزمن أو الذاكرة المستخدمة في الحالة الأكثر ملاءمة (أي عندما تكون المدخلات أو الظروف مواتية).

أمثلة على تحليل التعقيد في أفضل الحالات:

A. البحث الخطي:(Linear Search)
   - في أفضل الحالات، يتم العثور على العنصر المطلوب في البداية.
   - التعقيد في أفضل الحالات O(1)

B. خوارزمية الفقاعات:(Bubble Sort)
   - في أفضل الحالات، تكون العناصر مرتبة مسبقًا، مما يعني أنه لا يوجد تبادل مطلوب.
   - التعقيد في أفضل الحالات O(n)

C. البحث الثنائي:(Binary Search)
   - في أفضل الحالات، يمكن العثور على العنصر المستهدف في منتصف المصفوفة في أول محاولة.
   - التعقيد في أفضل الحالات O(1)

تحليل التعقيد

3- تحليل التعقيد في الحالات المتوسطة(Average-case Complexity Analysis) : تحليل التعقيد في الحالات المتوسطة هو أسلوب لتقييم أداء الخوارزمية من خلال قياس الزمن أو الذاكرة المستخدمة في الحالات الشائعة أو العادية، بدلاً من التركيز على الأسوأ أو الأفضل.

أمثلة على تحليل التعقيد في الحالات المتوسطة:

A. البحث الخطي:(Linear Search)
   - في المتوسط، يتم العثور على العنصر بعد فحص نصف العناصر في المصفوفة.
   - التعقيد في الحالات المتوسطة(O(n :

B. خوارزمية الفقاعات:(Bubble Sort)
   - في المتوسط، يتطلب الأمر عددًا من المقارنات والتبادلات، مما يجعل الأداء أقل من الأسوأ ولكن أعلى من الأفضل.
   - التعقيد في الحالات المتوسطة($O(n^2)$:

C. البحث الثنائي:(Binary Search)
   - في المتوسط، يتطلب الأمر عدة عمليات تقسيم للعثور على العنصر، مما يعني أن الأداء سيكون جيدًا مع البيانات المرتبة.
   - التعقيد في الحالات المتوسطة(O(log n :

خوارزميات الفرز (Sorting Algorithms)

خوارزميات الفرز هي تقنيات تُستخدم لترتيب عناصر مجموعة أو قائمة بناءً على معايير محددة، مثل الأعداد أو السلاسل النصية. يعد الفرز عملية أساسية في علوم الحاسوب، حيث تُستخدم في مجموعة متنوعة من التطبيقات، من تنظيم البيانات إلى تحسين أداء الخوارزميات الأخرى.

أهمية خوارزميات الفرز:

تحسين البحث: تُساعد بيانات مرتبة في تسريع عمليات البحث، حيث يمكن استخدام خوارزميات البحث الأكثر كفاءة مثل البحث الثنائي.

تسهيل المعالجة: تجعل البيانات المرتبة من السهل تنفيذ العمليات الأخرى مثل الدمج أو التصفية أو التجميع.

تحسين الأداء: يمكن أن تؤدي خوارزميات الفرز إلى تحسين الأداء العام للنظام من خلال تقليل التعقيد الزمني للخوارزميات اللاحقة التي تعتمد على البيانات المرتبة.

توفير الوقت والجهد: توفر خوارزميات الفرز الكثير من الوقت والجهد عند التعامل مع كميات كبيرة من البيانات، مما يجعلها أساسية في التطبيقات التي تتطلب معالجة البيانات بكفاءة.

خوارزميات الفرز Sorting Algorithms

- **أنواع خوارزميات الفرز:**

تتنوع خوارزميات الفرز، ولكل منها مزايا وعيوب، ويعتمد اختيار الخوارزمية المناسبة على مجموعة من العوامل، مثل حجم البيانات ونوعها ومتطلبات الأداء.
ومن بين الأنواع الشائعة لخوارزميات الفرز:

1. **فرز الفقاعات (Bubble Sort):** خوارزمية بسيطة وسهلة الفهم، ولكنها ليست فعالة مع البيانات الكبيرة.
2. **فرز الإدراج (Insertion Sort):** فعالة للبيانات الصغيرة والمترتبة جزئيًا.
3. **فرز الاختيار (Selection Sort):** تعتبر خوارزمية بسيطة، لكنها ليست فعالة بشكل عام.
4. **فرز السريع (Quick Sort):** واحدة من أكثر خوارزميات الفرز كفاءة، تعتمد على التقسيم.
5. **فرز الدمج (Merge Sort):** تعمل على تقسيم البيانات إلى أجزاء أصغر ثم دمجها بشكل مرتب، فعالة بشكل خاص في حالات البيانات الكبيرة.

خوارزمية الفرز الفقاعي Bubble Sort

خطوات عمل الفرز الفقاعي:

A. **البدء من بداية القائمة:** يبدأ الفرز من العنصر الأول في القائمة ويقارن بينه وبين العنصر التالي.

B. **المقارنة والتبديل:** إذا كان العنصر الأول أكبر من العنصر الثاني، يتم تبديلهما.

C. **الانتقال إلى العنصر التالي:** ينتقل إلى العنصر الثاني ويكرر نفس العملية مع العنصر الثالث، وهكذا حتى نهاية القائمة.

D. **التكرار:**

- بعد انتهاء الجولة الأولى، يبدأ الفرز من بداية القائمة مرة أخرى ويكرر العملية.
- تستمر هذه الجولات حتى لا تحدث أي تبديلات، مما يعني أن القائمة مرتبة.

مثال توضيحي:

- قائمة غير مرتبة[5, 3, 8, 4, 2] :
- الجولة الأولى:
  - (5, 3) → [3, 5, 8, 4, 2]
  - (5, 8) → [3, 5, 8, 4, 2]
  - (8, 4) → [3, 5, 4, 8, 2]
  - (8, 2) → [3, 5, 4, 2, 8]
- الجولة الثانية:
  - (3, 5) → [3, 5, 4, 2, 8]
  - (5, 4) → [3, 4, 5, 2, 8]
  - (5, 2) → [3, 4, 2, 5, 8]
  - وهكذا.

خوارزمية فرز الدمج Merge Sort

خوارزمية فرزالدمج تعتمد على مفهوم **التقسيم والحل** (Divide and Conquer)، حيث يتم تقسيم القائمة إلى أجزاء أصغر، ثم دمجها بشكل مرتب. تتضمن العملية مرحلتين رئيسيتين:

التقسيم:(Divide) : يتم تقسيم القائمة إلى نصفين حتى تصبح كل قائمة تحتوي على عنصر واحد فقط. يمكن تصور هذه العملية على أنها تقسيم القائمة إلى قطع أصغر حتى لا يتبقى سوى عناصر فردية.

- على سبيل المثال، إذا كانت القائمة هي [38, 27, 43, 3, 9, 82, 10]، فإن التقسيم سيكون كالتالي:
  - [38, 27, 43, 3, 9, 82, 10] → [38, 27, 43]و[3, 9, 82, 10]
  - [38, 27, 43] → [38]و[27, 43]
  - [27, 43] → [27]و[43]

الحل:(Merge) : بعد تقسيم القائمة إلى قطع فردية، يتم دمج هذه القطع بشكل مرتب. تتم مقارنة العناصر في القطع المختلفة وترتيبها أثناء الدمج.

- على سبيل المثال، بعد الحصول على القوائم الفردية [38] و [27] و [43]، يتم دمجها إلى:
  - [27, 38, 43]
- وهكذا حتى يتم دمج جميع العناصر في قائمة واحدة مرتبة.

العملية النهائية: بعد إتمام عمليات الدمج، ستحصل على قائمة مرتبة نهائيًا.

خوارزمية الفرز السريع Quick Sort

كيفية عمله واختيار النقطة المحورية:(Pivot)

الفرز السريع (Quick Sort) هو خوارزمية فرز فعالة تعتمد على مفهوم **التقسيم**، حيث يتم اختيار عنصر معين من القائمة (يسمى النقطة المحورية أو الـ **Pivot** ويتم إعادة ترتيب العناصر بحيث تكون العناصر الأقل من النقطة المحورية على اليسار والعناصر الأكبر على اليمين.

خطوات عمل الفرز السريع:

A. **اختيار النقطة المحورية:(Pivot Selection) :** يتم اختيار عنصر من القائمة ليكون النقطة المحورية. يمكن اختيار النقطة المحورية بطرق مختلفة، مثل اختيار العنصر الأول، الأخير، أو العنصر العشوائي، أو حتى الوسيط.

B. **التقسيم:(Partitioning)**

- يتم إعادة ترتيب العناصر بحيث تكون العناصر الأصغر من النقطة المحورية إلى اليسار والعناصر الأكبر إلى اليمين.
- بعد هذه العملية، تصبح النقطة المحورية في موقعها النهائي في القائمة المرتبة.

C. **الفرز المتكرر:(Recursive Sorting)**

- يتم تطبيق نفس الخطوات على الجزء الأيسر (العناصر الأقل من النقطة المحورية) والجزء الأيمن (العناصر الأكبر من النقطة المحورية) من القائمة.
- تستمر هذه العملية حتى يتم ترتيب جميع الأجزاء بشكل كامل.

مثال توضيحي:

- لنفترض أن لدينا القائمة[10, 80, 30, 90, 40, 50, 70] :
- إذا اخترنا النقطة المحورية 50:
  - بعد التقسيم: [10, 30, 40] [50][80, 90, 70]
  - ثم نكرر نفس العملية على [10, 30, 40] و.[80, 90, 70]

خوارزميات البحث

1- خوارزمية البحث الخطي Linear Search Algorithm : هي واحدة من أبسط خوارزميات البحث.
تُستخدم للبحث عن عنصر معين في قائمة أو مصفوفة عن طريق فحص كل عنصر من العناصر واحداً تلو الآخر.

خطوات عمل خوارزمية البحث الخطي:

البداية :تبدأ الخوارزمية من العنصر الأول في القائمة. **المقارنة :**تقارن العنصر الحالي بالعنصر الذي نبحث عنه.

النتيجة:

- إذا كان العنصر الحالي هو العنصر المطلوب، يتم إرجاع موقعه (الفهرس).
- إذا لم يكن كذلك، تنتقل الخوارزمية إلى العنصر التالي في القائمة.

التكرار :تستمر العملية حتى يتم العثور على العنصر المطلوب أو حتى يتم الوصول إلى نهاية القائمة.

الانتهاء :إذا تم الوصول إلى نهاية القائمة دون العثور على العنصر، تُرجع الخوارزمية قيمة تشير إلى عدم وجود العنصر.

مثال توضيحي : لنفترض أن لدينا قائمة من الأعداد:
[3, 5, 2, 9, 6] ونريد البحث عن العدد 9. ستقوم الخوارزمية بالتحقق من كل عنصر:

- 3) غير مطابق(
- 5) غير مطابق(
- 2) غير مطابق(
- 9) مطابق(

ستعيد الخوارزمية الفهرس 3 كونه موقع العدد 9.

خوارزميات البحث

2- خوارزمية البحث الثنائي(Binary Search Algorithm) : تعتبر خوارزمية البحث الثنائي من أكثر الخوارزميات كفاءة للبحث عن عنصر معين في قائمة مرتبة.

ترتيب القائمة: يجب أن تكون القائمة مرتبة بشكل تصاعدي أو تنازلي. إذا لم تكن مرتبة، فإن البحث الثنائي لن يعمل بشكل صحيح.

البيانات المخزنة: يمكن استخدام أي نوع من البيانات القابلة للمقارنة (مثل الأعداد، السلاسل، أو الكائنات) طالما يمكن ترتيبها.

فهم الهيكل: يجب أن يكون المستخدم قادرًا على فهم كيفية تقسيم القائمة إلى نصفين والتفاعل مع الفهارس.

خوارزمية البحث الثنائي: كيفية عملها

تحديد النطاق:

- يبدأ البحث من العنصر الأوسط في القائمة. يتم حساب الفهرس الأوسط باستخدام المعادلةmiddle=(low+high)/2 : حيث lowهو الفهرس الأدنى و highهو الفهرس الأعلى في القائمة.

المقارنة: يتم مقارنة العنصر الأوسط بالعنصر الذي نبحث عنه.

- **إذا كان العنصر الأوسط هو العنصر المطلوب،** فإن البحث ينتهي بنجاح.
- **إذا كان العنصر الأوسط أكبر من العنصر المطلوب،** يتم تكرار البحث في النصف الأيسر من القائمة من low إلى middle – 1
- **إذا كان العنصر الأوسط أصغر من العنصر المطلوب،** يتم تكرار البحث في النصف الأيمن من القائمة من middle + 1 إلى high

التكرار: تستمر العملية حتى يتم العثور على العنصر المطلوب أو حتى تصبح lowأكبر من high، مما يعني أن العنصر غير موجود في القائمة.

البرمجة الديناميكية Dynamic Programming

هي تقنية لحل المشاكل الرياضية المعقدة عن طريق تقسيمها إلى مشاكل فرعية أبسط. تستخدم هذه التقنية بشكل واسع في علم الحاسوب، خاصة في تحليل الخوارزميات. تتميز البرمجة الديناميكية بخصائص معينة تجعلها فعالة في حل المشاكل التي تحتوي على هيكلية تكرارية.

خصائص البرمجة الديناميكية:

A. **المشاكل الفرعية المتداخلة:** تتضمن مشاكل فرعية تتداخل، مما يعني أن نفس المشكلة يمكن أن تحل بعدة طرق، مما يؤدي إلى حسابها عدة مرات إذا تم استخدام الطرق التقليدية مثل الخوارزميات العودية.

B. **التخزين المؤقت:(Memoization) :** بدلاً من إعادة حساب النتائج لمشاكل فرعية، يتم تخزين النتائج في جدول أو مصفوفة لاستخدامها لاحقًا، مما يوفر الوقت.

C. **الهيكلة المثلى:** تركز البرمجة الديناميكية على إيجاد الحل الأمثل من خلال حل المشاكل الفرعية بشكل فعال.

مراحل البرمجة الديناميكية:

- **تحديد المشاكل الفرعية:** تحديد كيف يمكن تقسيم المشكلة إلى مشاكل فرعية.
- **تخزين النتائج:** استخدام التخزين المؤقت للاحتفاظ بنتائج المشاكل الفرعية.
- **إعادة استخدام النتائج:** عند الحاجة إلى حل لمشكلة فرعية، يمكن استخدام النتيجة المخزنة بدلاً من إعادة حسابها.
- **بناء الحل:** بعد حساب الحلول للمشاكل الفرعية، يتم بناء الحل النهائي للمشكلة الأصلية.

خوارزميات الجشع Greedy Algorithms

الخوارزميات الجشعة هي تقنيات لحل المشاكل التي تتبع استراتيجية اتخاذ القرارات المحلية المثلى في كل خطوة. تعتمد هذه الخوارزميات على اختيار أفضل خيار متاح في كل مرحلة، على أمل أن يؤدي هذا إلى حل أمثل للمشكلة بشكل عام. تعتبر الخوارزميات الجشعة فعالة وسهلة الفهم، ولكن لا يمكن استخدامها لحل جميع المشاكل، حيث قد لا تؤدي دائمًا إلى الحل الأمثل.

خصائص الخوارزميات الجشعة

1. **اختيار محلي:**
   - تتخذ الخوارزميات الجشعة قرارًا بناءً على الخيار الأفضل المتاح في الوقت الحالي، دون النظر إلى العواقب المستقبلية.
2. **مثالية جزئية:**
   - تعتمد على فكرة أن اتخاذ القرار الأفضل في كل خطوة سيوصلنا إلى الحل الأمثل في النهاية، لكن هذا ليس صحيحًا دائمًا.
3. **الحل التكراري:**
   - يتم تنفيذ الخوارزمية بشكل تكراري، مما يعني أنها ستعيد النظر في الخيارات المتاحة في كل مرحلة.
4. **الحل النهائي:**

في كثير من الأحيان، تكون الخوارزميات الجشعة أكثر كفاءة من الخوارزميات الأخرى، على الرغم من أنها قد لا توفر دائمًا الحل الأمثل

خاتمة

تمثل الخوارزميات المتقدمة أداة قوية في مجال علوم الحاسوب، حيث تلعب دورًا حيويًا في تأمين البيانات، تحسين الحلول، وتعزيز الأداء عبر استخدام تقنيات متطورة.

آمل ان تكونوا قد حققتم الفائدة