حل التكليف الأول (الجزء النظري)

1️⃣ القائمة المرتبطة الأحادية

(Singly Linked List)

هي بنية بيانات تتكون من مجموعة عقد، تحتوي كل عقدة على بيانات، بالإضافة إلى مؤشر واحد فقط يشير إلى العقدة التالية في القائمة.

الاستخدامات:

تُستخدم كأساس لبناء بعض هياكل البيانات الأخرى مثل المكدس (Stack).

مناسبة لإدارة الذاكرة بطريقة بسيطة عندما يكون التحرك في اتجاه واحد فقط.

مفيدة في التطبيقات التي لا تحتاج الرجوع إلى الخلف أثناء التنقل بين العناصر.

المميزات:

استهلاك أقل للذاكرة مقارنة بالقوائم الأخرى، لأن كل عقدة تحتوي على مؤشر واحد فقط.

عمليات الإضافة والحذف، خاصة في بداية القائمة، تتم بسرعة وكفاءة.

العيوب:

لا يمكن التنقل إلا في اتجاه واحد، من البداية إلى النهاية.

الوصول إلى عقدة معينة يتطلب المرور على جميع العقد السابقة لها.

لا يمكن الرجوع إلى العقدة السابقة مباشرة.

2️⃣ القائمة المرتبطة المزدوجة

(Doubly Linked List)

هي قائمة تتكون من عقد، تحتوي كل عقدة على بيانات ومؤشرين: أحدهما يشير إلى العقدة التالية، والآخر يشير إلى العقدة السابقة.

الاستخدامات:

تُستخدم في تطبيقات تحتاج التنقل للأمام والخلف، مثل سجل التصفح في المتصفحات.

تعتمد عليها خاصية التراجع (Undo) والإعادة (Redo) في البرامج.

مفيدة في أنظمة التشغيل لإدارة العمليات والمهام.

المميزات:

تتيح التنقل في الاتجاهين بسهولة.

تسهّل عمليات الحذف والإضافة في أي موقع داخل القائمة دون الحاجة للبحث المطوّل.

الوصول المباشر للعقدة السابقة يجعل التعامل مع البيانات أكثر مرونة.

العيوب:

تستهلك مساحة أكبر من الذاكرة بسبب وجود مؤشرين في كل عقدة.

العمليات البرمجية تصبح أكثر تعقيدًا مقارنة بالقائمة الأحادية.

3. القائمة المرتبطة الدائرية

(Circular Linked List)

هي نوع من القوائم المرتبطة تكون فيها العقدة الأخيرة مرتبطة بالعقدة الأولى، مما يجعل القائمة على شكل دائرة، ويمكن أن تكون أحادية أو مزدوجة.

الاستخدامات:

تُستخدم في أنظمة التشغيل لتنفيذ خوارزميات الجدولة مثل (Round Robin).

مناسبة للتطبيقات التي تعتمد على التكرار المستمر بين العناصر.

تُستخدم في تشغيل الوسائط المتعددة مثل قوائم تشغيل الموسيقى.

المميزات:

يمكن البدء في التنقل من أي عقدة داخل القائمة.

لا توجد نهاية فعلية للقائمة، مما يجعلها مناسبة للتطبيقات الدورية.

فعالة في إدارة الموارد التي تعمل بشكل متكرر.

العيوب:

قد تؤدي إلى حدوث حلقة لا نهائية إذا لم يتم التحكم في عملية التنقل.

تحتاج إلى عناية خاصة عند تحديد نقطة التوقف أثناء المرور على العناصر.

توجد طريقتان أساسيتان لتمثيل الرسوم البيانية في الحاسوب، والفرق بينهما يكمن في كفاءة المساحة والوقت:

١. مصفوفة الجوار (Adjacency Matrix):

المفهوم: هي عبارة عن مصفوفة ثنائية الأبعاد ($N \times N$)، حيث يمثل كل صف وعمود عقدة في الرسم البياني.

المميزات: تمتاز بالسرعة العالية ($O(1)$) عند التأكد من وجود صلة أو "حافة" بين أي نقطتين.

العيوب: تستهلك ذاكرة كبيرة لأنها تحجز مكاناً لجميع الاحتمالات الممكنة (حتى لو لم تكن هناك روابط فعلياً)، مما يجعلها غير فعالة في الرسوم البيانية "المتباعدة" (Sparse Graphs).

٢. قائمة الجوار (Adjacency List):

المفهوم: تعتمد على إنشاء قائمة مرتبطة (Linked List) لكل رأس (Vertex)، تحتوي فقط على العقد المتصلة به مباشرة.

المميزات: فعالة جداً في توفير مساحة التخزين، لأنها لا تخزن إلا العلاقات الموجودة فعلياً.

العيوب: البحث عن علاقة محددة بين نقطتين يستغرق وقتاً أطول مقارنة بالمصفوفة، لأننا قد نحتاج للمرور على القائمة بالكامل.