التكليف 1

الفرق بين:

1. Singly Linked List

2. Circular Linked List

3. Doubly Linked List

◆ أولًا Singly Linked List :

الاستخدامات

- تخزين البيانات بشكل ديناميكي
- تنفيذ Stack وQueue
- عند الحاجة لاستخدام ذاكرة أقل

المميزات(Pros)

- سهلة التنفيذ
- تستهلك ذاكرة أقل
- إدراج وحذف سريع

العيوب(Cons)

- لا يمكن الرجوع للخلف
- البحث أبطأ منArray
- لا يمكن الوصول العشوائي للعناصر

◆ ثانيًا Circular Linked List :

الاستخدامات

- Round Robin Scheduling
- الألعاب (الدوران بين اللاعبين)
- تشغيل القوائم المستمرة

المميزات

- لا توجد قيمة null
- مناسبة للتكرار المستمر
- استغلال أفضل للذاكرة

العيوب

- أصعب في التتبع(Debugging)
- قد تسبب Loop لا نهائي إذا لم تُعالج جيدًا

◆ ثالثًا Doubly Linked List :

الاستخدامات

- Undo / Redo
- المتصفحات(Forward & Backward)
- Music Playlists

المميزات

- يمكن التحرك للأمام والخلف
- الحذف أسهل منSingly

العيوب

- تستهلك ذاكرة أكبر
- أكثر تعقيدًا في التنفيذ

التكليف 2

الفرق بين Adjacency List وAdjacency Matrix في تمثيل الرسوم البيانية(Graph)

◆ أولًا Adjacency List :

التعريف:

هي طريقة لتمثيل الـ Graph باستخدام قائمة (List) لكل رأس(Vertex) ، تحتوي على جميع الرؤوس المتصلة به.

الخصائص:

- **كل Vertex له قائمة جيران.**
- **لا يتم تخزين الحواف غير الموجودة.**

المميزات:

- **تستهلك ذاكرة أقل في الرسوم البيانية قليلة الحواف.(Sparse Graph)**
- **سهلة في استعراض جميع الجيران لرأس معين.**
- **مناسبة للـ Graphs الكبيرة.**

العيوب:

- **التحقق من وجود حافة بين رأسين أبطأ.**
- **التنفيذ أكثر تعقيدًا من الـ Matrix.**

استخدامها عندما:

- **عدد الحواف قليل مقارنة بعدد الرؤوس.**
- **نحتاج إلى كفاءة في الذاكرة.**

◆ ثانيًا Adjacency Matrix :

التعريف:

هي مصفوفة ثنائية الأبعاد (2D Array) تمثل العلاقة بين الرؤوس، حيث:

- **القيمة 1 تعني وجود حافة.**
- **القيمة 0 تعني عدم وجود حافة.**

الخصائص:

- **حجم المصفوفة = عدد الرؤوس × عدد الرؤوس.**
- **يتم تخزين جميع العلاقات سواء كانت موجودة أم لا.**

المميزات:

- **التحقق من وجود حافة بين رأسين سريع جدًا.(O(1))**
- **سهلة الفهم والتنفيذ.**
- **مناسبة للـ Graphs الصغيرة أو الكثيفة.**

العيوب:

- **تستهلك ذاكرة كبيرة.**
- **غير فعّالة مع الرسوم البيانية قليلة الحواف.**

استخدامها عندما:

- **عدد الحواف كبير.(Dense Graph)**
- **نحتاج سرعة عالية في التحقق من وجود الحواف.**