مشروع خوارزميات 2

يتم تناول كل مسألة على التوالي:

المسألة الاولى:

قمنا بتعريف Node ببنية المعطيات"Insert name" واعطاءه parameterهم:

- لتعريف النود اليساري Left
- لتعريف النود اليميني Right
- لتعريف ابعاد المستطيل الممثل بالنود Data

وتم تعريف functions and constructors مناسبة لإكمال المشروع.

- التوابع:

- `getChildren()`: يعيد قائمة العقد الفرعية.

- `setParent(Node<T> parent)`: يعين العقدة الأب ويضيف العقدة الحالية كعقدة فرعية للأب.

- `addChild(T data)`: يضيف عقدة فرعية البيانات المعطاة.

- `addChild(Node<T> child)`: يضيف عقدة موجودة كعقدة فرعية.

- `getData()`: يعيد البيانات المرتبطة بالعقدة.

- `isRoot()`: يتحقق مما إذا كانت العقدة هي الجذر (أي ليس لديها أب).

- `isLeaf()`: يتحقق مما إذا كانت العقدة هي ورقة (أي ليس لديها أطفال).

- `removeParent()`: يزيل الإشارة إلى العقدة الأب.

تابعنا باشاء التابع Import الذي يقوم بتخزين الشجرة في البنية السابقة بعد استخراجها من الملف النصي، للقيام بذلك أنشأنا التابع Filereader الذي يأخذ محتوى الملف النصي لتخزينه في متغير من نوع string، وتقتضي خوارزمية التابع على مسح النص حتى الوصول إلى المحرف')'او')' وتقسيم النص تبعا لذلك ضمن حلقات عودية حتى الوصول الى المحارف التي تمثل nodes الأشجار و تخزينها بالشكل المناسب.

يليه التابع Export الذي يعاكسه بطريقة العمل. حيث يقوم برصف كل node بين ابنتيها واضافة أقواس ضمن حلقة عودية.

ننتقل إلى الطلب الثاني. نقوم تابع Import جديد يوافي المتطلبات الجديدة. التي تقتضي الملف النصي ان يحتوي مستطيلات مرسومة.

.....

قمنا بتعريف فئة عامة باسم Form1.

داخل هذه الفئة، هناك اثنين من الأساليب:import,_Import

الأسلوب Import:

يأخذ وسيطة واحدة، وهي String s.

يتحقق مما إذا كانت السلسلة المدخلة s تمثل ورقة (أي قيمة واحدة).

إذا كانت s ورقة، يحولها إلى عقدة باستخدام دالة stringToNode(s) ويعيدها.

إلا أنه يستدعي الأسلوب Import_بسلسلة الإدخال المعدلة ("(" + s + ")").

الأسلوبImport:

هذا الأسلوب الخاص يأخذ أيضًا وسيطة String s.

يعالج السلسلة المدخلة بشكل متكرر لبناء شجرة ثنائية.

إذا كانت s تمثل ورقة ، يحولها إلى عقدة باستخدام دالة stringToNode(s) ويعيدها.

إلا أنه يقوم بالخطوات التالية:

يقوم بالتحقق من أن جميع الأقواس التي فتحت قد تم إغلاقها

يزيل الأقواس الخارجية من s.

يتكرر عبر الأحرف في s للعثور على موضع أول حرف '|' أو '-' الذي ليس مدمجًا في الأقواس.

يحدد الحرف في تلك المواضع (c).

يقسم السلسلة إلى سلسلتين فرعيتين (sl) و (sr) استنادًا إلى الموضع المعثور عليه.

يستدعي Import(sl)_ وImport(sr)_ بشكل متكرر لبناء الأشجار الفرعية اليسرى واليمنى.

ينشئ عقدة جديدة بالأشجار الفرعية اليسرى واليمنى والحرف c، ويعيدها.

ملخص:

تُنشئ الشيفرة شجرة ثنائية من تمثيل السلسلة المدخلة.

تتعامل مع الأوراق (القيم الفردية) والعقد الداخلية (باستخدام العوامل | أو -).

يُفترض أن دالة stringToNode(s)تحول السلسلة إلى كائن Node.

.....

Export 1:

:**(Export)

- تأخذ دالة(`Export`) عقدة (`Node`) تُسمى `root` كدخل.
- تستدعي الدالة الخاصة `Export_` مع العقدة الجذر.
- إذا كانت نتيجة `Export_` ورقة (مُحددة بواسطة `stringIsLeaf`)، فإنها تعيد النتيجة مباشرة.
- وإلا، تزيل الأقواس المحيطة من النتيجة وتعيد السلسلة المعدلة.

3. **دالة `Export_`**:

- تُنشئ الدالة `Export_` سلسلة تُمثل الشجرة بشكل متكرر.
- إذا كانت العقدة الحالية تساوي `null`، فإنها تعيد `null`.
- إذا كانت العقدة الحالية ورقة، فإنها تعيد البيانات كسلسلة.
- وإلا، فإنها تُنشئ سلسلة بالتنسيق `(بيانات العقدة الفرعية اليسرى نوع العقدة الفرعية اليمنى)`.

4. **دالة تحليل العدد الصحيح (parseInteger)**:

- تحلل الأعداد الصحيحة من سلسلة.
- تستخدم التعبيرات العادية لاستخراج الintمنString

الintالمستخرج من من التابع يخزن ب ArrayList.

.....

1. **الطريقة `Import`**:

- تأخذ الطريقة `Import` كائن `Grid` كإدخال وتعيد كائنًا `Node`.

- إذا لم يكن هناك فواصل صف أو عمود (أي لا توجد خطوط أفقية أو عمودية)، فإنها تنشئ `Node` جديدًا بكامل بيانات `Grid`.

- وإلا، فإنها تقسم `Grid` بشكل متكرر إلى جزئين استنادًا إلى فاصل الصف أو العمود وتنشئ `Node` باستخدام البيانات المقسمة.

2. **الطريقة `lineIdxToBreak`**:

- تحدد هذه الطريقة مؤشر فاصل الصف أو العمود في `Grid`.

- تحقق مما إذا كان يجب عكس `Grid` (استنادًا إلى نوع الخط).

- ثم تتكرر عبر الصفوف (أو الأعمدة) وتحقق مما إذا كانت زوج الخلايا المجاورة يحتوي على الأحرف الصحيحة (إما `"+"` أو حرف الفاصل المحدد).

- إذا وجدت فاصلًا صالحًا، فإنها تعيد المؤشر؛ وإلا، تعيد `null`.

3. **الطريقة `cutItAt`**:

- تقسم هذه الطريقة `Grid` قبل القطع إلى جزئين استنادًا إلى المؤشر المعطى ونوع الخط.

- إذا كان الخط صفًا، فإنها تضيف صفوفًا إلى الجزء الأول (`out[0]`) حتى المؤشر وتضيف الصفوف المتبقية إلى الجزء الثاني (`out[1]`).

- إذا كان الخط عمودًا، فإنها تعكس `Grid` قبل القطع، وتقوم بتقسيم مماثل، ثم تعكس كلا الجزئين مرة أخرى.

- تعيد الطريقة مصفوفة تحتوي على كائنين `Grid` مقسمين.

4. **الإخراج للتصحيح**:

- تم تضمين عبارات تصحيح مؤقتة لطباعة معلومات حول عملية القطع.

- يتم استدعاء الطريقة `print()` على كائنات `Grid` لعرض محتوياتها.

بشكل عام، hن هذه الشيفرة جزءًا من خوارزمية تتكرر لتحليل وتقسيم هيكل شبيه بالشبكة. إنها تحدد فواصل الصف أو العمود وتنشئ هيكلًا شبيهًا بشجرة من العقد استنادًا إلى التقسيمات. يعتمد الغرض والس

.....

1. **الطريقة `Export`**:

- تأخذ الطريقة `Export` كائنًا `Node` كدخل وتعيد كائنًا `Grid`.

- إذا كان الجذر `root` مساويًا لـ `null`، فإنها تعيد `null`.

- إذا كان الجذر عقدة ورقية (أي أنه ليس لديه أطفال)، فإنها تستدعي الطريقة `gridBuilder()` على `root.data` وتعيد الـ `Grid` الناتج.

- وإلا، فإنها تستدعي الطريقة `Merger` مع الأطفال اليسرى واليمنى للجذر ونوع البيانات `root.data.type`.

2. **الطريقة `Merger`**:

- تأخذ هذه الطريقة كائنين (`a` و `b`) `Grid` ونوعًا `char` كدخل.

- إذا كان أي من `a` أو `b` يساوي `null`، فإنها تعيد `null`.

- تحسب الأبعاد (الصفوف والأعمدة) لكل من `a` و `b`.

- إذا كان النوع هو `'|'` (دمج عمودي)، فإنها تنشئ `Grid` جديدًا بنفس عدد الصفوف وعدد الأعمدة المجموعة من `a` و `b`. تملأ الشبكة الجديدة بدمج العناصر المقابلة من `a` و `b`.

- إذا كان النوع هو `'-'` (دمج أفقي)، فإنها تنشئ `Grid` جديدًا بعدد الصفوف المجموع ونفس عدد الأعمدة كـ `a`. تملأ الشبكة الجديدة بدمج العناصر المقابلة من `a` و `b`.

3. **ملاحظات**:

- تُستخدم عبارات `assert` للتحقق، مضمنة التأكد من تطابق الأبعاد كما هو متوقع.

- تُستخدم عبارات `continue` للانتقال إلى التكرار التالي في الحلقة.

بشكل عام يبدو hن هذه الشيفرة تتعامل مع دمج شبكتين استنادًا إلى النوع المحدد ('|' أو '-'). تقوم الطريقة `Merger` بدمج الشبكات، وتقوم الطريقة `Export` بمعالجة الهيكل الشجري بشكل متكرر لأداء الدمج.

.....

1. **البناء `Grid()`**:

- يبدأ شبكة فارغة (قائمة من قوائم الأحرف).

2. **البناء `Grid(int rowsCount, int colsCount)`**:

- ينشئ شبكة بعدد محدد من الصفوف والأعمدة.

- يبدأ كل خلية بحرف افتراضي (عادة مساحة `' '`).

3. **البناء `Grid(String input)`**:

- ينشئ شبكة من سلسلة الإدخال.

- تمثل سلسلة الإدخال الشبكة، حيث يتوافق كل حرف مع خلية.

- يتم إنشاء صفوف جديدة عند مواجهة الحرف `\n` (السطر الجديد).

4. **الطريقة `getRowsCount()`**:

- تُرجع عدد الصفوف في الشبكة.

5. **الطريقة `getColsCount()`**:

- تُرجع عدد الأعمدة في الشبكة.

- إذا كانت الشبكة فارغة، فإنها تُرجع 0.

6. **البناء `Grid(int initialCapacity)'**:

- ينشئ شبكة بسعة مبدئية محددة.

7. **البناء (`Grid(ArrayList arrayList`)**:

- ينشئ شبكة من `ArrayList` موجود.

8. **الطريقة `print()`**:

- يطبع الشبكة عن طريق التكرار عبر كل صف واستدعاء `PrintingArrayList.printCharArray(ac`) لكل صف.

- ملاحظة: فئة `PrintingArrayList` غير موجودة هنا، لكنها على الأرجح تطبع الأحرف في الصف.

9. **الطريقة `invert()`**:

- ينشئ شبكة جديدة تُسمى `invertedGrid`.

- ينسخ العناصر من الشبكة الأصلية (`this`) إلى `invertedGrid` بطريقة معكوسة (تبديل الصفوف والأعمدة).

- يُفرغ الشبكة الأصلية ويُستبدل محتواها بالشبكة المعكوسة.

.....

هذا البرنامج في لغة جافا يتبع الحزمة `Problem1.SubProblem4` ويقوم بمعالجة العقد التي تمثل المستطيلات للعثور على الأشجار الصالحة. إليك نظرة موجزة على وظائفه:

1. **الطريقة الرئيسية**:

   - تقوم بقراءة المدخلات من ملف باستخدام `Filereader.stringreader`.
   - تحوّل المدخلات إلى قائمة من كائنات `Node` باستخدام `Import`.
   - تبحث عن الأشجار الصالحة من قائمة العقد باستخدام `validTrees`.
   - تطبع كل شجرة صالحة وتصدّرها باستخدام `Form2.Export`.

2. **طريقة validTrees**:

   - تنشئ خريطتين لتصنيف العقد حسب العرض والطول.
   - تملأ هذه الخرائط بالعقد المعطاة.
   - تستدعي `countFromMaps` للبحث عن الأشجار الصالحة وحساب عددها، ثم تعيد قائمة الأشجار الصالحة.

3. **طريقة countFromMaps**:

   - الحالة الأساسية: إذا كان هناك عقد واحد فقط، يتحقق من صحته ويضيفه إلى قائمة النتائج، ثم يعيد.
   - الحالة التكرارية: يحاول دمج العقد حسب العرض والطول باستخدام `mergeAndCount`، ويتراكم عدد الأشجار الصالحة.

4. **طريقة mergeAndCount**:

   - تكرر عبر كل زوج من العقد في الخريطة الأساسية.
   - تدمجهما وتنشئ نسخًا عميقة من الخريطتين الأساسية والثانوية.
   - تزيل العقد الأصلية وتضيف العقد المدموج إلى الخرائط الجديدة.
   - تستدعي `countFromMaps` بالخرائط المحدثة وتحدث عدد الأشجار الصالحة.

5. **الطرق المساعدة**:

   - `deepCopyMap` تنشئ نسخة عميقة من خرائط العقد.
   - `removeNode` تزيل عقدًا من الخريطة.
   - `addNode` تضيف عقدًا إلى الخريطة استنادًا إلى العرض أو الطول.
   - `hasOneNode` تحقق مما إذا كانت الخريطة تحتوي على عقد واحد فقط.
   - `Import` تحوّل السلاسل المدخلة إلى كائنات `Node` باستخدام `Form1.stringToNode`.