الجمهورية العربيـة السـورية

المعهد العالي للعلوم التطبيقية والتكنولوجيا

اختصاص شبكات ونظم تشغيل

العام الدراسي 2022/2023

عملي البرمجة المتوازية

البرمجة المتوازية باستخدام Java

تقديم

بشار حسين

تاريخ: 21/03/2023

السؤال الأول:

قمنا بتنفيذ المطلوب عن طريق التعديل على توابع AraaySum الموجودة في ملف الجلسة. لم نتمكن من تنفيذ المطلوب على مصفوفة بحجم 10^9 بسبب مشاكل الذاكرة علماً أننا طبقنا الحلول الشائعة من تغيير إعدادات الذاكرة Xmx وزيادتها لتلائم الحجم المطلوب ولكن على مايبدو أن المشكلة هي حجم RAM الخاص بحاسبي لا يكفي.

النتائج:

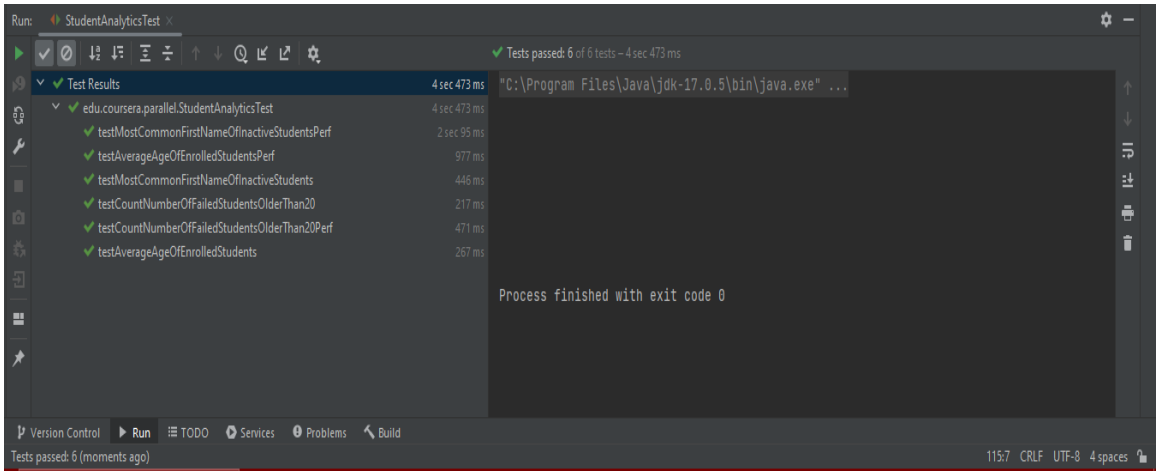
```
"C:\Program Files\Java\jdk-17.0.5\bin\java.exe" -ea -Didea.test.cyclic.buffer.size=1048576 "-javaagent:C:\Pr
Parallel Time execution for Random Array of size 100000000 is 50 ms number of frequency of 7 is 1
Sequential Time execution for Random Array of size 100000000 is 101 ms number of frequency of 7 is 1
Parallel Stream Time execution for Random Array of size 100000000 is 85 ms number of frequency of 7 is 1
Sequential Stream Time execution for Random Array of size 100000000 is 275 ms number of frequency of 7 is 1

Process finished with exit code 0
```

يمكن مراجعة الكود ضمن المجلد المسمى First&Third_Questions ضمن الصف ArraySum.

السؤال الثاني:

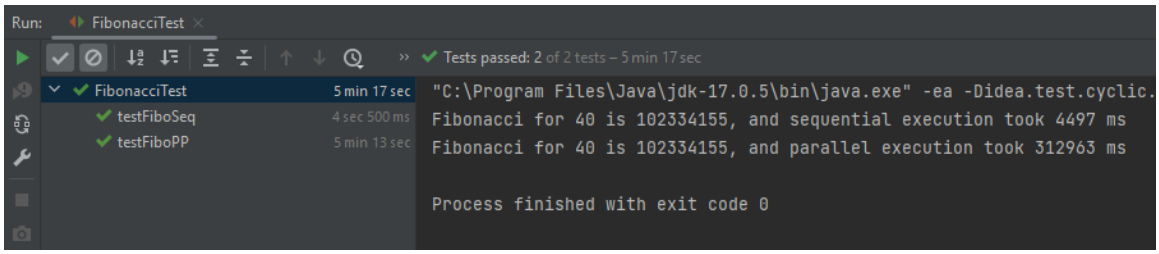
قمنا بتنفيذ المطلوب ويمكن مراجعة الكود ضمن المجلد المسمى Second_Question وهذه كانت نتائج التنفيذ:



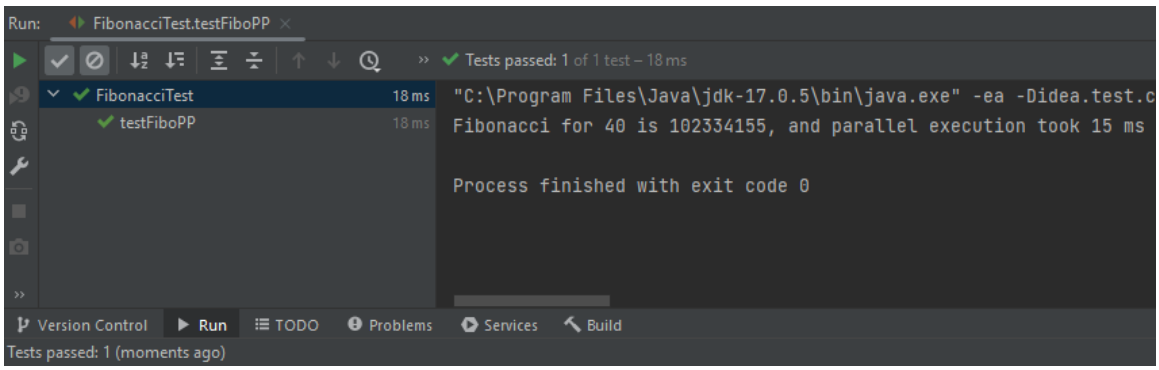
السؤال الثالث:

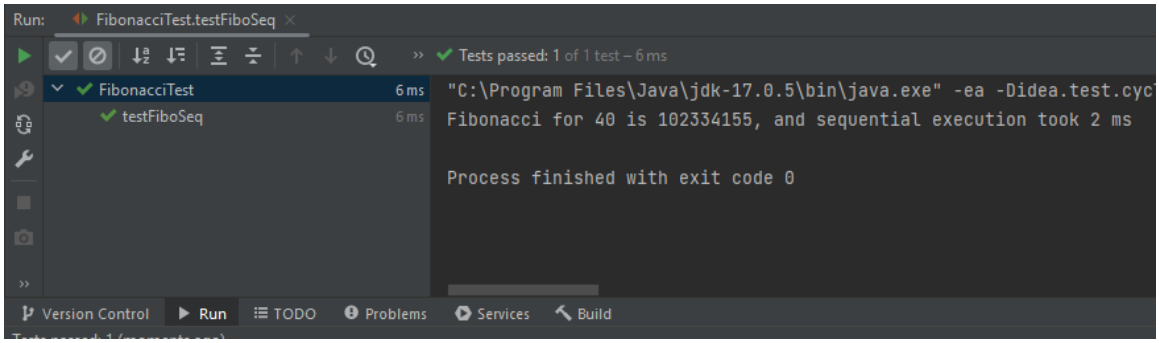
قمنا بتنفيذ المطلوب وإضافة Hashmap إلى الكود المأخوذ في الجلسة ولاحظنا وجود تحسن رهيب في النتيجة. قمنا بإضافة Hashmap وجعلناها static كي لا تتغير في كل مرة نخلق فيها غرض جديد وتضيع علينا معلومات التخزين.

نتيجة الاختبار من دون Hashmap



نتيجة الاختبار مع تطبيق Hashmap





تفسير النتائج:

تعقيد الحساب التسلسلي في المرة الأولى كان هو نفسه عدد فيبوناتشي أي في حالتنا التي هي $n = 40$ يكون التعقيد (عدد العمليات التي سينفذها تابع حساب فيبوناتشي التسلسلي) هو $fib(40)$ وهو من رتبة

10^8 وهذا يستغرق تقريباً ثانية في لغة C++ وكون لغة جافا أبطأ وبالإضافة إلى العمليات الأخرى التي لم نحسبها في التعقيد فيبدو زمن التنفيذ البالغ 5 ثواني منطقي جداً.

أما في حالة الحساب على التوازي فنلاحظ أنها أبطأ بكثير والسبب يعود إلى فتح عدد كبير جداً من النياسب يتناسب مع fib(n-20) (لقد طرحنا 20 لأننا نتوقف عن إنشاء نياسب عندما n أصغر او تساوي 20)أي في حالتنا هذه (n = 40) سننشئ مايقارب fib(20) نيسب أي 6765. وهناك وقت مستغرق لفتح نيسب جديد وإغلاقه وهذا يزيد وقت التنفيذ جداً. بالإضافة إلى ذلك، ليس من المنطقي جداً استخدام البرمجة المتوازية في هذه الحالة حيث أننا سنتخدم نياسب مختلفة لحساب fib الخاصة بنفس n.

لم نصادف هذه المشكلة في حالة مجموع مصفوفة لأن عدد النياسب المتشكلة كان قليل وهو يتناسب مع ceil(length of the array/10^6) أي في حالتنا التي كانت المصفوفة طولها 10^8 سيتشكل تقريبا 100 نيسب فقط. وتمت القسمة على 10^6 لأننا ضمن الكود نقوم بإيقاف إنشاء نياسب جديدة عندما يصبح طول المجال أصغر من 10^6.

في حالة استخدام HashMap يصبح التنفيذ المتسلسل تعقيده فقط هو عدد القيم التي سنملؤها في HashMap وهو هنا 40 قيمة فحسب بالتالي يصبح التعقيد لا يذكر. طبعاً يضرب التعقيد بتعقيد الإدخال والقراءة من HashMap والذي هو أصلا O(1).

ولا تزال حالة Parallel أبطأ بقليل من حالة sequential بسبب الوقت اللازم لفتح النياسب.