Минимална описана окръжност

Проект по системи за паралелна обработка

Изготвили: Калин Маринов, 81051

Кристиана Неделчева, 81052

Проверил: ……………………………………………………….

/ас. Христо Христов/

Съдържание

[Съдържание 2](#_Toc454669714)

[Условия на задачата 3](#_Toc454669715)

[Анализ и свойства 3](#_Toc454669716)

[Инкрементално решение 3](#_Toc454669717)

[Намиране на окръжност – наивен подход 4](#_Toc454669718)

[Архитектура на приложението и реализация на алгоритъма 4](#_Toc454669719)

[Проведени тестове и измервания 5](#_Toc454669720)

[Използвани материали 7](#_Toc454669721)

## Условия на задачата

Разглеждаме множество от точки в равнината. Точките имат целочислени координати в интервала [0, 47483647]. Съществуват безброй много окръжности в равнината, съдържащи това множество от точки.

Вашата задача е да напишете програма намираща минималната, описана около множеството от точки окръжност. Програмата намира минималната описана окръжност използвайки паралелни процеси (нишки).

## Анализ и свойства

* Съществува единствена минимална описана окръжност за дадено множество от точки
* Минималната окръжност винаги минава през две или три точки
  + В противен случай би съществувала друга покриваща окръжност с по-малък радиус
* При минимална окръжност инцидентна на две точки, тя има диаметър равен на разстоянието между точките
* Когато окръжността е дефинирана от три точки – тя представлява окръжността описана около триъгълника формиран от трите точки
* За всяка намерена окръжност през две точки, съдържаща всички останали – твърдим че тя е минимална.
  + Никоя окръжност с радиус по-малък от намерената не може да съдържа двете точки
  + Следователно при откриване такава окръжност можем да сме сигурни че тя е единственото решение.

## Инкрементално решение

Ще използваме итеративен подход, като на K-тата итерация ще намираме окръжността покриваща първите K на брой точки. Тази окръжност ще означим с Ck. Да разгледаме действието на алгоритъма на K-тата итерация:

Нека с означим първите точки и е намерената окръжност от предишната итерация, съдържаща това множество от точки.   
В зависимост от позицията на новата точка :

* Ако е вътрешна за , то (няма промяна в решението)
* В противен случай търсим нова окръжност за минаваща през точката

## Намиране на окръжност – наивен подход

След като знаем че търсената окръжност минава през дадена точка , можем да създадем алгоритъм, който генерира всички окръжности през две и три точки, като една от тях е .

Така постигаме високо ниво на паралелност на програмата, въпреки че избрания подход намиране всички комбинации от окръжности.

На K-тата итерация, сложността на алгоритъма за обхождане на окръжности през и една от другите точки, включително проверка на условието за минималност на всяка окръжност е:

Сложността за генериране на всички окръжности и проверка за минималност през и две други точки е съответно:

Важно е да се отбележи че при намиране на една окръжност през две точки, съдържаща всички съществуващи точки – тя е минимална, следователно можем да прекратим търсенето. Същото обаче не може да се твърди за окръжност през три точки.

При K-тата итерация вероятността точката да е извън намерената окръжност от предходната итерация е: . В случай че точката е вътрешна за окръжността, не е необходимо търсенето на нова окръжност – решението се запазва и сложността е O(1).

Така общата сложността на алгоритъм с итерации е:

# Архитектура на приложението и реализация на алгоритъма

За написването на приложението е използван езикът C# и Visual Studio 2015. За да може да се тества на тестовата машина на Linuxсе компилира кодът посредством .NET Core. За изпълнението на програмата под Linux използвахме Docker.

* Библиотеката SmallestCircle.Data се използва за
  + Итерация на точките (от файл или с генератор на произволни такива)
  + Моделиране на обектите точка и окръжност
* Библиотеката SmallestCircle.Calculation има за цел
  + Дефиниране на геометрични свойства – разстояние между точки, принадлежност към окръжност на точка или списък от точки
  + Изчисляване на окръжност – през две или три точки
  + Калкулация на минимална окръжност
    - Calculator class – калкулира минималната окръжност чрез итеративен подход, използва една нишка, без възможност за асинхронно четене на файла.
    - MultiCalculator class – Multi-threading и асинхронен вариант на алгоритъма от Calculator класа
    - DemoCalculator class – служи за единствено за демонстрация, принтиране на състоянието на нишките и онагледяване на алгоритъма на графичния интерфейс
  + SmallestCircle.Console – конзолна апликация за стартиране на проекта
  + SmallestCircle.Presentation – WPF апликация за демонстрация на работата на алгоритъма
    - Възможност за подване на точките през визуален интерфейс
    - Възможност за експортиране на точките
    - Демонстрация на многонишковия калкулатор с възможност спиране и продължаване

# Проведени тестове и измервания

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function Name** | **Inclusive Samples** | **Exclusive Samples** | **Inclusive Samples %** | **Exclusive Samples %** |
| Main | 5715 | 0 | 99,9 | 0 |
| Calculator | 5702 | 0 | 99,67 | 0 |
| CalculateCircle | 5700 | 10 | 99,63 | 0,17 |
| FindCircleCombination | 5609 | 259 | 98,04 | 4,53 |
| FromThreePoints | 2372 | 1250 | 41,46 | 21,85 |
| ContainsPoint | 2024 | 621 | 41,46 | 10,85 |
| DistanceTo | 1711 | 1120 | 35,38 | 19,58 |

В таблицата е показана йерархията на извикване на методите. От нея можем да забележим, че FindCircleCombination е 98% от програмата. Ще използваме този процент като p = 0.98 за закона на Амдал, който ще използваме в следващата таблица. За теоретично ускорение ще използваме броя на използваните процесорните ядра, тъй като в оптимални условия ще имаме толкова голямо ускорение, колкото повече процесорни ядра използваме.

Резултатите в долната таблица са направени на предоставения Linux – CentOS 6.8 с процесор Intel Xeon Processor X5650. По време на тестовете на програмата й беше изключен визуални интерфейс и й беше включен“тихия” режим. Тестовете на Windows са направени на процесор Intel Core i7-4770K. Програмата е тествана с файл с големина 182KB, в който има 10000 точки.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Брой ядра на процесора** | **Време за изпълнение в ms (Linux)** | **Време за изпълнение**  **в ms (Windows)** | **Ускорение Linux** | **Очаквано ускорение (от закона на Амдал)** | **Ефективност** |
| 1 | 8874 | 637 | 1 | 1 | 1 |
| 2 | 4543 | 350 | 1,953 | 1,961 | 0,977 |
| 4 | 1525 | 215 | 5,819 | 3,774 | 1,455 |
| 8 | 1338 | 194 | 6,632 | 7,018 | 0,829 |
| 12 | 1262 | - | 7,032 | 9,836 | 0,586 |
| 16 | 1105 | - | 8,031 | 12,308 | 0,502 |
| 24 | 1002 | - | 8,856 | 16,438 | 0,369 |

## Използвани материали

[SEC] <http://www.cs.uu.nl/docs/vakken/ga/slides4b.pdf>

[SECIO] <https://www.nayuki.io/page/smallest-enclosing-circle>