**Trường Đại Học Tài Nguyên Và Môi Trường**



**TP Hồ Chí Minh**

---------------------------------------------------------------



**BÁO CÁO ĐỒ ÁN**

**Ứng Dụng Trí Tuệ Nhân Tạo**

**Trong việc Phát triển Game**

---------------------------------------------------------------

Giảng Viên Hướng Dẫn : **Phạm Minh Khan**

Nhóm Sinh viên thực hiện: **Nhóm 7**

**Đoàn Ngọc Sơn 0950080021**

**Nguyễn Đình Huy 0950080033**

**Võ Ngọc Bảo 0950080026**

Môn học: Trí Tuệ Nhân Tạo

Lớp: 09-ĐH-THMT

***TP. Hồ Chí Minh,20 tháng 11 năm 2023***

**MỤC LỤC**

[**I.** **Giới thiệu và Apache Spark** 5](#_Toc151485811)

[**1.** **Tổng quan về Spark** 5](#_Toc151485812)

[**2.** **Tổng quan về MLlib** 5](#_Toc151485813)

[**3.** **Ưu điểm của Apache Spark** 5](#_Toc151485814)

[**II.** **Cài đặt Apache Spark** 7](#_Toc151485815)

[**1.** **Cài đặt JDK 11** 7](#_Toc151485816)

[**2.** **Cài đặt Spark** 7](#_Toc151485817)

[**III.** **Demo thực nghiệm** 11](#_Toc151485818)

[Cấu Trúc Dữ Liệu Chính 12](#_Toc151485819)

[Xử lí bài Toán 12](#_Toc151485820)

[Giới thiệu Sơ lược thuật toán đối với Game 12](#_Toc151485821)

[Định Nghĩa Bài Toán 13](#_Toc151485822)

[Cải Tiến Thuật Toán 16](#_Toc151485823)

[Thống Kê và Kết Luận 17](#_Toc151485824)

[**BẢNG PHÂN CÔNG CÔNG VIỆC** 19](#_Toc151485825)

**NHẬN XÉT CỦA GIẢNG VIÊN**

....................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................

# **Giới thiệu và Apache Spark**

## **Tổng quan về Spark**

Apache Spark là một framework điện toán cụm (cluster-computing) mã nguồn mở được phát triển sơ khởi vào năm 2009 bởi AMPLab tại đại học California. Sau này, Spark đã được trao cho Apache Software Foundation vào năm 2013 và được phát triển cho đến nay. Spark cung cấp các API cho nhiều ngôn ngữ bậc cao như Java, Scala, Python hay R và hỗ trợ engine được tối ưu hóa cho việc thực thi trên đồ thị. Nó cho phép xây dựng các mô hình dự đoán nhanh chóng với việc tính toán được thực hiện trên một nhóm các máy tính, có thể tính toán cùng lúc trên toàn bộ tập dữ liệu mà không cần phải trích xuất mẫu tính toán thử nghiệm. Tốc độ xử lý của Spark có được do việc tính toán được thực hiện cùng lúc trên nhiều máy khác nhau. Đồng thời việc tính toán được thực hiện ở bộ nhớ trong (in-memories) hay thực hiện hoàn toàn trên RAM.

## **Tổng quan về MLlib**

MLlib là thư viện máy học của Spark. MLlib được thiết kế để đơn giản hóa việc xử lý máy học, khả năng mở rộng và tích hợp dễ dàng với các công cụ khác. Với khả năng mở rộng, khả năng tương thích ngôn ngữ và tốc độ của Spark, các nhà khoa học dữ liệu có thể tập trung vào các vấn đề về dữ liệu và mô hình thay vì giải quyết sự phức tạp xung quanh dữ liệu phân tán (như cơ sở hạ tầng, cấu hình, v.v.)

Được xây dựng dựa trên Spark Core, MLlib cung cấp các thuật toán máy học cơ sở cho các bài toán phân lớp, phân cụm, hồi quy và lọc cộng tác. Ngoài ra, MLlib cung cấp các chức năng giúp xử lý trên dữ liệu lớn như: trích rút thuộc tính, biến đổi, giảm và chọn thuộc tính, giảm số chiều dữ liệu, tăng tốc xử lý, các thư viện đại số tuyến tính, thống kê và xử lý dữ liệu, …

## **Ưu điểm của Apache Spark**

***Tốc độ***

Apache Spark cực kỳ phổ biến với các nhà khoa học dữ liệu vì tốc độ của nó. Spark nhanh hơn 100 lần so với Hadoop khi xử lý dữ liệu quy mô lớn. Apache Spark sử dụng hệ thống tính toán trong bộ nhớ (RAM) trong khi Hadoop sử dụng không gian bộ nhớ cục bộ để lưu trữ dữ liệu. Spark có thể xử lý lên tới nhiều petabyte dữ liệu cụm của hơn 8000 node cùng một lúc.

***Dễ sử dụng***

* Apache Spark cung cấp nhiều API tương thích với nhiều ngôn ngữ lập trình.
* Cung cấp nhiều công cụ phân tích
* Ngoài Map và Reduce, Spark còn hỗ trợ máy học, thuật toán đồ thị, truyền dữ liệu, truy vấn SQL, v.v.

***Xử lý dữ liệu realtime***

Realtime đang dần trở thành tính năng tiêu chuẩn của các ứng dụng hiện nay. Một giải pháp xử lý dữ liệu lớn tức thì đang được sử dụng rộng rãi bởi nhiều nền tảng nổi tiếng như Uber, Netflix, Pinterest là *Spark Streaming*, một phần mở rộng của Spark với tính khả mở và chịu lỗi cao cho phép xử lý big data realtime. Trong phân tích dữ liệu, Spark Streaming đóng vai trò cung cấp nền tảng để đẩy dữ liệu vào các mô hình phân tích tức thời, tăng hiệu năng của mô hình.

***Xử lý đồ thị***

Grapx là nền tảng xử lý đồ thị dựa trên Spark, ​​giúp ta dễ dàng chạy phân tích trên dữ liệu đồ thị với các toán tử và thuật toán built-in. Nó cũng cho phép chúng ta lưu trữ và giải nén dữ liệu biểu đồ để tránh phải tính toán lại.

***Cộng đồng sử dụng lớn***

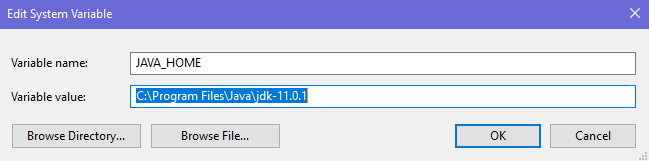
Do Spark là nguồn mở, cộng đồng rất lớn của nó đem lại rất nhiều lợi ích cho người dùng Spark.

# **Cài đặt Apache Spark**

## **Cài đặt JDK 11**

Tải bản cài đặt Java 11 trên trang web: <https://www.oracle.com/java/technologies/javase-jdk11-downloads.html>

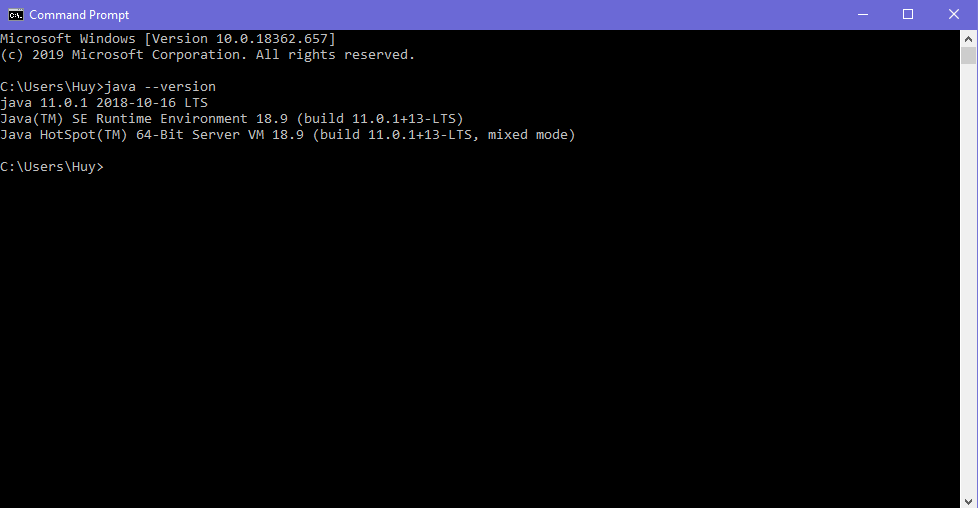
Sau khi cài đặt, set biến môi trường JAVA\_HOME.



Thêm đường dẫn sau vào Path

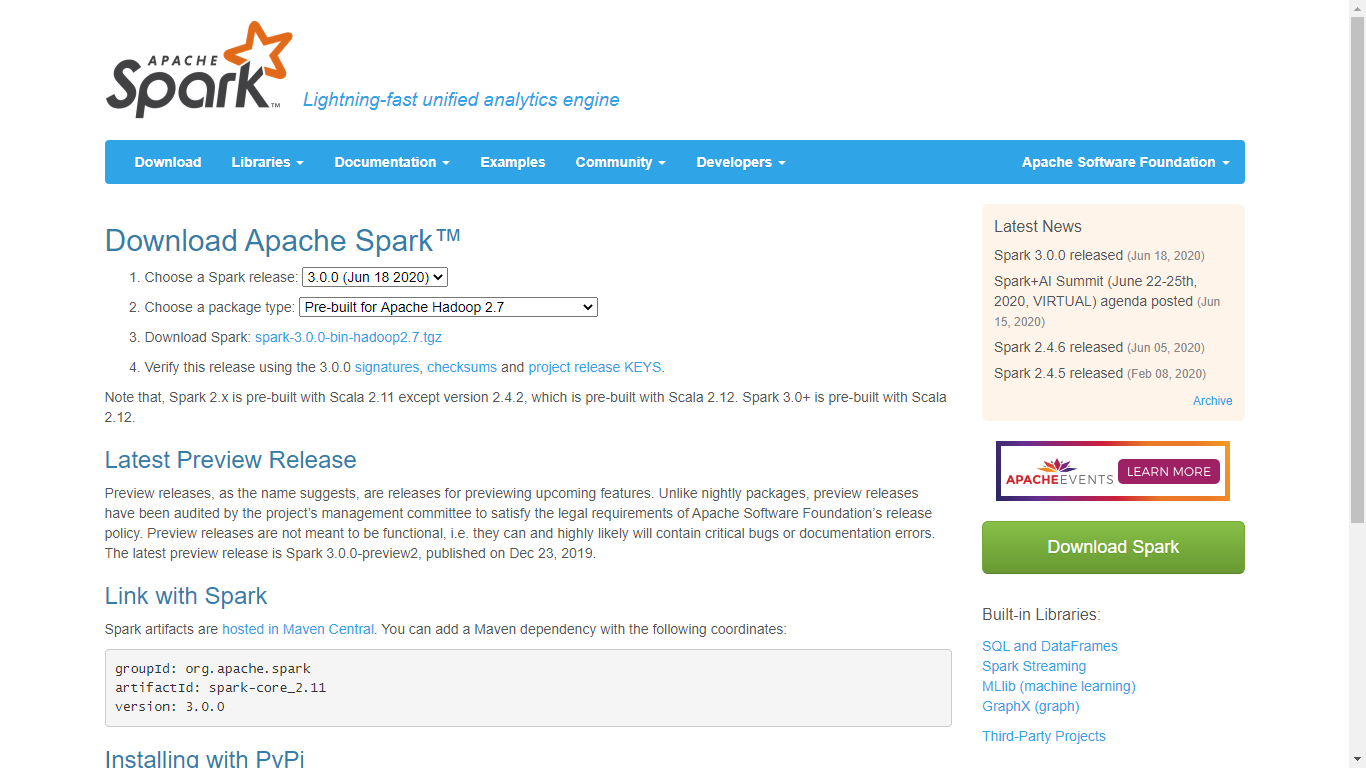


Kiểm tra version của Java bằng dòng lệnh *java --version*

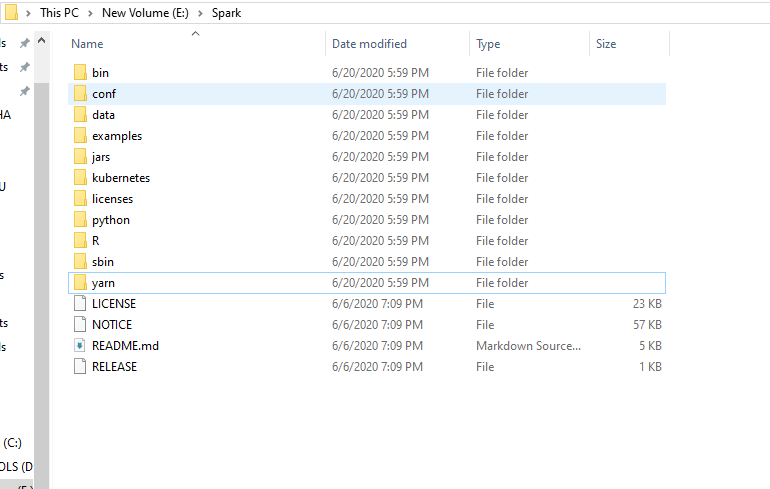


## **Cài đặt Spark**

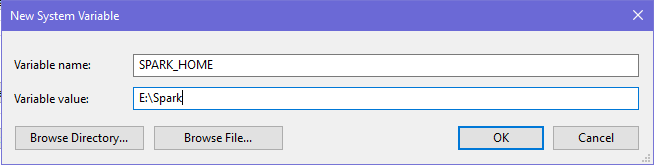
Tải bản Prebuild mới nhất của Spark trên trang web chính thức <https://spark.apache.org/downloads.html>



Sau khi tải về giải nén file đã tải vào thư mục bất kì.



Tạo biến môi trường SPARK\_HOME với đường dẫn là folder vừa giải nén

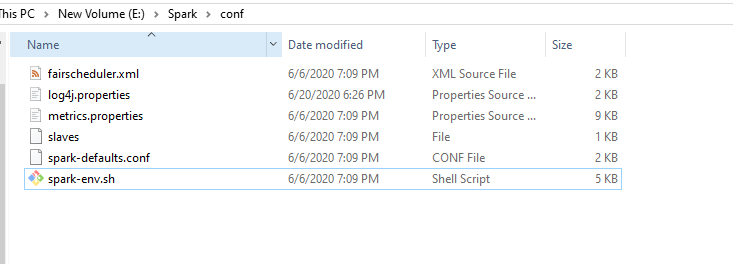


Thêm đường dẫn sau vào Path



Vào thư mục conf trong thư mục Spark, mở file log4j.properties.template, thay thế INFO bằng WARN

Xóa đuôi .template của tất cả file trong thư mục conf

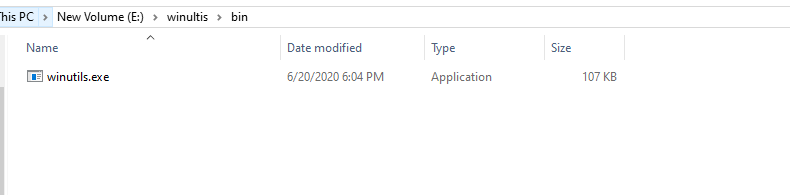


Cài đặt winutils cho Hadoop 2.7

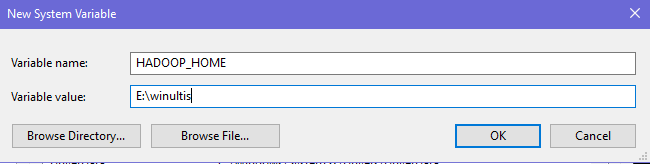
Để sử dụng được Spark, trước tiên ta tải *winutils.exe* của Hadoop 2.7 từ website:

<https://github.com/steveloughran/winutils/blob/master/hadoop-2.7.1/bin/winutils.exe>

Tạo 1 thư mục tên winutils, trong thư mục đó tạo 1 thư mục con tên bin, sau đó sao chép file vừa tải vào thư mục bin



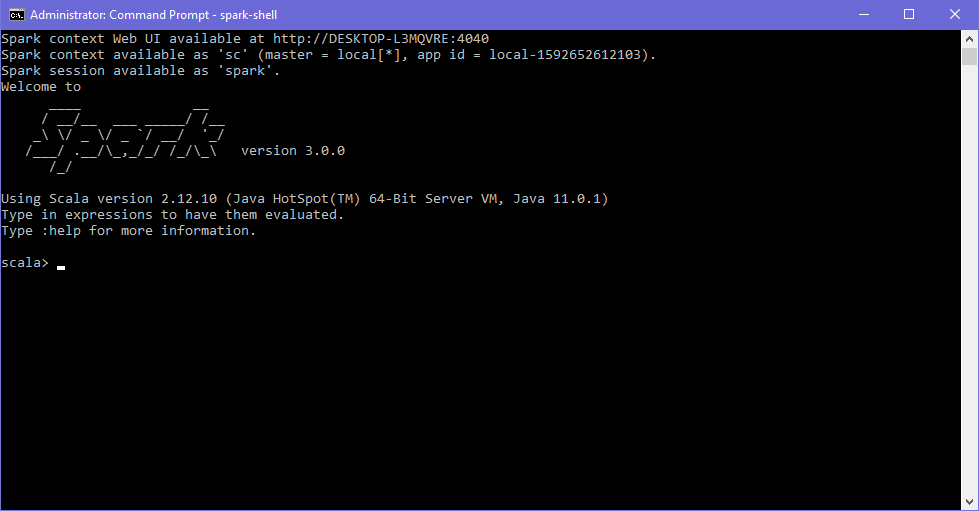
Tạo biến môi trường HADOOP\_HOME với đường dẫn tới thư mục winutils vừa tạo



Thêm đường dẫn sau vào Path



Mở CMD, chạy thử Spark bằng dòng lệnh *spark-shell*

**

# **Demo thực nghiệm**

A group of cartoon characters in a line

Description automatically generated

Nội Dung : Demo Của Chúng em là Áp dụng giải thuật tìm kiếm vào Game Sokoban

A screenshot of a video game

Description automatically generated

Lựa chọn giải thuật tìm kiếm để giải bài toán SOKOBAN,Tiến hành thực thi giải thuật bằng python

|  |  |  |
| --- | --- | --- |
| **Class state** | | |
| **Thuộc tính** | **Kiểu** | **Chức năng** |
| Board | String[,] | Ma trận lưu map của trạng thái hiện tại |
| State\_parent | State | Trạng thái cha của trạng thái hiện tại |
| Cost | Int | Giá trị phụ dùng để tính hàm lượng giá |
| Heuristic | Float | Giá trị lượng giá của trạng thái hiện tại |
| Check\_points | Int[(,)] | Danh sách vị trí các điểm đến |
| **Phương thức** | **Kiểu** | **Chức năng** |
| Get\_line(self) | String[,,] | Hàm quay lui từ trạng thái mục tiếu tìm được đến trạng thái đầu tiên để sinh ra đường đi |
| Compute\_heuristic(self) | float | Hàm tính toán khoảng cách từ tất cả box hiện tại đến tất cả checkpoint trên map |

# Cấu Trúc Dữ Liệu Chính

# Xử lí bài Toán

Giới thiệu Các thuật toán mà tụi em sẽ sử dụng

**Breadth First Search (BFS)**, là một giải thuật vét cạn, dùng để duyệt qua tất cả các trạng thái di chuyển có thể xảy ra khi áp dụng vào sokoban để tìm tới trạng thái mục tiêu

**A\*(**  **A-star Pathfinding**), là một giải thuật heuristic sử dụng một hàm lượng giá và một hàng đợi ưu tiên để tính toán các nước đi ngắn nhất dẫn đến trạng thái mục tiêu

# Giới thiệu Sơ lược thuật toán đối với Game

**Breadth First Search :**

Chúng ta sẽ từ trạng thái ban đầu, sinh ra tối đa 4 trạng thái với các nước đi của người chơi tiếp theo có thể đi. Từ đó tạo thành một cấu trúc cây. Cuối cùng dùng BFS để vét cạn cây chắc chắn sẽ tìm được trạng thái mục tiêu

**A\* :**

Cách làm tương tự với BFS, tuy nhiên các trạng thái sinh ra sẽ có giá trị lượng giá riêng và được sắp xếp vào một hàng đợi ưu tiên để từ đó giải thuật có thể lấy từ hàng đợi ra duyệt để đi đến trạng thái mục tiêu nhanh

# Định Nghĩa Bài Toán

**Có nghĩa xác định trạng thái ban đầu và trạng thái kết thúc**

**Đây Là giao diện Khi bắt đầu trò chơi**

**A screenshot of a video game

Description automatically generated**

**A blue hand with a knife

Description automatically generated**

**Và Đây Là Giao diện Khi clear xong màn**

A screenshot of a video game

Description automatically generated A green hand with black background

Description automatically generatedA green hand with a black background

Description automatically generated

**Với Thuật Toán BREADTH FIRST SEARCH**

**A screenshot of a video game

Description automatically generated**

**Giải thuật này sẽ sinh ra những bước đi hợp lệ từ một trạng thái cho trước. Sau đó sinh ra một cấu trúc cây. Đến khi trạng thái đạt trạng thái mục tiêu thì dừng lại và từ đó quay lui về trạng thái ban đầu để tìm ra đường đi**

**A screenshot of a video game

Description automatically generated**

**Với Thuật Toán A STAR ALGORITHM**

**Heuristic = 4 + 6 = 10**

**Giải thuật này cũng dựa vào việc sinh ra cấu trúc cây để duyệt bằng BFS, tuy nhiên, các thứ tự của các node con sẽ được sắp xếp dựa vào giá trị lượng giá (heuristic) có được trên từng bước sinh ra node**

**A screenshot of a game

Description automatically generated**

# Cải Tiến Thuật Toán

**A screenshot of a video game

Description automatically generated**

Trường hợp 1 : Box bị kẹt ở tường =>Không thể Kéo ra => Loại

Trường hợp 2: Không bị kẹt tường nhưng lại kẹt các box với nhau => Không thể Kéo ra ⬄Loại

Trường hợp 3 : có nghĩa là khi sinh ra 4 hướng đi nhưng sẽ có 1 số trường hợp như trường hợp 3 đó là người chơi không thể đi sang trái hoặc đi xuống => Loại

# Thống Kê và Kết Luận

A graph on a screen

Description automatically generated

Đây là thống kê giữa BFS và A\* trên 3 Phương diện

Phương diện thứ nhất Là trên Dung Lượng => cho ta thấy BFS Nhiếu Nhiều tài nguyên hơn A\*=> 90% Bỡi vì giải thuật BFS vét cạn cả case , có nghĩa là giải thuật này sẽ dò hết tất cả map để tìm đường vậy nên sẽ tốn dụng lượng hơn là điều tất yếu

A graph on a screen

Description automatically generated

**Phương diện Thứ 2 là số lượng trạng thái duyệt**

A graph of a diagram

Description automatically generated with medium confidence

**Phương diện thứ 3 là phương diện thời gian**

# **BẢNG PHÂN CÔNG CÔNG VIỆC**

|  |  |
| --- | --- |
| **NỘI DUNG** | **NGƯỜI THỰC HIỆN** |
| Giới thiệu và tìm hiểu thuật toán và Làm word +PP | Võ Ngọc Bảo  Nguyễn Đình Huy |
| Demo thực nghiệm | Đoàn Ngọc Sơn |