BỘ CÔNG THƯƠNG



TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP THỰC PHẨM TP.HCM

KHOA CÔNG NGHỆ THÔNG TIN

****

----🙞🙜🕮🙞🙜----

**TÍNH TOÁN SONG SONG**

**Đề tài: Tìm hiểu công nghệ Hadoop và xây dựng mô hình song song hóa Thuật toán Kruskal**

*GVHD : Trần Đình Toàn*

*Nhóm thực hiện*: Nhóm 11

Trần Thái Chân - 2001180118

Lê Minh Nhựt - 2001180152

Bùi Thị Như Hảo - 2001180282

Thành phố Hồ Chí Minh – 12/2019

**LỜI MỞ ĐẦU**

Có bao giờ bạn băn khoăn lượng dữ liệu của các công ty công nghệ lớn như Facebook, Amazon, Google, ... thu thập mỗi ngày và được xử lý như thế nào mà vẫn đảm bảo tốc độ truy cập một cách nhanh chóng và hiệu quả nhất.

Từ đấy chúng ta biết đến **BIG DATA**. Nó là thuật ngữ dùng để chỉ một tập hợp dữ liệu rất lớn và rất phức tạp đến nỗi những công cụ, ứng dụng xử lí dữ liệu truyền thống không thể nào đảm đương được. Tuy nhiên, Big Data lại chứa trong mình rất nhiều thông tin quý giá mà nếu trích xuất thành công, nó sẽ giúp rất nhiều cho việc kinh doanh, nghiên cứu khoa học, dự đoán các dịch bệnh sắp phát sinh và thậm chí là cả việc xác định điều kiện giao thông theo thời gian thực.

Hiện nay việc xử lý **BIG DATA** đang một trong những ưu tiên hàng đầu của các công ty công nghệ trên toàn thế giới. Nên những framwork giúp việc xử lý **BIG DATA** cũng đang ngày càng được chú ý và phát triển mạnh.

Trong đề tài này sẽ giới thiệu về Hadoop, một trong các công nghệ cốt lõi cho việc lưu trữ và truy cập số lượng lớn dữ liệu.

**NỘI DUNG:**

Chương 1. Giới thiệu Hadoop

[Chương 2. Cách cài đặt Hadoop](#_Toc524997448)

[Chương 3. Giới thiệu thuật toán Kruskal](#_Toc524997448)

[Chương 4. Cài đặt thuật toán Kruskal và Code chương trình](#_Toc524997448)

**[Chương 1. Giới thiệu Hadoop](#_Toc524997448)**

**[1.1 Hadoop là gì?](#_Toc524997448)**

[Hiện nay việc xử lý BIG DATA đang một trong những ưu tiên hàng đầu của các công ty công nghệ trên toàn thế giới. Những framwork giúp việc xử lý BIG DATA cũng đang ngày càng được chú ý và phát triển mạnh.](#_Toc524997448)

[Có nhiều định nghĩa về Hadoop như sau:](#_Toc524997448)

**[- Apache Hadoop định nghĩa:](#_Toc524997448)**[“Apache Hadoop là một framework dùng để chạy những ứng dụng trên một cluster lớn được xây dựng trên những phần cứng thông thường. Hadoop hiện thực mô hình Map/Reduce, đây là mô hình mà ứng dụng sẽ được chia nhỏ ra thành nhiều phân đoạn khác nhau, và các phần này sẽ được chạy song song trên nhiều node khác nhau. Thêm vào đó, Hadoop cung cấp 1 hệ thống file phân tán (HDFS) cho phép lưu trữ dữ liệu lên trên nhiều node. Cả Map/Reduce và HDFS đều được thiết kế sao cho framework sẽ tự động quản lý được các lỗi, các hư hỏng về phần cứng của các node.”](#_Toc524997448)**[- Wikipedia định nghĩa:](#_Toc524997448)**[“Hadoop là một framework nguồn mở viết bằng Java cho phép phát triển các ứng dụng phân tán có cường độ dữ liệu lớn một cách miễn phí. Nó cho phép các ứng dụng có thể làm việc với hàng ngàn node khác nhau và hàng petabyte dữ liệu. Hadoop lấy được phát triển dựa trên ý tưởng từ các công bố của Google về mô hình MapReduce và hệ thống file phân tán Google File System (GFS).”](#_Toc524997448)**[Vậy ta có thể kết luận như sau:](#_Toc524997448)**[1) Hadoop là một framework cho phép phát triển các ứng dụng phân tán.  
2) Hadoop viết bằng bằng Java. Tuy nhiên, nhờ cơ chế streaming, Hadoop cho phép phát triển các ứng dụng phân tán bằng cả java lẫn một số ngôn ngữ lập trình khác như C++, Python, Pearl.  
3) Hadoop cung cấp một phương tiện lưu trữ dữ liệu phân tán trên nhiều node, hỗ trợ tối ưu hoá lưu lượng mạng, đó là HDFS. HDSF che giấu tất cả các thành phần phân tán, các nhà phát triển ứng dụng phân tán sẽ chỉ nhìn thấy HDFS như một hệ thống file cục bộ bình thường.](#_Toc524997448)

[4) Hadoop giúp các nhà phát triển ứng dụng phân tán tập trung tối đa vào phần logic của ứng dụng, bỏ qua được một số phần chi tiết kỹ thuật phân tán bên dưới (phần này do Hadoop tự động quản lý).  
5) Hadoop là Linux-based. Tức Hadoop chỉ chạy trên môi trường Linux2.](#_Toc524997448)

**[1.2 Hadoop giải quyết các vấn đề](#_Toc524997448)**

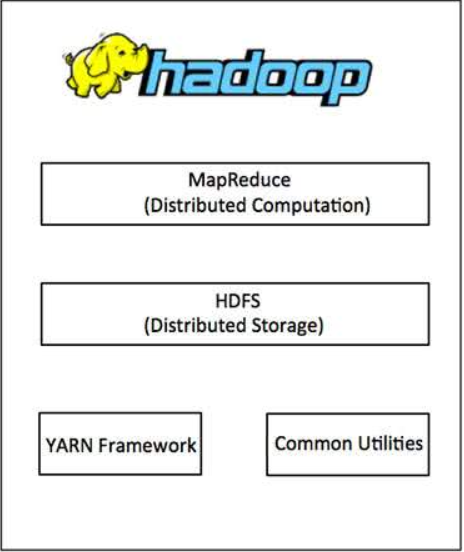
[- Xử lý và làm việc với khối lượng dữ liệu khổng lồ tính bằng Petabyte.](#_Toc524997448)

[- Xử lý trong môi trường phân tán, dữ liệu lưu trữ ở nhiều phần cứng khác nhau, yêu cầu xử lý đồng bộ.](#_Toc524997448)

[- Các lỗi xuất hiện thường xuyên.](#_Toc524997448)

[- Băng thông giữa các phần cứng vật lý chứa dữ liệu phân tán có giới hạn.](#_Toc524997448)

**[1.3 Kiến trúc hadoop](#_Toc524997448)** [- Một cụm Hadoop nhỏ gồm 1 master node và nhiều worker/slave node. Toàn bộ cụm chứa 2 lớp, một lớp MapReduce Layer và lớp kia là HDFS Layer. Mỗi lớp có các thành phần liên quan riêng. Master node gồm JobTracker, TaskTracker, NameNode, và DataNode. Slave/worker node gồm DataNode, và TaskTracker. Cũng có thể slave/worker node chỉ là dữ liệu hoặc node để tính toán.](#_Toc524997448)

[](#_Toc524997448)

[Hadoop gồm 4 module:](#_Toc524997448)

[- Hadoop Common: Đây là các thư viện và tiện ích cần thiết của Java để các module khác sử dụng. Những thư viện này cung cấp hệ thống file và lớp OS trừu tượng, đồng thời chứa các mã lệnh Java để khởi động Hadoop.](#_Toc524997448)

[- Hadoop YARN: Đây là framework để quản lý tiến trình và tài nguyên của các cluster.](#_Toc524997448)

[- Hadoop Distributed File System (HDFS): Đây là hệ thống file phân tán cung cấp truy cập thông lượng cao cho ứng dụng khai thác dữ liệu.](#_Toc524997448)

[- Hadoop MapReduce: Đây là hệ thống dựa trên YARN dùng để xử lý song song các tập dữ liệu lớn.](#_Toc524997448)

**[1.4 Hoạt động Hadoop](#_Toc524997448)**

[Giai đoạn 1:](#_Toc524997448)

[Một user hay một ứng dụng có thể submit một job lên Hadoop (hadoop job client) với yêu cầu xử lý cùng các thông tin cơ bản:](#_Toc524997448)

[- Truyền dữ liệu lên server(input) để bắt đầu phân tán dữ liệu và đưa ra kết quả (output).](#_Toc524997448)

[- Các dữ liệu được chạy thông qua 2 hàm chính là map và reduce.](#_Toc524997448)

[Map: sẽ quét qua toàn bộ dữ liệu và phân tán chúng ra thành các dữ liệu con.](#_Toc524997448)

[Reduce: sẽ thu thập các dữ liệu con lại và sắp xếp lại chúng.](#_Toc524997448)

[- Các thiết lập cụ thể liên quan đến job thông qua các thông số truyền vào.](#_Toc524997448)

[Giai đoạn 2:](#_Toc524997448)

[- Hadoop job client submit job (file jar, file thực thi) và bắt đầu lập lịch làm việc(JobTracker) đưa job vào hàng đợi .](#_Toc524997448)

[- Sau khi tiếp nhận yêu cầu từ JobTracker, server cha(master) sẽ phân chia công việc cho các server con(slave). Các server con sẽ thực hiện các job được giao và trả kết quả cho server cha.](#_Toc524997448)

[Giai đoạn 3:](#_Toc524997448)

[- TaskTrackers dùng để kiểm tra đảm bảo các MapReduce hoạt động bình thường và kiểm tra kết quả nhận được (quá trình output).](#_Toc524997448)

[- Khi “chạy Hadoop” có nghĩa là chạy một tập các trình nền - daemon, hoặc các chương trình thường trú, trên các máy chủ khác nhau trên mạng của bạn. Những trình nền có vai trò cụ thể, một số chỉ tồn tại trên một máy chủ, một số có thể tồn tại trên nhiều máy chủ.](#_Toc524997448)

**[1.5 Ưu điểm của Hadoop và Các điểm thuận lợi khi dùng hadoop](#_Toc524997448)**

[Ưu điểm của Hadoop:](#_Toc524997448)

[- Hadoop framework cho phép người dùng nhanh chóng viết và kiểm tra các hệ thống phân tán. Đây là cách hiệu quả cho phép phân phối dữ liệu và công việc xuyên suốt các máy trạm nhờ vào cơ chế xử lý song song của các lõi CPU.](#_Toc524997448)

[- Hadoop không dựa vào cơ chế chịu lỗi của phần cứng fault-tolerance and high availability (FTHA), thay vì vậy bản thân Hadoop có các thư viện được thiết kế để phát hiện và xử lý các lỗi ở lớp ứng dụng.](#_Toc524997448)

[- Hadoop có thể phát triển lên nhiều server với cấu trúc master-slave để đảm bảo thực hiện các công việc linh hoạt và không bị ngắt quãng do chia nhỏ công việc cho các server slave được điều khiển bởi server master.](#_Toc524997448)

[- Hadoop có thể tương thích trên mọi nền tảng như Window, Linux, MacOs do được tạo ra từ Java.](#_Toc524997448)

[Các điểm thuận lợi khi dùng hadoop:](#_Toc524997448)

[- Robus and Scalable – Có thể thêm node mới và thay đổi chúng khi cần.](#_Toc524997448)

[- Affordable and Cost Effective – Không cần phần cứng đặc biệt để chạy Hadoop.](#_Toc524997448)

[- Adaptive and Flexible – Hadoop được xây dựng với tiêu chí xử lý dữ liệu có cấu trúc và không cấu trúc.](#_Toc524997448)

[- Highly Available and Fault Tolerant – Khi 1 node lỗi, nền tảng Hadoop tự động chuyển sang node khác.](#_Toc524997448)

**[Chương 2. Cách cài đặt Hadoop](#_Toc524997448)**

[Việc cài đặt Hadoop trên windows về cơ bản cũng tương tự như trên Linux. Tuy nhiên, cài Hadoop trên Linux đơn giản hơn vì ngay từ đầu Hadoop được viết cho nền tảng Linux, sau này mới phát triển thêm cho nền tảng Win. Có một số lỗi hay gặp do sự khác biệt về hệ điều hành. Do đó, bên cạnh việc hướng dẫn chi tiết về cách cài đặt Hadoop trên Windows 10, Nhóm sẽ đưa ra các lỗi và cách khắc phục mà Nhóm thường gặp trong chính quá trình cài đặt.](#_Toc524997448)

## [2.1. Yêu cầu về hệ thống trước khi cài đặt](#_Toc524997448)

* [Windows: 8, 10](#_Toc524997448)
* [Java:](#_Toc524997448)**[version 8](#_Toc524997448)**[.](#_Toc524997448)

****[một số lưu ý trước khi cài:](#_Toc524997448)****

[1, Đường dẫn (path) của java phải không có dấu cách. Nếu có, sẽ bị lỗi không thể format namenode. Thông thường khi cài đặt, java sẽ mặc định được cài vào C:\Program Files (có dấu cách).](#_Toc524997448)

[2, Cần phải thiết lập JAVA\_HOME và thêm java vào Path của hệ thống.](#_Toc524997448)

[https://www.oracle.com/java/technologies/javase/javase-jdk8-downloads.html](#_Toc524997448)

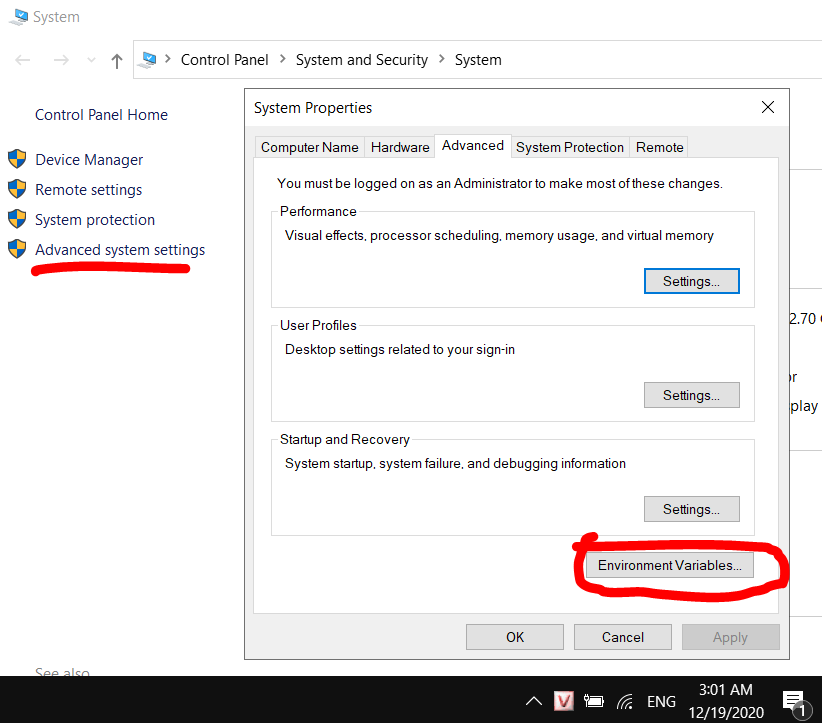
## [2.2. Cài đặt Hadoop](#_Toc524997448)

### [2.2.1 Tải Hadoop](#_Toc524997448)

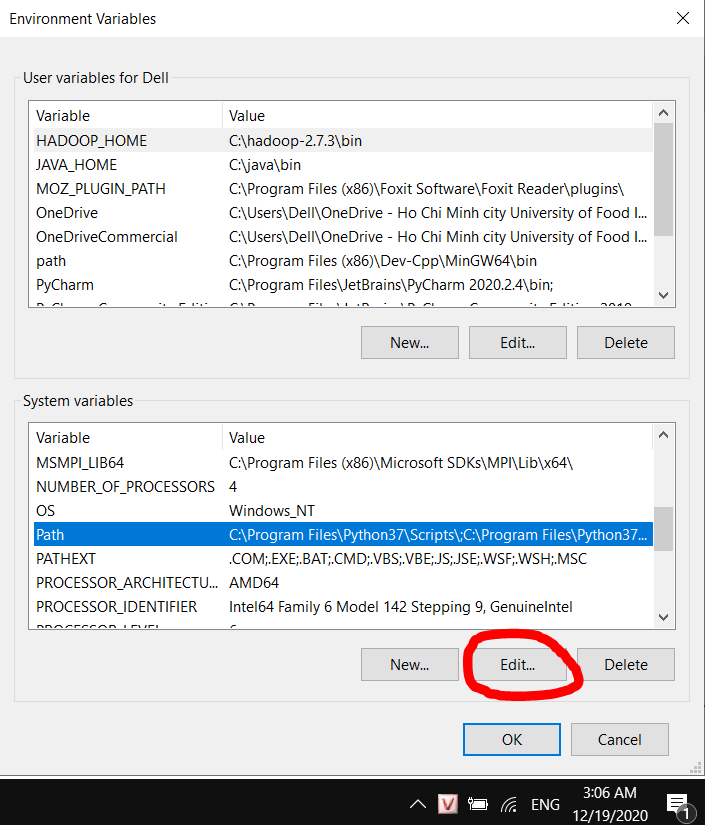
[Hadoop hiện có sẵn gói binary cho windows.  Trong hướng dẫn này nhóm chọn Hadoop-2.7.3 . Tải về từ link sau:](#_Toc524997448) *[https://www.apache.org/dyn/closer.cgi/hadoop/common/hadoop-2.7.3/hadoop-2.7.3.tar.gz](https://www.apache.org/dyn/closer.cgi/hadoop/common/hadoop-3.1.3/hadoop-3.1.3.tar.gz)*

[Sau khi tải về, giải nén và copy vào ổ đĩa muốn cài. Lưu ý, tên đường dẫn phải ko có dấu cách. Ví dụ, để Hadoop trong thư mục: C:\hadoop-2.7.3 Sau đó, thiết lập HADOOP\_HOME và thêm vào System Path (tương tự với Java). Xem hướng dẫn trong các hình sau:](#_Toc524997448)

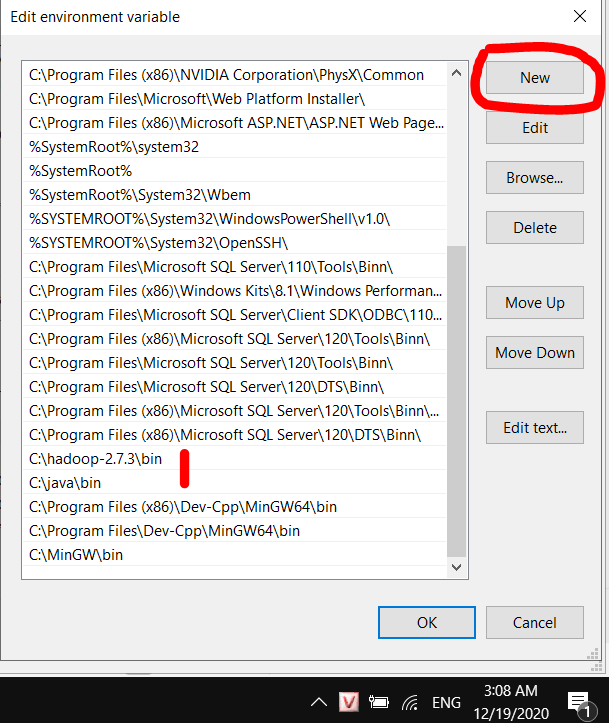
[Click vào Environment Variables để thay đổi](#_Toc524997448)

[](#_Toc524997448)

[Chọn ‘Path’, rồi click Edit để sửa](#_Toc524997448)

[](#_Toc524997448)

[Chọn New để thêm đường dẫn Java và Hadoop](#_Toc524997448)

[](#_Toc524997448)

### [2.2.2. Cấu hình Hadoop](#_Toc524997448)

[Các file cho phép cấu hình Hadoop nằm trong thư mục: ~\hadoop-2.7.3\etc\hadoop.](#_Toc524997448)

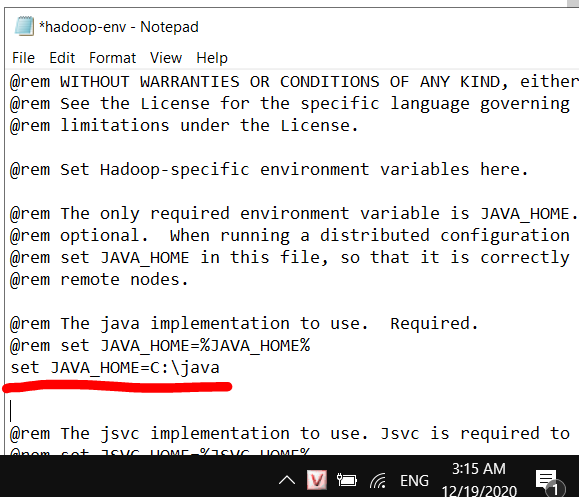
[Các file cần điều chỉnh để cấu hình cho hệ thống gồm:](#_Toc524997448)*[core-site.xml, hdfs-site.xml, yarn-site.xml, và mapred-site.xml](#_Toc524997448)*

[Ngoài ra, cũng cần thay đổi một số cấu hình trong file cấu hình môi trường của Hadoop trên windows là](#_Toc524997448)*[hadoop-env.cmd](#_Toc524997448)*

****[Bước 1.](#_Toc524997448)****[Với file môi trường](#_Toc524997448)*[hadoop-env.cmd](#_Toc524997448)*[:](#_Toc524997448)

[Cần thêm Path của JAVA\_HOME vào](#_Toc524997448)*[hadoop-env.cmd](#_Toc524997448)*

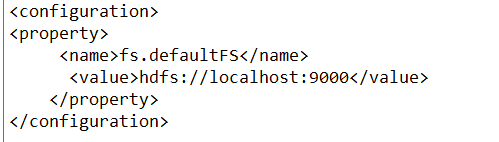
[set JAVA\_HOME=C:\Java](#_Toc524997448)

[](#_Toc524997448)

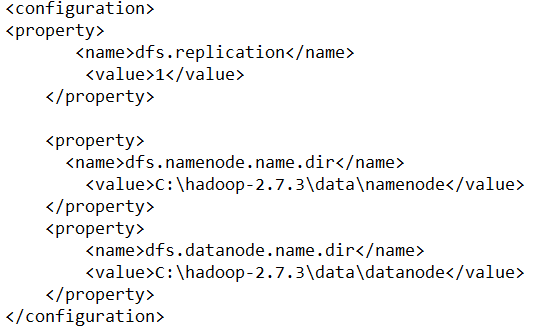
****[Bước 2](#_Toc524997448)****[. Các file cấu hình cần được điều chỉnh như sau:](#_Toc524997448)

[Lưu ý: Trong thư mục HADOOP\_HOME, tạo thêm folder ‘tmp’ và ‘data’. Trong thư mục ‘data’, tạo 2 thư mục con ‘namenode’ và ‘datanode’](#_Toc524997448)

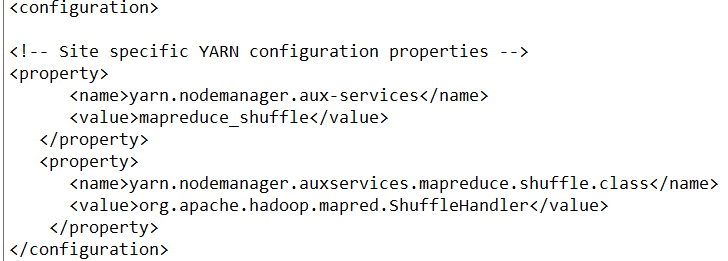
****[a. core-site.xml:](#_Toc524997448)****

[](#_Toc524997448)

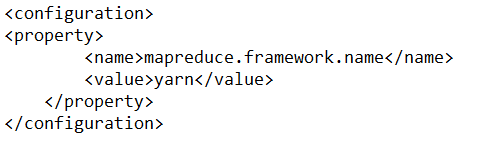
****[b. hdfs-site.xml](#_Toc524997448)****

[](#_Toc524997448)

**[c. yarn-site.xm](#_Toc524997448)**[l](#_Toc524997448)

[](#_Toc524997448)

**[d. mapred-site.xml](#_Toc524997448)**

[](#_Toc524997448)

****[Bước 3.](#_Toc524997448)****[Cấu hình lại cho Hadoop tương thích với Windows:](#_Toc524997448)

[Lưu ý : trong file binary của Hadoop, một số cấu hình chưa thích hợp, nên chạy sẽ bị lỗi Do đó, cần tải file trong link sau:](#_Toc524997448) <https://github.com/s911415/apache-hadoop-3.1.0-winutils>

[Sau khi tải về, chỉ việc copy thư mục ‘bin’ trong này sang Hadoop (thay cho thư mục ‘bin’ cũ của hadoop.)](#_Toc524997448)

### [2.2.3. Khởi động Hadoop](#_Toc524997448)

[Việc đầu tiên sau bước cấu hình là format Namenode của HDFS. Lưu ý, việc này chỉ thực hiện 1 lần sau khi cài đặt. Câu lệnh (trên cmd)](#_Toc524997448)

[](#_Toc524997448)

[Bây giờ, khởi chạy các daemons của YARN và HDFS Chạy tất cả các daemons: Trên cmd hiện thời, có thể dùng %HADOOP\_HOME% thay vì phải đánh/copy đường dẫn home của Hadoop. Để chạy tất cả daemons một lần, chạy file start-all.cmd trong thư mục ‘sbin’:](#_Toc524997448)

[](#_Toc524997448)

[Để khởi động YARN và HDFS riêng rẽ: Cũng trong thư mục ‘sbin’, chạy các file ‘start-dfs.cmd’ và ‘start-yarn.cmd’](#_Toc524997448)

[Nếu không gặp lỗi gì, đã hoàn thành việc cài đặt Hadoop trên Windows. Có thể truy cập Web UI của HDFS và YARN để xem thông tin hoạt động của Hadoop. - HDFS master Web UI:](#_Toc524997448)<http://localhost:9870/> [- YARN master Web UI:](#_Toc524997448)[http://localhost:8088/](http://localhost:9870/) [Và bây giờ có thể chạy job trên Hadoop. có thể xem trong các bài viết liên quan về chạy MapReduce jobs hay bài viết về các câu lệnh của HDFS.](#_Toc524997448)

[Để dừng chạy Hadoop daemons:](#_Toc524997448)

[](#_Toc524997448)

## [2.3. Một số lỗi thường gặp khi cài đặt và chạy Hadoop jobs trên Windows.](#_Toc524997448)

## ****[2.3.1. Lỗi ko format được Namenode: (hdfs namenode -format)](#_Toc524997448)**** [Nguyên nhân: Do ko cài java 8 hoặc Java path có dấu cách. Như mình đã lưu ý ở trên, đường dẫn home của Java và Hadoop không được có dấu cách. Khi bạn cần sửa đường dẫn, cần nhớ update trong System Environment Variables.](#_Toc524997448) ****[2.3.2. Khởi chạy được HDFS, ko mở được web UI](#_Toc524997448)**** [Nguyên nhân: Trong /ect/hosts (linux) hoặc C:\Windows\System32\drivers\etc\hosts, có thể chưa đặt localhost.](#_Toc524997448)

[Giải pháp: Trong file trên, thêm dòng:](#_Toc524997448)

[](#_Toc524997448)

****[2.3.3. Lỗi không khởi chạy được YARN master (ResourceManager):](#_Toc524997448)****

[Error:](#_Toc524997448)[NoClassDefFoundError org/apache/hadoop/yarn/server/timelineservice/collector/TimelineCollectorManage](https://stackoverflow.com/questions/51118358/noclassdeffounderror-org-apache-hadoop-yarn-server-timelineservice-collector-tim)**[r](#_Toc524997448)** [Giải pháp: Copy \*.timeservice\*.jar trong ..\share\hadoop\yarn\timelineservice vào thư mục: ..\share\hadoop\yarn](#_Toc524997448)

**[2.3.4.](#_Toc524997448)**[Lỗi khi chạy Mapreduce jobs: failed](#_Toc524997448)** [Error: CreateSymbolicLink error (1314): required privilege is not held by the client.”](#_Toc524997448)**[(Lỗi xảy ra do username không có quyền quản trị).](#_Toc524997448) **[Giải pháp:](#_Toc524997448)** [Bước 1. Chọn phím Windows (biểu tượng cửa sổ) Bước 2: Nhập tìm ‘Run’, và enter Bước 3: Trong khung lệnh của Run, nhập “secpol.msc” rồi enter. Khi đó, thư mục Local Security Policy (Chính sách Bảo mật  cục bộ) sẽ hiện lên. Bước 4: Di chuyển đến mục  Local Policy (Chính sách cục bộ) , chọn User Right Assignment (thiết lập quyền người dùng) Bước 5: Di chuyển xuống dòng ‘Create symbolic link’ và chọn Add user or group (thêm người dùng hoặc nhóm). Thêm tên người dùng của bạn. Bước 6. Tắt hết chương trình và khởi động máy. Bây giờ bạn đã có quyền để tạo Create Symbolic Link.](#_Toc524997448)

****[Lưu ý:](#_Toc524997448)****[Trong trường hợp vẫn ko được, có thể cần phải Disable (tắt) tính năng dưới đây (và khởi động lại máy): Local Policy > Security Options > User Account Control: Run all administrators in Admin Approval](#_Toc524997448)

**[Chương 3. Giới thiệu về thuật toán Kruskal](#_Toc524997448)**

**[3.1](#_Toc524997448)** **[Thuật toán Kruskal là gì?](#_Toc524997448)**  [Thuật toán Kruskal là một](#_Toc524997448)[thuật toán](https://vi.wikipedia.org/wiki/Thu%E1%BA%ADt_to%C3%A1n" \o "Thuật toán)[trong](#_Toc524997448)[lý thuyết đồ thị](https://vi.wikipedia.org/wiki/L%C3%BD_thuy%E1%BA%BFt_%C4%91%E1%BB%93_th%E1%BB%8B" \o "Lý thuyết đồ thị)[để tìm](#_Toc524997448)[cây bao trùm nhỏ nhất](https://vi.wikipedia.org/wiki/C%C3%A2y_bao_tr%C3%B9m_nh%E1%BB%8F_nh%E1%BA%A5t" \o "Cây bao trùm nhỏ nhất)[của một](#_Toc524997448)[đồ thị](https://vi.wikipedia.org/wiki/%C4%90%E1%BB%93_th%E1%BB%8B_(l%C3%BD_thuy%E1%BA%BFt_%C4%91%E1%BB%93_th%E1%BB%8B)" \o "Đồ thị (lý thuyết đồ thị))[liên thông](https://vi.wikipedia.org/w/index.php?title=Li%C3%AAn_th%C3%B4ng_(l%C3%BD_thuy%E1%BA%BFt_%C4%91%E1%BB%93_th%E1%BB%8B)&action=edit&redlink=1" \o "Liên thông (lý thuyết đồ thị) (trang chưa được viết))[có trọng số. Nói cách khác, nó tìm một tập hợp các](#_Toc524997448)[cạnh](https://vi.wikipedia.org/wiki/%C4%90%E1%BB%93_th%E1%BB%8B_(l%C3%BD_thuy%E1%BA%BFt_%C4%91%E1%BB%93_th%E1%BB%8B)" \o "Đồ thị (lý thuyết đồ thị))[tạo thành một cây chứa tất cả các](#_Toc524997448)[đỉnh](https://vi.wikipedia.org/wiki/%C4%90%E1%BB%93_th%E1%BB%8B_(l%C3%BD_thuy%E1%BA%BFt_%C4%91%E1%BB%93_th%E1%BB%8B)" \o "Đồ thị (lý thuyết đồ thị))[của đồ thị và có tổng trọng số các cạnh là nhỏ nhất.Thuật toán Kruskal là một ví dụ của](#_Toc524997448)[thuật toán tham lam](https://vi.wikipedia.org/wiki/Gi%E1%BA%A3i_thu%E1%BA%ADt_tham_lam" \o "Giải thuật tham lam)[.](#_Toc524997448)

[Thuật toán này xuất bản lần đầu tiên năm 1956, bởi](#_Toc524997448)[Joseph Kruskal](https://vi.wikipedia.org/w/index.php?title=Joseph_Kruskal&action=edit&redlink=1" \o "Joseph Kruskal (trang chưa được viết))[.](#_Toc524997448)

[Thuật toán sẽ duy trì một đồ thị con phi chu trình TT, gọi là một rừng (forest), của G (V, E). Ban đầu, khởi tạo T=V, TT là tập chỉ gồm các đỉnh mà không có cạnh nào cả. Tại mỗi bước, thuật toán sẽ cố gắng thêm cạnh vào TT, để cuối cùng TT sẽ là một cây khung. Do ta muốn tổng trọng số của TT nhỏ nhất có thể, ta sẽ chọn cạnh cạnh có trọng số nhỏ nhất, gọi là ee, trong số các cạnh không ở trong TT và thêm vào TT. Sẽ có hai trường hợp:](#_Toc524997448)

- Nếu T∪{e}T∪{e} không có chu trình, ta sẽ thêm ee vào TT. Thao tác thêm ee vào TT sẽ tương đương với gộp hay cây trong TT lại với nhau thành một cây mới.

- Nếu T∪{e}T∪{e} có chu trình, ta sẽ thử cạnh tiếp theo.

Tính chất (6) cho chúng ta biết thêm cạnh theo thứ tự tăng dần của trọng số như vậy có lẽ sẽ cho chúng ta cây khung nhỏ nhất.

**Thuật toán Kruskal** 32 Generic-MST(G, c) A = { } //Bất biến: A là tập con các cạnh của CKNN nào đó while A chưa là cây khung do tìm cạnh (u, v) là an toàn đối với A = A ∪{(u, v)} // A vẫn là tập con các cạnh của CKNN nào đó return A - A là rừng. - Cạnh an toàn được bổ sung vào A có trọng số nhỏ nhất trong số các cạnh nối các cặp thành phần liên thông của nó. Trước khi áp dụng thuật toán Kruskal để tìm cây khung tối thiểu được cải tiến ở trên, ta có nhận xét: - Việc tìm cây khung tối thiểu của thuật toán là đi tìm các cạnh thoả mãn hai điều kiện là trọng số của cạnh được chọn phải là nhỏ nhất và khi thêm vào cây khung không tạo thành chu trình trong cây khung. Mỗi khi tìm được một cạnh thì cũng tìm được đỉnh của cạnh ấy để thêm vào tập đỉnh của cây khung, ký hiệu tập đỉnh của cây khung là V\*.

Để hiểu giải thuật Kruskal, theo dõi ví dụ sau:



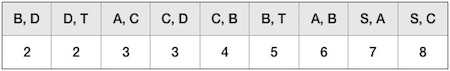
**Bước 1: Xóa tất cả các vòng và các cạnh song song**

Xóa tất cả các vòng và các cạnh song song từ độ thị ban đầu.



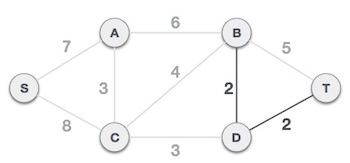
**Bước 2: Sắp xếp tất cả các cạnh theo trọng số tăng dần**

Bước tiếp theo là tạo một tập các cạnh và trọng số và sắp xếp chúng theo thứ tự tăng dần về trọng số. (Giá trị trọng số là số hiển thị bên cạnh các cạnh trong hình minh họa trên.)

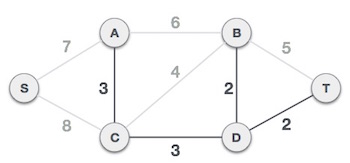


**Bước 3: Thêm một cạnh có trọng số thấp nhất**

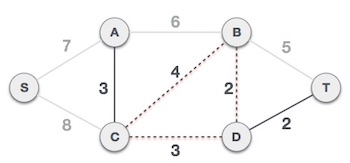
Bây giờ chúng ta bắt đầu thêm các cạnh vào đồ thị bắt đầu từ cạnh có trọng số thấp nhất. Tại bất cứ thời điểm nào, chúng ta cũng cần kiểm tra các thuộc tính của cây khung có còn được duy trì hay không. Trong trường hợp khi thêm một cạnh mà làm phá vỡ thuộc tính của cây khung, thì chúng ta cần cân nhắc việc không thêm cạnh đó vào đồ thị.



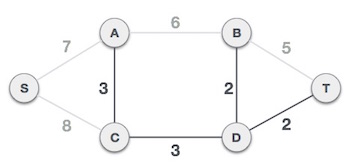
Trọng số nhỏ nhất trong hình là 2 và đó là các cạnh là BD và DT, do đó chúng ta thêm hai cạnh này vào. Việc thêm hai cạnh này không vi phạm các đặc điểm của cây khung do đó chúng ta có thể tiếp tục lựa chọn cạnh tiếp theo.

Trọng số tiếp theo là 3 và đó là các cạnh AC.CD. Chúng ta thêm hai cạnh này. 

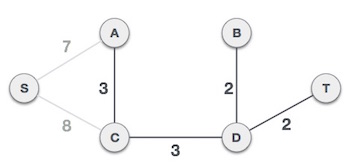
Trọng số tiếp theo trong hình là 4 và chúng ta thấy rằng khi thêm nó vào đồ thị sẽ tạo nên một chu trình trong đồ thị. Như hình minh họa:



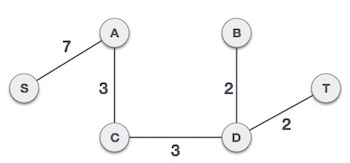
Do đó chúng ta bỏ qua trọng số này. Tiến trình của chúng ta sẽ bỏ qua/tránh việc thêm các cạnh mà khi thêm cạnh đó vào sẽ tạo nên một chu trình trong đồ thị.



Tiếp theo với hai trọng số 5 và 6, chúng ta cũng thấy rằng chúng cũng tạo nên một chu trình. Do đó chúng ta bỏ qua hai trọng số này và chuyển tới trọng số tiếp theo.



Bây giờ đồ thị của chúng ta chỉ còn một nút để thêm vào. Giữa hai cạnh có trọng số lần lượt là 7 và 8, chúng ta thêm cạnh có trọng số nhỏ hơn.



Bằng việc thêm cạnh SA, chúng ta đã có một cây khung nhỏ nhất theo giải thuật của Kruskal.

**3.2 Tư tưởng thuật toán**

Cho một [đồ thị có trọng số](https://vi.wikipedia.org/w/index.php?title=%C4%90%E1%BB%93_th%E1%BB%8B_c%C3%B3_tr%E1%BB%8Dng_s%E1%BB%91&action=edit&redlink=1" \o "Đồ thị có trọng số (trang chưa được viết)) với n [đỉnh](https://vi.wikipedia.org/wiki/%C4%90%E1%BB%89nh" \o "Đỉnh). Yêu cầu tìm ra [cây khung nhỏ nhất](https://vi.wikipedia.org/wiki/C%C3%A2y_khung_nh%E1%BB%8F_nh%E1%BA%A5t" \o "Cây khung nhỏ nhất).

Thuật toán Kruskal dựa trên mô hình xây dựng [cây khung nhỏ nhất](https://vi.wikipedia.org/wiki/C%C3%A2y_khung_nh%E1%BB%8F_nh%E1%BA%A5t" \o "Cây khung nhỏ nhất) bằng thuật toán hợp nhất.

- Thuật toán không xét các cạnh với thứ tự tuỳ ý.

- Thuật toán xét các cạnh theo thứ tự đã sắp xếp theo trọng số.

Để xây dựng tập n-1 [cạnh](https://vi.wikipedia.org/w/index.php?title=C%E1%BA%A1nh&action=edit&redlink=1" \o "Cạnh (trang chưa được viết)) của [cây khung](https://vi.wikipedia.org/wiki/C%C3%A2y_bao_tr%C3%B9m" \o "Cây bao trùm) nhỏ nhất - tạm gọi là tập K, Kruskal đề nghị cách kết nạp lần lượt các cạnh vào tập đó theo nguyên tắc như sau:

- Ưu tiên các cạnh có [trọng số](https://vi.wikipedia.org/w/index.php?title=Tr%E1%BB%8Dng_s%E1%BB%91&action=edit&redlink=1" \o "Trọng số (trang chưa được viết)) nhỏ hơn.

- Kết nạp cạnh khi nó không tạo [chu trình](https://vi.wikipedia.org/wiki/Chu_tr%C3%ACnh_(l%C3%BD_thuy%E1%BA%BFt_%C4%91%E1%BB%93_th%E1%BB%8B)" \o "Chu trình (lý thuyết đồ thị)) với tập cạnh đã kết nạp trước đó.

Đó là một nguyên tắc chính xác và đúng đắn, đảm bảo tập K nếu thu đủ n - 1 cạnh sẽ là [cây khung nhỏ nhất](https://vi.wikipedia.org/wiki/C%C3%A2y_khung_nh%E1%BB%8F_nh%E1%BA%A5t" \o "Cây khung nhỏ nhất).

3.3 Mô tả thuật toán

Giả sử ta cần tìm cây bao trùm nhỏ nhất của đồ thị *G*. Thuật toán bao gồm các bước sau.

* Khởi tạo rừng *F* (tập hợp các [cây](https://vi.wikipedia.org/wiki/C%C3%A2y_(l%C3%BD_thuy%E1%BA%BFt_%C4%91%E1%BB%93_th%E1%BB%8B)" \o "Cây (lý thuyết đồ thị))), trong đó mỗi đỉnh của *G* tạo thành một cây riêng biệt
* Khởi tạo tập *S* chứa tất cả các cạnh của *G*
* Chừng nào *S* còn [khác rỗng](https://vi.wikipedia.org/w/index.php?title=Kh%C3%A1c_r%E1%BB%97ng&action=edit&redlink=1" \o "Khác rỗng (trang chưa được viết)) và *F* gồm hơn một cây
  + Xóa cạnh nhỏ nhất trong *S*
  + Nếu cạnh đó nối hai cây khác nhau trong *F*, thì thêm nó vào *F* và hợp hai cây kề với nó làm một
  + Nếu không thì loại bỏ cạnh đó.

Khi thuật toán kết thúc, rừng chỉ gồm đúng một cây và đó là một cây bao trùm nhỏ nhất của đồ thị *G*.

Cho đồ thị **G=(X, E)**.

Bước 1: Sắp xếp các cạnh của đồ thị theo thứ tự trọng số tăng dần.

Bước 2: Khởi tạo T:= Ø

Bước 3: Lần lượt lấy từng cạnh thuộc danh sách đã sắp xếp. Nếu T+{e} không chứa chu trình thì gán T:=T+{e}.

Bước 4: Nếu T đủ n-1 phần tử thì dừng, ngược lại làm tiếp bước 3.

**Ghi chú**

* Trong quá trình xây dựng T thì các cạnh có thể không [liên thông](https://vi.wikipedia.org/wiki/T%E1%BA%ADp_h%E1%BB%A3p_li%C3%AAn_th%C3%B4ng" \o "Tập hợp liên thông) nhau lúc đó T chỉ là [rừng](https://vi.wikipedia.org/wiki/R%E1%BB%ABng" \o "Rừng) chứ chưa trở thành [cây](https://vi.wikipedia.org/wiki/C%C3%A2y" \o "Cây).
* Khi [thuật toán](https://vi.wikipedia.org/wiki/Thu%E1%BA%ADt_to%C3%A1n" \o "Thuật toán) dừng:

Nếu T chưa đủ n - 1 cạnh thì [đồ thị](https://vi.wikipedia.org/wiki/%C4%90%E1%BB%93_th%E1%BB%8B" \o "Đồ thị) G không liên thông(không có cây khung)

Ngược lại thì T là [cây khung](https://vi.wikipedia.org/wiki/C%C3%A2y_bao_tr%C3%B9m" \o "Cây bao trùm) cần tìm.

Nếu *E* là số cạnh và *V* là số đỉnh của đồ thị thì thuật toán Kruskal chạy trong thời gian *[O](https://vi.wikipedia.org/wiki/K%C3%AD_hi%E1%BB%87u_O_l%E1%BB%9Bn" \o "Kí hiệu O lớn)*(*E* [log](https://vi.wikipedia.org/wiki/L%C3%B4garit" \o "Lôgarit) *V*).

**Thuật toán Kruskal:**

**Begin**

Bước 1 (Sắp xếp)

Bước 2 (Khởi tạo): T = ∅; //Khởi tạo tập cạnh cây khung là ∅ d(H) = 0’ //Khởi tạo độ dài nhỏ nhất cây khung là 0

Bước 3 (Lặp): while (|T; E = E \ {e}; //Loại cạnh e ra khỏi đồ thị if (T∪{e} không tạo nên chu trình ) then { T = ∪{e}; // Kết nạp e vào tập cạnh cây khung d(H) = d(H) + d(e); //Độ dài của tập cạnh cây khung endif; endwhile;

Bước 4 (Trả lại kết quả): if (|T| < n-1) then ; else Return( T, d(H));

**end.**

**Chương 4. Cài đặt thuật toán Kruskal và Code chương trình**

**4.1 Cài đặt thuật toán Kruskal**

Thuật toán Kruskal tốt đối với những đồ thị thưa thớt, nhiều đỉnh, nhưng ít cạnh (vẫn liên thông).

Nhận thấy rằng, khi cài đặt chương trình tìm cây khung nhỏ nhất bằng thuật toán Kruskal, việc kết nạp các cạnh vào cây khung không cần điều kiện cho cạnh được kết nạp là phải liên thông với cây khung.

Việc kết nạp một cạnh nào đó mà có trọng số nhỏ nhất trong các cạnh chưa được kết nạp, dễ nhận thấy là chỉ cần có trọng số nhỏ nhất và không tạo thành chu trình khi thêm vào tập cạnh của cây khung.

Chương trình tìm cây khung nhỏ nhất theo thuật toán Kruskal cho đồ thị biểu diễn dưới dạng danh sách trọng số được thể hiện dưới đây với các thủ tục:

• Thủ tục Init(): đọc dữ liệu biểu diễn bằng danh sách trọng số.

• Thủ tục Heap(): sắp xếp các cạnh theo thứ tự tăng dần của trọng số bằng thuật toán Heap Sort.

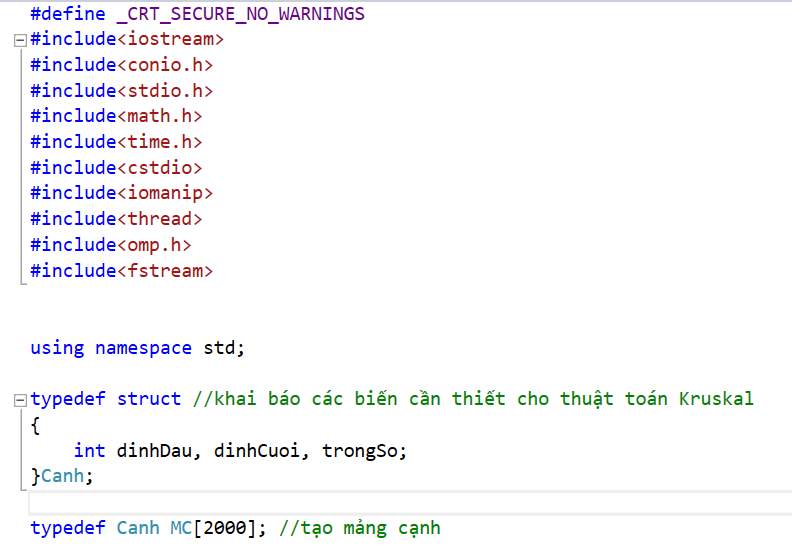
• Thủ tục Find(),Union() : tìm và kiểm tra khi kết nào cạnh vào cây khung có tạo nên chu trình hay không.

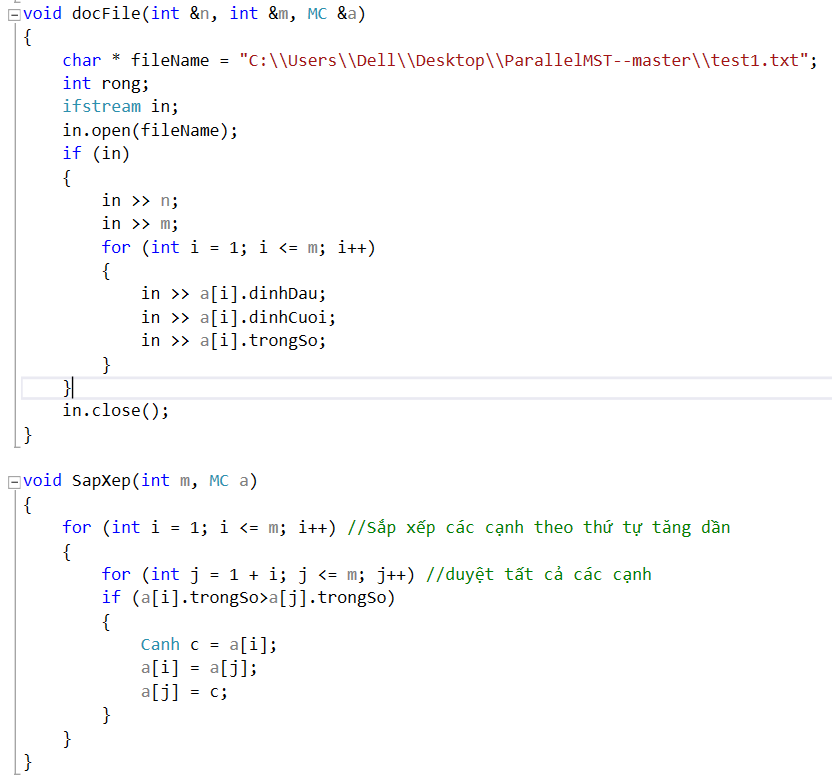
• Thủ tục Result() : đưa ra tập cạnh và độ dài nhỏ nhất của cây khung.

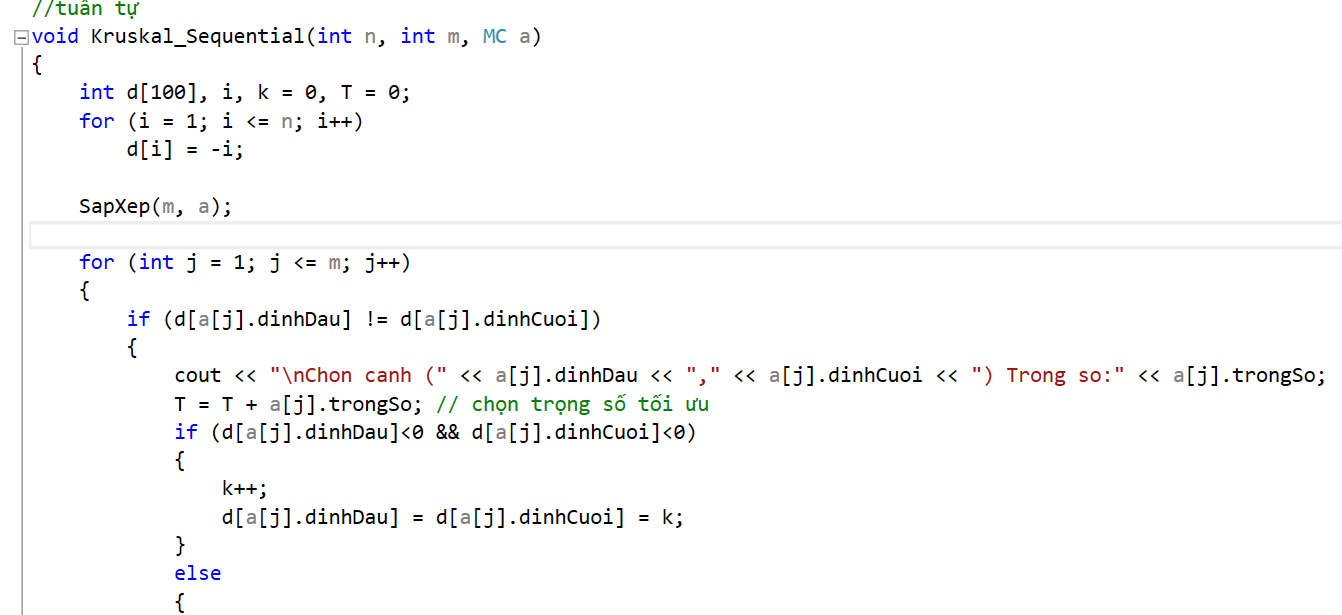
• Tuy nhiên, ở đây nhóm mình sẽ viết gom lại được trình bày ở phần tiếp theo.

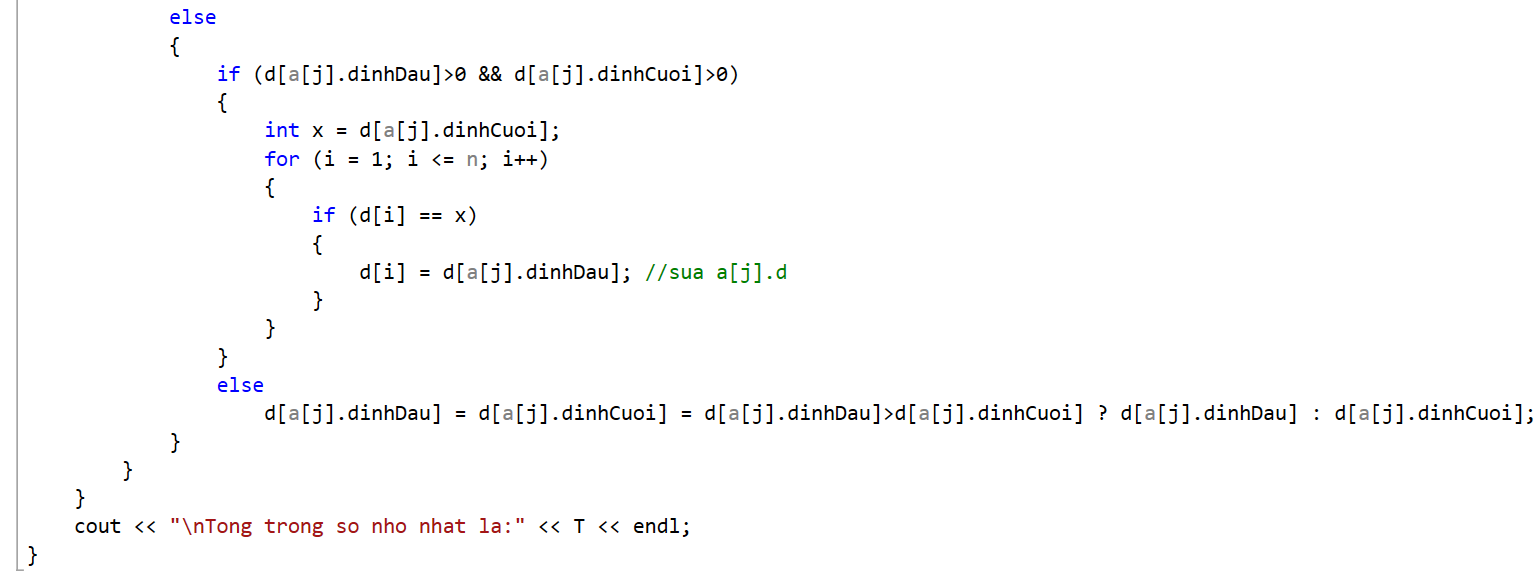
**4.2** **Code chương trình**

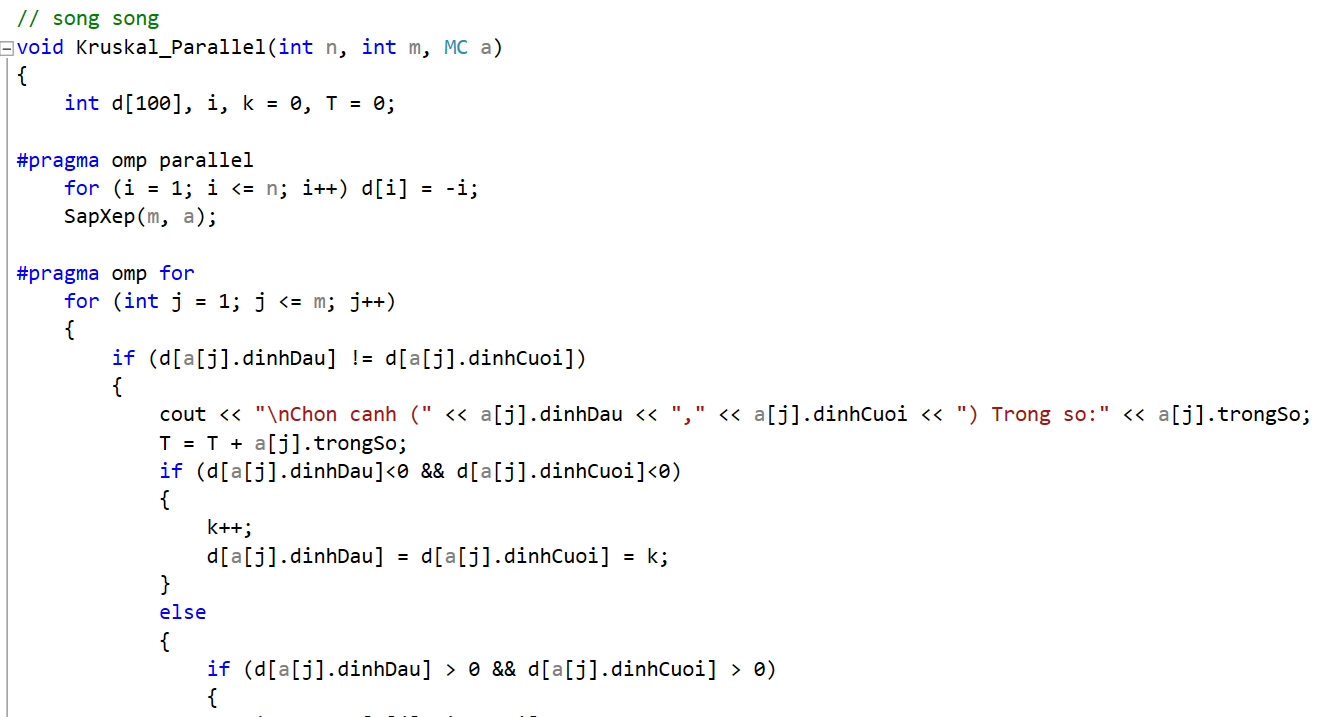
Chương trình tìm cây khung nhỏ nhất theo thuật toán Kruskal được thể hiện như sau:

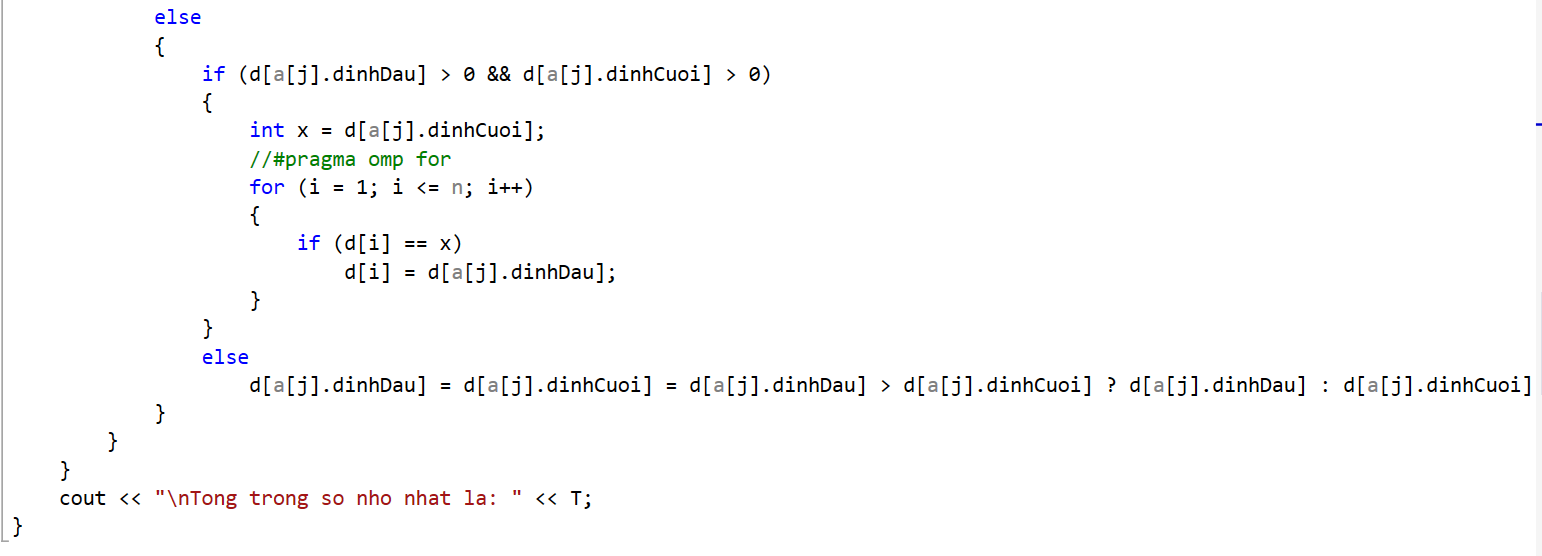


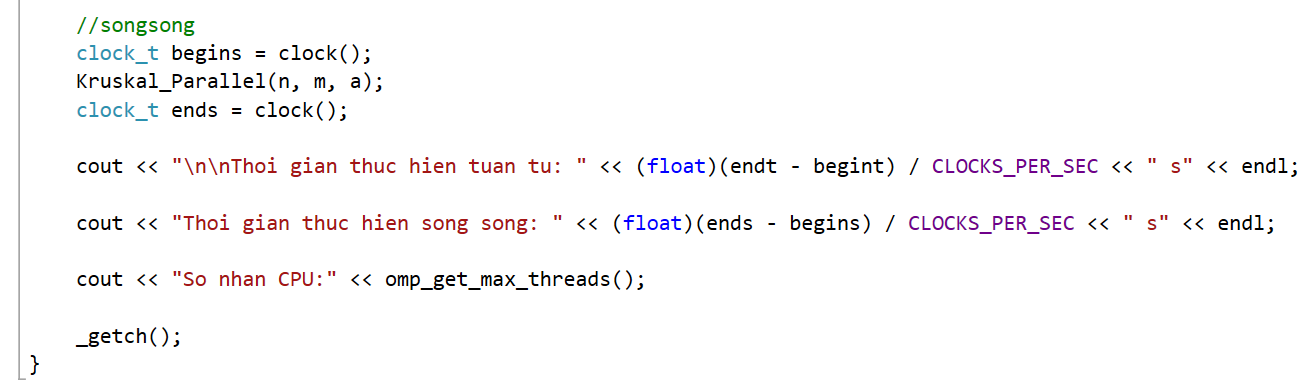
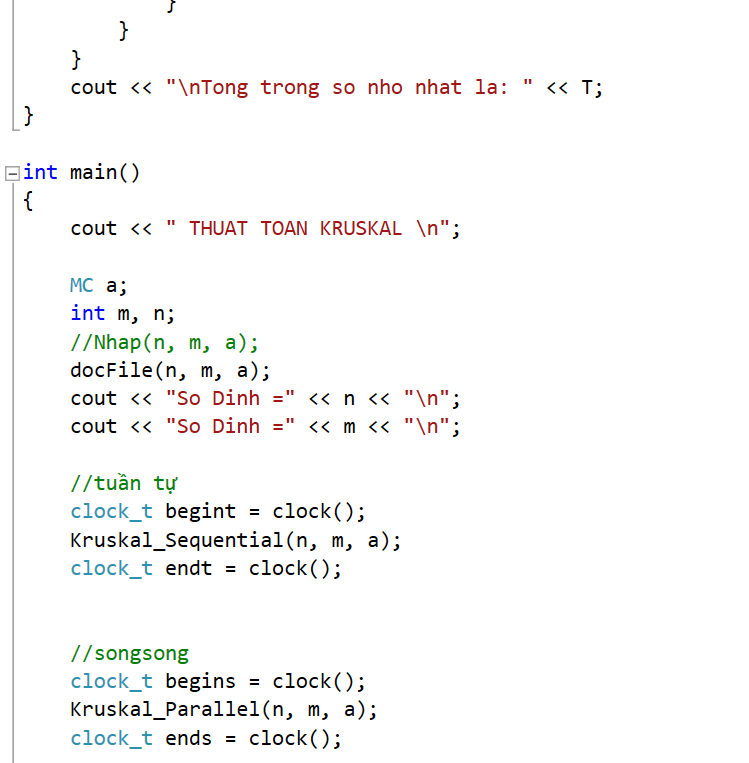












**4.3 Tài liệu tham khảo**

<https://topdev.vn/blog/hadoop-la-gi/>

<https://viblo.asia/p/tim-hieu-ve-hadoop-bJzKmOBXl9N>

<https://bizflycloud.vn/tin-tuc/hadoop-la-gi-gioi-thieu-nhanh-ve-cong-cu-phan-tich-big-data-pho-bien-nhat-the-gioi-20181222102718157.htm>

<https://www.giaithuatlaptrinh.com/?p=1140>

<https://vi.wikipedia.org/wiki/Thu%E1%BA%ADt_to%C3%A1n_Kruskal>

<https://github.com/account/unverified-email>

Hết