BÁO CÁO CÔNG VIỆC

**Công việc số:** 53

**Tên công việc:** Hỗ trợ Tú chay code C# thuật toán Apriori từ [**Github**](https://github.com/mhdbouk/Apriori) với dữ liệu, tách lõi thuật toán

**Người thực hiện:** Hạ Quang Dũng

**Ngày bắt đầu:** 16/09/2024

**Ngày kết thúc:** 20/09/2024

# **Mục lục**

[**Mục lục 1**](#_mk7b1h3jycpl)

[**1. Lõi thuật toán 3**](#_cg5ejkysd7lx)

[1.1. ItemSet 3](#_nmwgvrvns9si)

[1.2. AssociationRule 3](#_ai00614fbx98)

[1.3. Helper 4](#_21r5nl7m11zv)

[1.4. Apriori 5](#_q20tkoxj4dmo)

[**2. Cách sử dụng 9**](#_6ikhgxv25if)

[**3. Kết quả đạt được 12**](#_8aulqraaf88k)

[3.1. 50 mẫu dữ liệu 12](#_lu1g3vjpm9rk)

[3.2. 100 mẫu dữ liệu 13](#_d5j6isf96gqh)

[3.3. 200 mẫu dữ liệu 14](#_y57kfh7j8dyx)

[3.4. 300 mẫu dữ liệu 15](#_ljnhh4p1t0gc)

[3.5. 400 mẫu dữ liệu 16](#_vln01jwwd0)

[3.6. 500 mẫu dữ liệu 17](#_kthnt46z0947)

# 1. Lõi thuật toán

Sử dụng Class Liblrary để code thuật toán

## 1.1. ItemSet

ItemSet là một cấu trúc dữ liệu quan trọng trong thuật toán Apriori, đại diện cho một tập hợp các mục (items) trong một bộ dữ liệu. Mỗi itemset thường biểu diễn một nhóm các đối tượng hoặc sự kiện liên quan được tìm thấy trong các giao dịch.

**Chức năng chính:**

* Quản lý danh sách các mục trong một giao dịch hoặc trong một tập hợp con (subset) của giao dịch.
* Lưu trữ thông tin về mức độ xuất hiện (frequency) của một itemset trong bộ dữ liệu, phục vụ cho việc tính toán độ phổ biến (support).
* Hỗ trợ so sánh, kiểm tra tính bằng nhau, và kết hợp các itemset với nhau để sinh ra các tập lớn hơn trong quá trình xây dựng các tập hợp phổ biến.

**Ứng dụng:**Được sử dụng trong việc tìm kiếm các tập mục phổ biến (frequent itemsets) trong quá trình chạy thuật toán Apriori, dựa vào các ngưỡng hỗ trợ (support threshold).

| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace AprioriAlgorithm  {  public class ItemSet : Dictionary<List<string>, int>  {  public string Label { get; set; }  public int Support { get; set; }  }  } |
| --- |

## 1.2. AssociationRule

AssociationRule biểu diễn các luật kết hợp (association rules) được phát hiện từ các tập mục phổ biến. Các luật kết hợp này chỉ ra mối quan hệ giữa các mục trong các tập hợp dữ liệu, ví dụ: "nếu một khách hàng mua item A thì họ có xu hướng mua item B."

**Chức năng chính:**

* Định nghĩa các luật kết hợp theo dạng "nếu-thì" (if-then), trong đó "nếu" là phần vế trái (antecedent) và "thì" là phần vế phải (consequent) của luật.
* Lưu trữ các giá trị đo lường liên quan đến một luật, bao gồm độ hỗ trợ (support), độ tin cậy (confidence), và các chỉ số khác như lift hay conviction.
* Cung cấp chức năng kiểm tra và tính toán độ tin cậy của các luật dựa trên các itemset đã phát hiện được.

**Ứng dụng:**Được sử dụng để khám phá ra những mẫu hoặc quy luật hữu ích từ dữ liệu, ví dụ trong các hệ thống gợi ý sản phẩm, phân tích hành vi người dùng.

| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace AprioriAlgorithm  {  public class AssociationRule  {  public string Label { get; set; }  public double Confidance { get; set; }  public double Support { get; set; }  }  } |
| --- |

## 1.3. Helper

Helper là tập hợp các hàm và công cụ hỗ trợ (utility functions) giúp cho việc tính toán và xử lý dữ liệu trong quá trình thực hiện thuật toán Apriori. Mục tiêu chính của lớp này là giảm bớt sự phức tạp trong các thao tác cơ bản và cung cấp các phương pháp tiện lợi.

**Chức năng chính:**

* Cung cấp các hàm để xử lý dữ liệu như việc sinh ra các tập hợp con (subsets) của một tập hợp item, điều này rất quan trọng trong quá trình tạo ra các tập hợp mục ứng viên (candidate itemsets).
* Thực hiện các phép toán trên tập hợp như kiểm tra xem một tập hợp con có tồn tại trong một tập hợp lớn hơn hay không, hoặc kết hợp hai tập hợp item lại với nhau.
* Hỗ trợ các phép tính thống kê hoặc các hàm toán học liên quan đến việc tính toán độ hỗ trợ, độ tin cậy và các thước đo khác cho các luật kết hợp.

**Ứng dụng:**Giúp tăng tốc độ và đơn giản hóa các bước tính toán trong thuật toán Apriori, giảm thiểu các đoạn mã lặp lại và giúp mã nguồn dễ bảo trì hơn.

| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace AprioriAlgorithm  {  public static class Helper  {  public static string ToDisplay(this List<string> list, string separator = ", ")  {  if (list.Count == 0)  return string.Empty;  StringBuilder sb = new StringBuilder();  sb.Append(list[0]);  for (int i = 1; i < list.Count; i++)  {  sb.Append(string.Format("{0}{1}", separator, list[i]));  }  return sb.ToString();  }  public static string ToDisplay(this List<string> list, string exclude, string separator = ", ")  {  List<string> dump = new List<string>();  foreach (var item in list)  {  if (item == exclude) continue;  dump.Add(item.ToString());  }  return dump.ToDisplay();  }  public static string ToPercentString(this object item)  {  return item.ToString() + " %";  }  }  } |
| --- |

## 1.4. Apriori

Apriori là thành phần chính thực thi thuật toán Apriori, dùng để khai phá các tập hợp mục phổ biến và các luật kết hợp từ một tập dữ liệu lớn. Thuật toán này hoạt động theo nguyên tắc "Apriori" - nghĩa là mọi tập hợp con của một tập hợp mục phổ biến cũng phải là phổ biến.

**Chức năng chính:**

* Xác định và trích xuất các tập hợp mục phổ biến (frequent itemsets) từ tập dữ liệu dựa trên ngưỡng hỗ trợ (support threshold) do người dùng đặt ra.
* Sinh ra các ứng viên (candidate itemsets) từ các tập hợp con đã tìm được và kiểm tra xem chúng có đạt điều kiện phổ biến không.
* Tính toán các luật kết hợp từ các tập hợp mục phổ biến bằng cách sử dụng độ tin cậy (confidence) và các ngưỡng khác để lọc ra những luật có giá trị.
* Cung cấp các phương pháp và giao diện để cấu hình thuật toán, ví dụ như thay đổi ngưỡng hỗ trợ, độ tin cậy hoặc đầu vào của tập dữ liệu.

**Ứng dụng:**Được áp dụng trong các hệ thống phân tích dữ liệu lớn, đặc biệt là trong các lĩnh vực như bán lẻ (phân tích giỏ hàng), tài chính (phân tích hành vi giao dịch), và các hệ thống đề xuất sản phẩm hoặc dịch vụ.

| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  namespace AprioriAlgorithm  {  public class Apriori  {  private List<string> list;  private List<string> distinctValues;  private List<ItemSet> itemSets;  // 1. Constructor mới nhận List<string> làm đầu vào  public Apriori(List<string> transactions)  {  list = transactions.Where(a => !string.IsNullOrWhiteSpace(a)).ToList();  itemSets = new List<ItemSet>();  SetDistinctValues(list);  }  // 2. Setup Method: Thiết lập danh sách các giá trị duy nhất  public void SetDistinctValues(List<string> values)  {  var data = new HashSet<string>();  foreach (var item in values)  {  var row = item.Split(' ');  foreach (var value in row)  {  if (!string.IsNullOrWhiteSpace(value))  data.Add(value);  }  }  distinctValues = data.OrderBy(a => a).ToList();  }  // 3. Core Method: Tìm tập mục phổ biến dựa trên chiều dài (length) và ngưỡng hỗ trợ (support)  public ItemSet GetItemSet(int length, int support, bool candidates = false, bool isFirstItemList = false)  {  var result = GetPermutations(distinctValues, length).ToList();  var data = result.Select(p => p.ToList()).ToList();  var itemSet = new ItemSet  {  Support = support,  Label = (candidates ? "C" : "L") + length.ToString()  };  foreach (var item in data)  {  int count = list.Count(word => item.All(i => word.Split(' ').Contains(i)));  if ((candidates && count > 0) || (isFirstItemList && count >= support) || count >= support)  {  itemSet[item] = count;  itemSets.Add(itemSet);  }  }  return itemSet;  }  // 4. Helper Method: Tìm tất cả các tổ hợp (combinations) có độ dài count từ danh sách đầu vào  public static IEnumerable<IEnumerable<T>> GetPermutations<T>(IEnumerable<T> items, int count)  {  int i = 0;  foreach (var item in items)  {  if (count == 1)  yield return new T[] { item };  else  {  foreach (var result in GetPermutations(items.Skip(i + 1), count - 1))  yield return new T[] { item }.Concat(result);  }  ++i;  }  }  // 5. Tìm các luật kết hợp từ tập mục phổ biến  public List<AssociationRule> GetRules(ItemSet items)  {  var rules = new List<AssociationRule>();  foreach (var item in items)  {  foreach (var set in item.Key)  {  rules.Add(GetSingleRule(set, item));  if (item.Key.Count > 2)  rules.Add(GetSingleRule(string.Join(", ", item.Key.Except(new[] { set })), item));  }  }  return rules.OrderByDescending(r => r.Support).ThenByDescending(r => r.Confidance).ToList();  }  // 6. Tạo một luật kết hợp đơn lẻ từ tập mục  private AssociationRule GetSingleRule(string set, KeyValuePair<List<string>, int> item)  {  var setItems = set.Split(',').Select(x => x.Trim()).ToList();  var rule = new AssociationRule();  var sb = new StringBuilder();  sb.Append(set).Append(" => ");  var list = item.Key.Except(setItems).ToList();  sb.Append(string.Join(", ", list));  rule.Label = sb.ToString();  // Tính toán hỗ trợ của luật  int totalSet = 0;  var foundSetItem = itemSets.FirstOrDefault(setItem => setItem.ContainsKey(setItems));  if (foundSetItem != null)  {  foundSetItem.TryGetValue(setItems, out totalSet);  }  foreach (var first in itemSets)  {  var myItem = first.Keys.Where(a => a.ToDisplay() == set);  if (myItem.Count() > 0)  {  first.TryGetValue(myItem.FirstOrDefault(), out totalSet);  break;  }  }  rule.Confidance = Math.Round(((double)item.Value / totalSet) \* 100, 2);  rule.Support = Math.Round(((double)item.Value / this.list.Count) \* 100, 2);  return rule;  }  }  } |
| --- |

# 2. Cách sử dụng

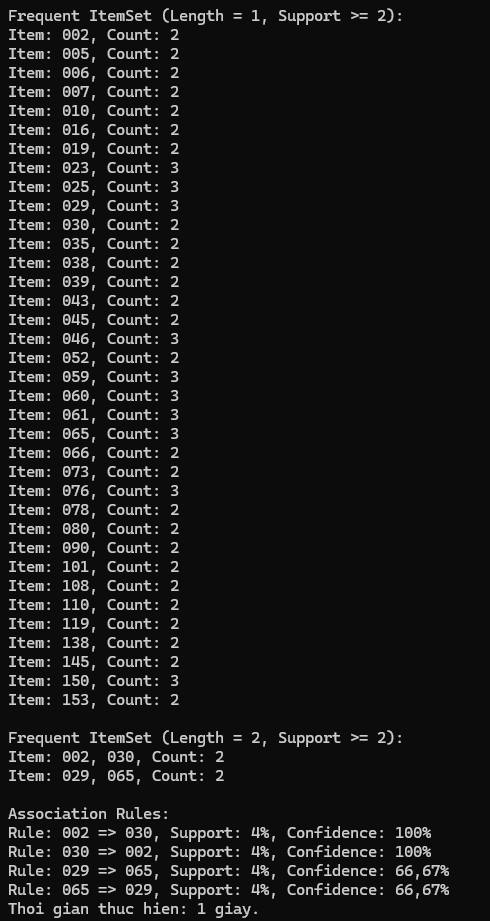
Bộ dữ liệu được lấy từ [Retail Transactions Dataset](https://www.kaggle.com/datasets/prasad22/retail-transactions-dataset) và được chuyển đổi sang code từ 001 - 162 để phù hợp đầu vào chương trình.

Chỉnh sửa danh sách giao dịch và ngưỡng hỗ trợ

| using System;  using System.Collections.Generic;  using System.Linq;  using AprioriAlgorithm;  using System.Text.RegularExpressions;  using System.Diagnostics;  namespace ConsoleApp  {  internal class Program  {  static void Main(string[] args)  {  // Example usage of the new function  List<string> transactions = new List<string>  {  "119 061 041",  "114 043 046 008 055",  "147",  "151 045",  "104",  "155 016",  "113 006 065 046 029",  "149 076 049 080 042",  "115",  "145 003 065",  "108 065 080 029 019",  "150 050 066",  "140 061",  "156 002 044 030",  "138 039 007",  "094 079 005",  "082 028",  "137 032 038 069 060",  "126 025",  "101",  };  int support = 2; // Ngưỡng hỗ trợ tối thiểu  Stopwatch stopwatch = new Stopwatch();  stopwatch.Start();  RunApriori(transactions, support);  stopwatch.Stop();  // Đổi từ mili giây sang giây  double seconds = stopwatch.ElapsedMilliseconds / 1000;  Console.WriteLine($"Thời gian thực hiện: {seconds} giây.");  // Kết thúc chương trình  Console.WriteLine("\nPress any key to exit...");  Console.ReadKey();  }  /// <summary>  /// Chạy thuật toán Apriori với danh sách giao dịch và ngưỡng hỗ trợ được chỉ định.  /// </summary>  /// <param name="transactions">Danh sách giao dịch (mỗi giao dịch là một chuỗi chứa các mục, cách nhau bằng dấu cách).</param>  /// <param name="support">Ngưỡng hỗ trợ tối thiểu cho các tập phổ biến.</param>  static void RunApriori(List<string> transactions, int support)  {  int k = 1;  List<ItemSet> itemSets = new List<ItemSet>();  bool next;  // Khởi tạo đối tượng Apriori với danh sách giao dịch  Apriori apriori = new Apriori(transactions);  do  {  next = false;  // Lấy tập mục phổ biến có độ dài k với ngưỡng hỗ trợ tối thiểu  var L = apriori.GetItemSet(k, support);  if (L.Count > 0)  {  List<AssociationRule> rules = new List<AssociationRule>();  // Tạo các luật kết hợp nếu k != 1  if (k != 1)  {  rules = apriori.GetRules(L);  }  // Hiển thị tập mục phổ biến  Console.WriteLine($"\nFrequent ItemSet (Length = {k}, Support >= {support}):");  foreach (var item in L)  {  Console.WriteLine($"Item: {string.Join(", ", item.Key)}, Count: {item.Value}");  }  // Hiển thị các luật kết hợp  if (rules.Count > 0)  {  Console.WriteLine("\nAssociation Rules:");  foreach (var rule in rules)  {  Console.WriteLine($"Rule: {rule.Label}, Support: {rule.Support}%, Confidence: {rule.Confidance}%");  }  }  next = true;  k++;  itemSets.Add(L);  }  } while (next);  }  }  } |
| --- |

# 3. Kết quả đạt được

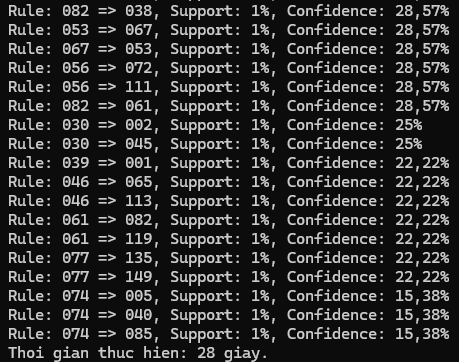
## 3.1. 50 mẫu dữ liệu



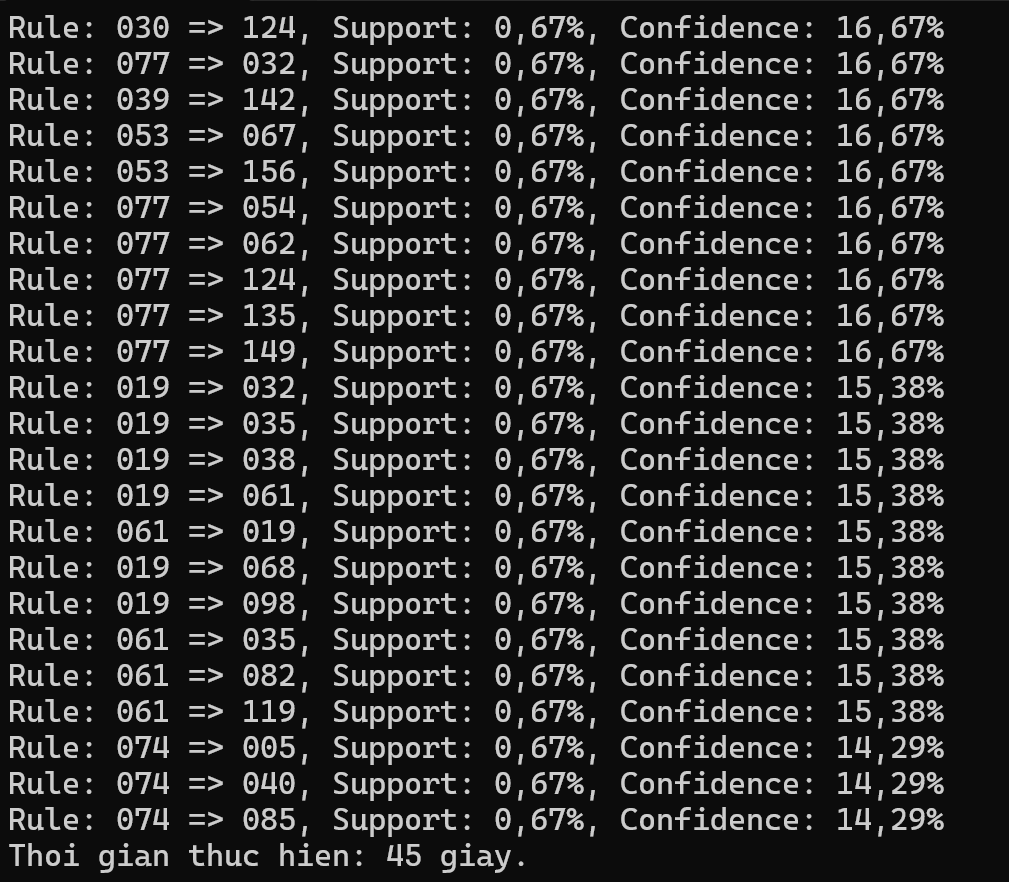
## 3.2. 100 mẫu dữ liệu



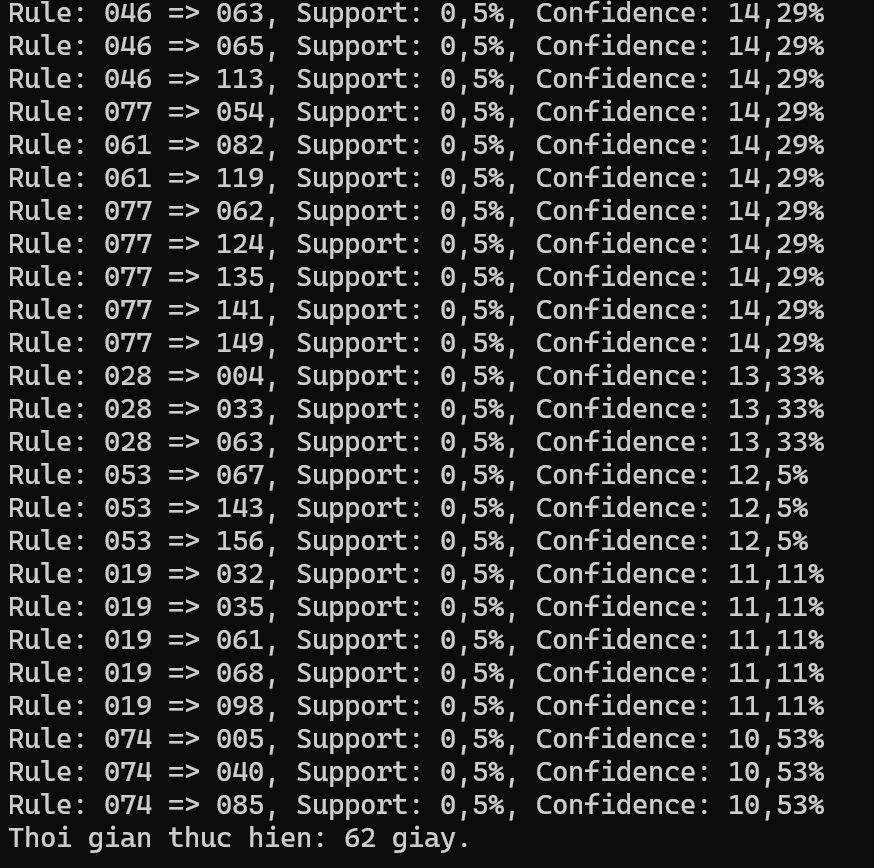
## 3.3. 200 mẫu dữ liệu



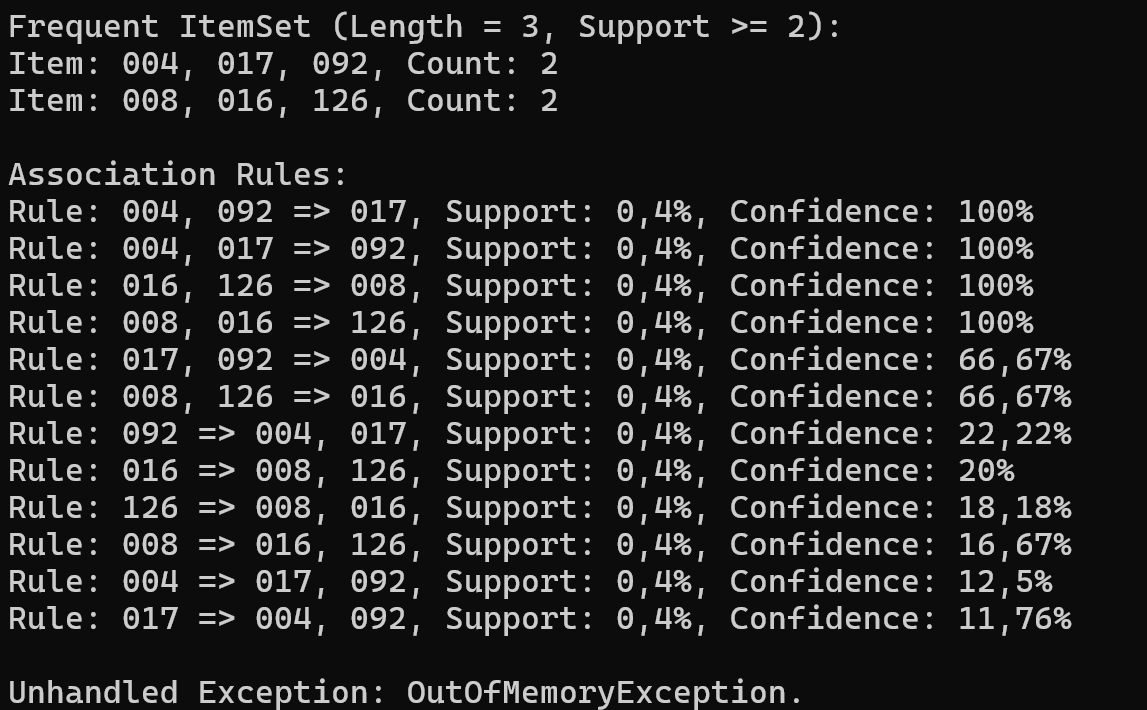
## 3.4. 300 mẫu dữ liệu



## 3.5. 400 mẫu dữ liệu



## 3.6. 500 mẫu dữ liệu



Từ mẫu thứ 500 trở đi ta bị tràn bộ nhớ do phải xử lý bộ dữ liệu quá lớn.