

# Tungnt.net



Home » So sánh tốc độ List collection và HashSet collection trong C#

## So sánh tốc độ List collection và HashSet collection trong C#

30/11/2016 - 06/12/2016 by tungnt185 — Leave a Comment

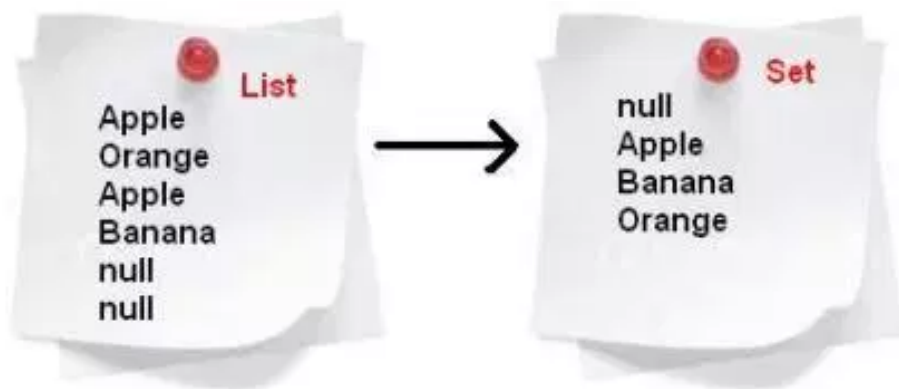


## Mở đầu

5 (100%) 3 vote[s]

việc sử dụng kiểu **List** nên nhớ quên này không phải để thay đổi.

Trong bài viết này chúng ta sẽ cùng so sánh tốc độ của kiểu **List** và **HashSet** (một kiểu collection xuất hiện bắt đầu từ **.NET 3.5**) để biết được khi nào thì nên sử dụng **List** và khi nào thì nên dùng kiểu **HashSet**.



## Kiểu List<T>

Kiểu **List<T>** chứa một danh sách item, truy cập qua chỉ số. Không giống như kiểu mảng là một kiểu cố định kích thước, list tự động tăng kích thước khi cần. Đây chính là lý do tại sao List đôi khi còn được gọi là kiểu mảng động.

Cấu trúc bên trong của List thực chất là một mảng, nếu mảng này hết chỗ nó sẽ tạo ra một mảng mới lớn hơn và copy toàn bộ item từ mảng cũ sang mảng mới. Ngày nay thì chúng ta thấy hầu hết lập trình viên đều dùng kiểu List thay vì kiểu mảng. Tuy nhiên trong một số trường hợp nếu item là cố định thì bạn nên dùng kiểu mảng để có performance tốt hơn.

```
1  {
2  {
3      var intList = new List<int>(100);
4
5      intList.Add(5);
6      intList.RemoveAt(0);
7
8      //Add item at index
9      intList.Insert(0, 10);
10     intList.Insert(1, 7);
11
12     var first = intList[0];
13
14     // Index of an item
15     var index = intList.IndexOf(4);
16
17     // Check List contains item
18     bool contain = intList.Contains(4);
19
20     // Iterate over all objects
21     foreach (var item in intList)
22         Console.WriteLine(item);
23 }
```

## Thêm/Xóa phần tử ở đầu hoặc giữa List

Nếu bạn cần thêm/xóa một phần tử ở đầu hoặc giữa List thì nó sẽ cần dịch chuyển một hoặc nhiều item. Trường hợp xấu nhất, nếu bạn thêm/xóa một phần tử ở đầu **List** nó sẽ phải dịch tất cả item còn lại trong **List**. Nếu List càng lớn thì sẽ càng mất thời gian và resource để thực hiện.

## Thêm/Xóa phần tử ở cuối List

Thêm/Xóa một phần tử ở cuối **List** ngược lại rất nhanh và không phụ thuộc vào kích thước của List vì không item nào cần phải dịch chuyển.

Như tìm kiếm một phần tử đang một số hàm như **indexOf**, **contains**, **Find** thì kiểu **List** sẽ thực hiện duyệt qua tất cả các item để tìm ra phần tử đó. Trong trường hợp xấu nhất, nếu phần tử nằm ở cuối **List** thì tất cả các item cần được duyệt qua.

TÌM KIẾM CÓ TÀI TRỢ



Internet servers

Apache server

## Truy cập một phần tử bởi chỉ số

Là phép toán thực hiện rất nhanh vì bản chất bên trong List là một mảng. Tốc độ truy cập không bị ảnh hưởng bởi độ lớn của List.

## Kiểu HashSet<T>

**HashSet** là kiểu xuất hiện từ .NET 3.5, nó chứa các item duy nhất, nghĩa là nó không thể chứa các item trùng lặp và thứ tự của item cũng không quan trọng. VD: {a,b,c} tương đương {c,b,a}

Sử dụng **HashSet** khi bạn cần tìm kiếm nhanh trên một tập các item không lặp. VD trong bài toán thực tế mình đang phải làm là kiểm tra một danh sách **Order** có nằm trong một danh sách **Order** khác không?

Kiểu **HashSet** dựa trên hàm hash để tính toán ra chỉ số của item vì vậy phép tìm kiếm có tốc độ rất nhanh. Dưới đây là một số phép toán trên kiểu **HashSet**

```
1 private static void HashSetOperation()  
2 {  
3     var intHashSet = new HashSet<int>() { 1, 2, 3, 4, 5 };
```

```
8 //Check Set contains item
9 bool contain = intHashSet.Contains(1);
10
11 // Iterate over all objects
12 foreach (var item in intHashSet)
13     Console.WriteLine(item);
14
15 //Delete all items
16 intHashSet.Clear();
17 }
```

## So sánh tốc độ List<T> và HashSet<T>

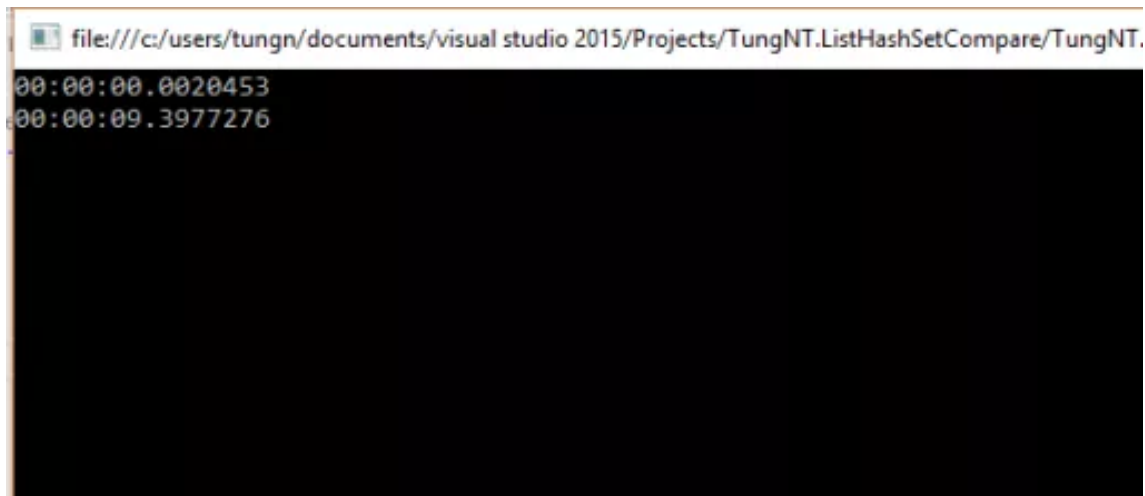
Mặc dù trong các tài liệu đều chỉ ra kiểu **HashSet** nhanh hơn **List** nhưng thực tế có phải luôn luôn như vậy? Chúng ta sẽ cùng kiểm nghiệm qua một ví dụ benchmark đơn giản: Kiểm tra một danh sách có nằm trong một danh sách khác hay không:

```
C#
2 {
3     //const int COUNT = 50;
4     const int COUNT = 50000;
5
6     HashSet<int> intHashSet = new HashSet<int>();
7     Stopwatch stopWatch = new Stopwatch();
8     for (int i = 0; i < COUNT; i++)
9     {
10         intHashSet.Add(i);
11     }
12
13     stopWatch.Start();
14     for (int i = 0; i < COUNT; i++)
15     {
16         intHashSet.Contains(i);
17     }
18     stopWatch.Stop();
19
20     Console.WriteLine(stopWatch.Elapsed);
21
22     stopWatch.Reset();
```

```
27     }
28
29     stopwatch.Start();
30     for (int i = 0; i < COUNT; i++)
31     {
32         intList.Contains(i);
33     }
34     stopwatch.Stop();
35
36     Console.WriteLine(stopwatch.Elapsed);
37 }
```

Kết quả thực hiện cho thấy sự khác nhau rõ rệt về performance:

- 50 item **List.Contains** mất **0.000018s**, **HashSet.Contains** mất **0.000023s**
- 50000 item **List.Contains** mất **9.38s**, **HashSet.Contains** mất **0.002s**



Từ kết quả trên có thể thấy rằng không phải lúc nào performance của **HashSet** cũng nhanh hơn **List**. Với số lượng item nhỏ (**<50 item**) thì tốc độ của **List** còn nhanh hơn so với **HashSet** nhưng khi số lượng item

# Kết luận

Kiểu **List** tương đối nhanh khi bạn truy cập item bằng index nhưng khi tìm kiếm một item trong List lớn (50 item trở lên) thì nó sẽ chậm dần. Ngược lại kiểu **HashSet** lại rất nhanh khi tìm kiếm trên một tập item lớn. Vì vậy mà tùy từng trường hợp cụ thể bạn cần lựa chọn kiểu collection cho phù hợp với bài toán của mình.

*Nếu bạn có bất kì câu hỏi hay kinh nghiệm nào liên quan đến bài viết này xin hãy để lại comment bên dưới và đừng quên chia sẻ cho bạn bè nếu thấy hữu ích.*

Happy coding. Stay tuned.

P/s: Source code example trong bài các bạn có thể download tại đây: [ListAndHashSetBenchmark](#)

TÌM KIẾM CÓ TÀI TRỢ



Api mvc

Api rest

Share this:

Share 67

Tweet

SHARE

 Email

Save

Like this:

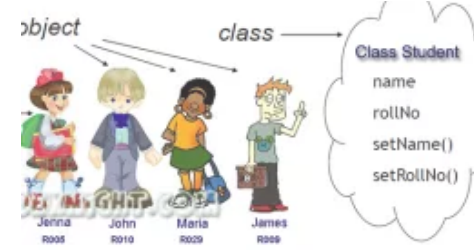
Loading...

[HOME](#)[ABOUT](#)[WEB](#)[MOBILE](#)[ARCHITECTURE](#)[TOOL](#)[OTHERS](#)[🇻🇳 TIẾNG VIỆT](#)[🇺🇸 ENGLISH](#)

Tính năng mới của từng phiên bản  
.NET Framework  
In ".NET Framework"



Giới thiệu Microsoft .NET Framework  
In ".NET Framework"



Các nguyên lý của lập trình hướng  
đối tượng (Object Oriented  
Programming)  
In "Object Oriented Programming"

Filed Under: .NET Framework

Tagged With: HashSet

## Leave a Reply

Your email address will not be published. Required fields are marked \*

Name \*

Email \*



[HOME](#)

[ABOUT](#)

[WEB](#)

[MOBILE](#)

[ARCHITECTURE](#)

[TOOL](#)

[OTHERS](#)

 [TIẾNG VIỆT](#)

 [ENGLISH](#)

## Comment

POST COMMENT

☐ Notify me of follow-up comments by email.

☐ Notify me of new posts by email.

Search the site ...

MY PROFILE

HOME

ABOUT

WEB

MOBILE

ARCHITECTURE

TOOL

OTHERS

 TIẾNG VIỆT

 ENGLISH



SUBSCRIBE TO BLOG VIA EMAIL

Enter your email address to subscribe to this blog  
and receive notifications of new posts by email.  
Join 2,265 other subscribers

[HOME](#)[ABOUT](#)[WEB](#)[MOBILE](#)[ARCHITECTURE](#)[TOOL](#)[OTHERS](#)[🇻🇳 TIẾNG VIỆT](#)[🇺🇸 ENGLISH](#)

## TOP POSTS & PAGES



Sử dụng các HTTP Verbs GET  
POST PUT DELETE trong Web API



Sự khác nhau giữa Web Service,  
WCF, WCF REST, Web API



Cách tạo asp.net web api service  
dùng cho mobile app

[HOME](#)

[ABOUT](#)

[WEB](#)

[MOBILE](#)

[ARCHITECTURE](#)

[TOOL](#)

[OTHERS](#)

 [TIẾNG VIỆT](#)

 [ENGLISH](#)

[Programming](#)



Fiddler

[Sử dụng fiddler test web service](#)

## ARCHIVES

[May 2017 \(1\)](#)

[January 2017 \(1\)](#)

[December 2016 \(1\)](#)

[November 2016 \(1\)](#)

[October 2016 \(1\)](#)

[August 2016 \(1\)](#)

[July 2016 \(1\)](#)

[June 2016 \(1\)](#)

[May 2016 \(1\)](#)

[April 2016 \(1\)](#)

[March 2016 \(1\)](#)

[January 2016 \(2\)](#)

[HOME](#)[ABOUT](#)[WEB](#)[MOBILE](#)[ARCHITECTURE](#)[TOOL](#)[OTHERS](#)[!\[\]\(8bba887393ca45b761e5cb49e755e762\_img.jpg\) TIẾNG VIỆT](#)[!\[\]\(6bb0e4f14c4133b37d2887cb37e67ddd\_img.jpg\) ENGLISH](#)

---

[September 2015 \(3\)](#)

---

---

[August 2015 \(2\)](#)

---

---

[July 2015 \(1\)](#)

---

---

[June 2015 \(1\)](#)

---

---

[May 2015 \(2\)](#)

---

---

[April 2015 \(2\)](#)

---

---

[March 2015 \(5\)](#)

---

---

[February 2015 \(2\)](#)

---

---

[January 2015 \(2\)](#)

---

---

[November 2014 \(2\)](#)

---

---

[October 2014 \(1\)](#)

---

---

[September 2014 \(1\)](#)

---

---

[August 2014 \(1\)](#)

---

---

[July 2014 \(2\)](#)

---

---

[May 2014 \(2\)](#)

---

---

[January 2014 \(1\)](#)

---

---

[November 2013 \(1\)](#)

---

[HOME](#)

[ABOUT](#)

[WEB](#)

[MOBILE](#)

[ARCHITECTURE](#)

[TOOL](#)

[OTHERS](#)

 [TIẾNG VIỆT](#)

 [ENGLISH](#)

---

January 2012 (1)

---

March 2011 (4)

---

January 2011 (2)

---

June 2010 (1)

---

## CATEGORIES

.NET Framework (6)

---

Architecture (6)

---

ASP.NET (1)

---

ASP.NET MVC (2)

---

Blogging Technique (1)

---

Database (3)

---

Docker (2)

---

Entity Framework (1)

---

Expression Blend (2)

---

Javascript (1)

---

[HOME](#)[ABOUT](#)[WEB](#)[MOBILE](#)[ARCHITECTURE](#)[TOOL](#)[OTHERS](#)[!\[\]\(3cb60d42b10e53f9522bb0b392c1c4cd\_img.jpg\) TIẾNG VIỆT](#)[!\[\]\(d0262bbe9d2356661a2e89321dfcc781\_img.jpg\) ENGLISH](#)

---

[Others \(9\)](#)

---

---

[SignalR \(1\)](#)

---

---

[SoftSkill \(3\)](#)

---

---

[Team Foundation Server \(4\)](#)

---

---

[Tool \(1\)](#)

---

---

[Visual Studio \(6\)](#)

---

---

[WCF \(5\)](#)

---

---

[Web API \(5\)](#)

---

---

[Windows Phone \(9\)](#)

---

## TAGS

[.NET](#)[ASP.NET MVC 6](#)[AuthenticationService](#)[BDD](#)[Bootstrap](#)[Clean](#)[Clean Code](#)[Conferences](#)[Continuous Delivery](#)[Continuous Integration](#)[CultureInfo](#)[Fiddler](#)

[HOME](#)

[ABOUT](#)

[WEB](#)

[MOBILE](#)

[ARCHITECTURE](#)

[TOOL](#)

[OTHERS](#)

 [TIẾNG VIỆT](#)

 [ENGLISH](#)

[Lập trình hướng đối tượng](#)

[membership](#)

[Microsoft Technology Day](#)

[Neo4j](#)

[nosql](#)

[OOP](#)

[Refactoring](#)

[SignalR](#)

[Soft Skill](#)

[SQL Optimization](#)

[SQL Server](#)

[TDD](#)

[Team Foundation Server 2010](#)

[Team Foundation Server 2013](#)

[Team Foundation Server 2015](#)

[Test Driven Development](#)

[Testing](#)

[Tips&Trick](#)

[UI Prototype](#)

[Visual Studio 2010](#)

[Visual Studio 2013](#)

[Visual Studio 2015](#)

[WCF](#)

[WCF REST](#)

[Web API](#)

[Web Service](#)

[Windows Phone 7](#)

[Windows Phone 8.1](#)

[Winform](#)

[FOLLOW ME ON TWITTER & FACEBOOK](#)

[My Tweets](#)




[HOME](#)[ABOUT](#)[WEB](#)[MOBILE](#)[ARCHITECTURE](#)[TOOL](#)[OTHERS](#)[🇻🇳 TIẾNG VIỆT](#)[🇺🇸 ENGLISH](#)[Like Page](#)[Share](#)

Be the first of your friends to like this

## Hawk Host

Hosting tốc độ cao  
với chất lượng ổn định

[ĐĂNG KÝ NGAY](#) namecheap

## Register Your Domains!

As low as **\$3.98** /year

[Register Now](#)

[HOME](#)

[ABOUT](#)

[WEB](#)

[MOBILE](#)

[ARCHITECTURE](#)

[TOOL](#)

[OTHERS](#)

 [TIẾNG VIỆT](#)

 [ENGLISH](#)

[Arraylist C#](#)

---

Copyright © 2019 · Magazine Pro Theme on Genesis Framework · WordPress · [Log in](#)