Merge Sort

[**DATA STRUCTURE & ALGORITHM**](https://www.stdio.vn/articles/index/13/data-structure-algorithm)

[SHIRO NGUYỄN](https://www.stdio.vn/users/index/98/shiro-nguyen)Merge Sort là một trong những thuật toán sắp xếp có độ phức tạp trung bình và đạt hiệu quả về mặt thời gian. Do đó đối với các chương trình cần tối ưu, Merge Sort là một lựa chọn tốt. Bài viết này tôi sẽ giới thiệu và hướng dẫn các bạn về Merge Sort và cách cài đặt nó trên ngôn ngữ C++.

Giới thiệu

Merge Sort là một trong những thuật toán sắp xếp có độ phức tạp trung bình và đạt hiệu quả về mặt thời gian. Do đó đối với các chương trình cần tối ưu, Merge Sort là một lựa chọn tốt. Bài viết này tôi sẽ giới thiệu và hướng dẫn các bạn về Merge Sort và cách cài đặt nó trên ngôn ngữ C++.

Tiền đề bài viết

Bài viết này nằm trong loạt bài viết về thuật toán sắp xếp.

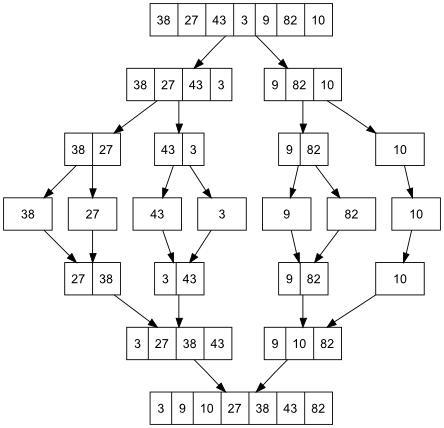
Đối tượng hướng đến

Những lập trình viên [mới bắt đầu](https://training.stdio.vn/trainees/subject/1/cpp-can-ban-den-nang-cao) và dành cho những người muốn tìm hiểu thêm về thuật toán sắp xếp Merge Sort.

Merge Sort

Ý Tưởng

Ý tưởng chúng ta sẽ chia mảng lớn thành những mảng con nhỏ hơn bằng cách chia đôi mảng lớn và chúng ta tiếp tục chia đôi các mảng con cho tới khi mảng con nhỏ nhất chỉ còn 1 phần tử. Sau đó chúng ta sẽ tiếng hành so sánh 2 mảng con có cùng mảng cơ sở (khi chúng ta chia đôi mảng lớn thành 2 mảng con thì mảng lớn đó chúng ta gọi là mảng cơ sở của 2 mảng con đó) khi so sánh chúng sẽ vừa sắp xếp vừa ghép 2 mảng con đó lại thành mảng cơ sở, chúng ta tiếp tục so sánh và ghép các mảng con lại đến khi còn lại mảng duy nhất thì đó là mảng đã được sắp xếp.



Code

1. void Merge(int a[], int left, int mid, int right);
3. // left, right la bien trai va bien phai cua mang
4. void MergeSort(int a[], int left, int right)
5. {
6. if (right > left)
7. {
8. int mid; // Phan tu o giua
9. mid = (left + right) / 2;
10. MergeSort(a, left, mid); // Goi de quy mang con ben trai
11. MergeSort(a, mid + 1, right); // Goi de quy mang con ben phai
12. Merge(a, left, mid, right); // Goi ham so sanh hai mang con
13. }
14. }
16. void Merge(int a[], int left, int mid, int right)
17. {
18. int \*temp; // Khoi tao mang tam de sap xep
19. int i = left; // Vi tri phan tu dau tien cua mang con ben trai
20. int j = mid + 1; // Vi tri phan tu dau tien cua mang con ben phai
22. temp = new int[right - left + 1]; // Khoi tao so luong phan tu cua mang tam
24. for (int k = 0; k <= right - left; k++)
25. {
26. // Kiem phan tu cua mang con ben trai va ben phai
27. if (a[i] < a[j])
28. {
29. // Neu a[i] < a[j] thi copy phan tu ben trai vao mang tam
30. temp[k] = a[i];
31. i++; // Vi tri phan tu tiep theo cua mang
32. }
33. else // Nguoc lai copy phan tu cua mang con ben phai vao mang tam
34. {
35. temp[k] = a[j];
36. j++; // Vi tri phan tu tiep theo cua mang
37. }
39. // Kiem tra mang con ben trai con phan tu nao khong
40. if (i == mid + 1)
41. {
42. // Nguoc lai dua cac phan tu con lai cua mang con ben phai vao mang tam
43. while (j <= right)
44. {
45. k++;
46. temp[k] = a[j];
47. j++;
48. }
49. break;
50. }
52. // Kiem tra mang con ben phai con phan tu nao khong
53. if (j == right + 1)
54. {
55. // Nguoc lai dua cac phan tu con lai cua mang con ben phai vao mang tam
56. while (i <= mid)
57. {
58. k++;
59. temp[k] = a[i];
60. i++;
61. }
62. break;
63. }
64. }
66. for (int k = 0; k <= right - left; k++) // Chep mang tam vao mang chinh
67. a[left + k] = temp[k];
68. delete temp;
69. }
71. int main()
72. {
73. int a[10], n = 10;
74. for (int i = 0; i < n; i++)
75. {
76. cout << "Nhap so " << i + 1 << ": ";
77. cin >> a[i];
78. }
79. MergeSort(a, 0, 9);
80. for (int i = 0; i < n; i++)
81. cout << a[i] << " ";
82. return 0;
83. }

Ưu và nhược điểm

* Ưu điểm: Sắp sếp nhanh hơn so với các thuật toán cơ bản (Insertion Sort, Selection Sort, Interchage Sort), nhanh hơn Quick Sort trong một số trường hợp.
* Nhược điểm: thuật toán khó cài đặt, không nhận dạng được mảng đã được sắp.
* Hiệu quả được tính bằng công thức O(n log n).