Bài 1. Hãy trình bày ngắn gọn tư tưởng của các giải thuật sắp xếp?

|  |  |
| --- | --- |
| Selection sort | Thuật toán này tìm kiếm phần tử nhỏ nhất trong mảng và đưa nó lên đầu mảng, sau đó tìm phần tử nhỏ nhất tiếp theo và đưa lên vị trí tiếp theo của mảng và tiếp tục quá trình này đến khi tất cả các phần tử đã được sắp xếp. |
| Insection sort | Giải thuật sắp xếp một mảng dữ liệu bằng cách chèn lần lượt từng phần tử vào vị trí đúng trong mảng đã sắp xếp trước đó. |
| Interchange sort | Giải thuật sắp xếp bằng cách đổi chỗ lần lượt từng cặp phần tử liền kề cho đến khi mảng được sắp xếp đúng thứ tự. |
| Bubble sort | Thuật toán sắp xếp đơn giản nhất hoạt động bằng cách hoán đổi nhiều lần các phần tử liền kề nếu chúng sai thứ tự. |
| Heap sort | Kỹ thuật sắp xếp dựa trên so sánh dựa trên cấu trúc dữ liệu Binary Heap.  Nó tương tự như sắp xếp lựa chọn, nơi đầu tiên chúng ta tìm phần tử lớn nhất và đặt phần tử lớn nhất ở cuối.  Chúng ta lặp lại quá trình tương tự cho các phần tử còn lại. |
| Quick sort | Một thuật toán sắp xếp, còn được gọi là sắp xếp kiểu phân chia (Part Sort). Là một thuật toán hiệu quả dựa trên việc phân chia mảng dữ liệu thành các nhóm phần tử nhỏ hơn. |
| Merge sort | Một trong những thuật toán có độ phức tạp ở mức trung bình và cùng sử dùng phương pháp chia để trị giống thuật toán sắp xếp nhanh Quick Sort. |

Bài 2. Hãy trình bày ưu điểm và hạn chế của các giải thuật sắp xếp, đề xuất cách tốt nhất để khắc phục hạn chế của các giải thuật này.

|  |  |  |
| --- | --- | --- |
| Selection sort | Dễ cài đặt, dễ hiểu | Độ phức tạp lớn |
| Insection sort | Dễ cài đặt, dễ hiểu | Độ phức tạp lớn |
| Interchange sort | Dễ cài đặt, dễ hiểu | Độ phức tạp lớn |
| Bubble sort | Dễ cài đặt, dễ hiểu | Độ phức tạp lớn |
| Heap sort | Khó cài đặt | Độ phức tạp nhỏ, tối ưu |
| Quick sort | Khó cài đặt | Độ phức tạp nhỏ, tối ưu |
| Merge sort | Khó cài đặt | Độ phức tạp nhỏ, tối ưu |

Đề xuất giải pháp:

- Với những bài toán có giới hạn nhỏ, thì nên dùng những phương pháp Selection sort, Insection sort, Interchange sort, Bubble sort vì nhanh, hiệu quả.

- Với những bài toán có giới hạn lớn, thì nên dùng những phương pháp Heap sort, Quick sort, Merge sort để có thể tối ưu được chương trình chạy một cái hiệu quả.

Bài 3. Giả sử sử dụng hàm Random trong C/C++ để phát sinh ngẫu nhiên dãy số nguyên có kích thước 100, 1.000, 5.000, 10.000, 50.000 số. Hãy cài đặt các giải thuật sắp xếp để sắp dãy số nguyên đã cho theo thứ tự tăng dần và nhận xét về thời gian thực hiện của các giải thuật.

Với dãy số nguyên có kích thước 100, 1000, 5000, ta có thể dùng các giải thuật sắp xếp đơn giản, dễ hiểu, hiệu quả như Selection sort, Insection sort, Interchange sort.

Cài đặt:

void SelectionSort(){

for (int i = 0; i + 1 < n; ++i){

int pos = i;

for (int j = i + 1; j < n; ++j){

if (a[j] < a[pos]){

pos = j;

}

}

swap(a[i], a[pos]);

}

}

void InsertionSort(){

for (int i = 1; i < n; ++i){

int cur = a[i];

int pos = i - 1;

while (pos >= 0 && a[pos] > cur){

a[pos + 1] = a[pos];

pos -= 1;

}

a[pos + 1] = cur;

}

}

void InterchangeSort(){

for (int i = 0; i + 1 < n; ++i){

for (int j = i + 1; j < n; ++j){

if (a[i] > a[j]){

swap(a[i], a[j]);

}

}

}

}

Thời gian thực hiện sẽ là O()

- Đối với dãy số nguyên có kích thước 10.000, 50.000, ta sẽ dùng thuật toán tối ưu hơn như là Heap Sort, Quick Sort …

Cài đặt:

void buildHeap(int a[], int n, int i){

int par = i;

int lf\_node = 2 \* i + 1;

int rf\_node = 2 \* i + 2;

if (lf\_node < n && a[lf\_node] > a[par]){

par = lf\_node;

}

if (rf\_node < n && a[rf\_node] > a[par]){

par = rf\_node;

}

if (par != i){

swap(a[i], a[par]);

buildHeap(a, n, par);

}

}

Độ phức tạp thuật toán: O()

Bài 4: File Bai4.cpp

Bài 5: File Bai5.cpp

Bài 6: File Bai6.cpp

Bài 7: File Bai7.cpp

Bài 8:

Giải thuật chèn trực tiếp là một giải thuật sắp xếp đơn giản nhưng có độ phức tạp lớn trong trường hợp tốt nhất là O(N), trong đó N là số lượng phần tử cần sắp xếp. Tuy nhiên, giải thuật này có thể được cải tiến để đạt được độ phức tạp tốt hơn trong trường hợp tốt nhất là O(NlogN) bằng cách sử dụng thuật toán sắp xếp đệ quy như Merge Sort hoặc Quick Sort.

Ý tưởng cải tiến giải thuật chèn trực tiếp như sau:

- Chia dãy số cần sắp xếp thành các mảng con có kích thước nhỏ hơn hoặc bằng một ngưỡng N0. Trong trường hợp này, ngưỡng N0 có thể được chọn sao cho giải thuật chèn trực tiếp hoạt động tốt trên các mảng có kích thước nhỏ hơn hoặc bằng N0.

- Sử dụng giải thuật chèn trực tiếp để sắp xếp các mảng con có kích thước nhỏ hơn hoặc bằng N0.

- Sử dụng thuật toán sắp xếp đệ quy, chẳng hạn như Merge Sort hoặc Quick Sort, để sắp xếp các mảng con có kích thước lớn hơn N0.

- Trộn các mảng con đã được sắp xếp lại với nhau để tạo ra dãy số đã được sắp xếp.

Khi sử dụng giải thuật cải tiến này, ta sẽ có độ phức tạp trong trường hợp tốt nhất là O(NlogN) vì trong trường hợp này, các mảng con được sắp xếp bởi giải thuật chèn trực tiếp có kích thước nhỏ hơn hoặc bằng N0, và các mảng con có kích thước lớn hơn N0 được sắp xếp bởi thuật toán sắp xếp đệ quy. Tuy nhiên, độ phức tạp trong trường hợp xấu nhất của thuật toán này vẫn là O(N^2) do giải thuật chèn trực tiếp được sử dụng để sắp xếp các mảng con nhỏ.

Bài 9:

Nếu cho một dãy đã có thứ tự thì giải thuật chèn trực tiếp thực hiện sắp xếp nhanh nhất.

Giải thích:

Giải thuật chèn trực tiếp sử dụng phương pháp duyệt từng phần tử của dãy và chèn phần tử đó vào đúng vị trí trong dãy đã được sắp xếp. Với dãy đã có thứ tự, giải thuật sẽ không cần thực hiện việc chèn phần tử nào vào dãy đã sắp xếp, do đó thời gian thực hiện sẽ rất nhanh.

Trong khi đó, giải thuật chọn trực tiếp sẽ luôn thực hiện N^2 lần so sánh giữa các phần tử và hoán đổi chúng để đưa phần tử nhỏ nhất về đầu dãy. Với dãy đã có thứ tự, giải thuật vẫn phải thực hiện các lần so sánh và hoán đổi này, dẫn đến thời gian thực hiện không đổi so với khi sắp xếp một dãy bất kỳ.

Tương tự, giải thuật nổi bọt cũng sẽ thực hiện N^2 lần so sánh giữa các phần tử để đẩy phần tử nhỏ nhất lên đầu dãy. Với dãy đã có thứ tự, giải thuật vẫn phải thực hiện các lần so sánh này, do đó thời gian thực hiện cũng không thay đổi.

Ví dụ minh họa: Giả sử ta có dãy đã được sắp xếp tăng dần [1, 2, 3, 4, 5, 6]. Khi sử dụng giải thuật chèn trực tiếp để sắp xếp lại dãy này, ta chỉ cần duyệt qua từng phần tử một và chèn chúng vào vị trí đúng trong dãy đã sắp xếp. Với dãy này, không có phần tử nào cần được chèn, do đó giải thuật chỉ cần duyệt qua mỗi phần tử một và kết thúc thực hiện trong thời gian O(N). Trong khi đó, giải thuật chọn trực tiếp và nổi bọt vẫn phải thực hiện các lần so sánh và hoán đổi, do đó thời gian thực hiện của hai giải thuật này vẫn là O(N^2) mặc dù dãy đã được sắp xếp từ trước.

Bài 10: File Bai10.cpp

Bài 12: File Bai12.cpp