Nội dung

Các thuật toán sắp xếp nâng cao

* Sắp xếp nhanh
* Sắp xếp trộn

Sắp xếp nhanh – quick sort

* Chia dãy dựa trên chốt - pivot
  + Chốt : đầu, giữa, cuối
  + Random
  + Trung bị 3 phần tử: 3 phần tử đầu, trung vị của (đầu, giữa, cuối)
* Nửa trái <= chốt
* Nửa phải >= chốt
* Sau mỗi lần chia đổi chỗ chốt với phần tử cuối của nửa trái
* Kết thúc mỗi lần lặp **nửa trái | chốt | nửa phải**
* Lần tiếp lại đệ quy để sắp xếp tiếp với nửa trái và nửa phải

Đặc điểm

* Cận thời gian không cố định, phụ thuộc vào việc chia dãy có đều hay không
* Nếu chia đều --> T(n)=O(nlogn)
* Nếu chia lệch --> T(n) = O(n2)
* Để tránh rơi vào trường hợp tồi nhất --> chọn chốt random hoặc trung vị (đủ nhanh và cũng đủ tốt để KHÔNg rơi vào trường hợp tồi nhất)
* Phù hợp với sắp xếp trên mảng (vì phải đổi chỗ các phần tử khá là nhiều)

Visualization : <https://visualgo.net/en/sorting>

|  |
| --- |
| #include <stdio.h>    /\*  // Cho C++  void swap(int& a, int &b)  {  int c = a;  a=b;  b=c;  }  \*/  // cho C  // hoan doi gia tri 2 phan tu o vi tri i va j trong mang A  void swap(int \*A, int i, int j)  {  int c = A[i];  A[i] = A[j];  A[j] = c;  }      int partition(int A[], int L, int R, int indexPivot) {  int pivot = A[indexPivot];  // doi cho phan tu chot ve cuoi nua phai  //swap(A[indexPivot], A[R]);  swap(A,indexPivot, R);  int storeIndex = L; // vi tri cuoi cung cua nua trai  for(int i = L; i <= R-1; i++){  if(A[i] < pivot){  //swap(A[storeIndex], A[i]);  swap(A, storeIndex, i);  storeIndex++;  }  }  // doi phan tu chot ve cuoi cua nua trai  //swap(A[storeIndex], A[R]);  swap(A,storeIndex, R);  return storeIndex;  }    // chot la phan tu giua  void quickSort(int A[], int L, int R) {  if(L < R){  int index = (L + R)/2;  index = partition(A, L, R, index);  if(L < index) // check xem co phan tu de goi de quy  quickSort(A, L, index-1);  if(index < R)  quickSort(A, index+1, R);  }  }    void printArr(int \*A, int size)  {  printf("Day hien tai:\n");  for(int i=0; i<size; i++)  printf("%d, ",A[i]);  printf("\n");  }  int main()  {  int A[] = {1,4,6,78,23,4,5,6,2,5,6,8,5,3,43,2,34,23,4,5,56,78};  int size = 22;    printArr(A,size);  quickSort(A,0, size-1);  printArr(A,size);  return 0;  } |

**Bài tập 1**. cho 1 file chứa các số thực có định dạng như sau

|  |
| --- |
| 5  12.5  6  6  25.7  6 |

Với dòng đầu tiên là số lượng phần tử

Các dòng tiếp theo là giá trị các phần tử

Hãy xây dựng chương trình có sử dụng **sắp xếp nhanh** để lọc các phần tử trùng khỏi file và ghi các phần tử không trùng sang file mới theo format tương tự file ban đầu (các phần tử được sắp theo thứ tự tăng dần)

|  |
| --- |
| 3  6  12.5  25.7 |

**Bài tập 2**. Cho 1 file văn bản chứa các từ tiếng anh (tương tự bài về nhà số 5)

Hãy thống kê và in ra danh sách các từ **theo thứ tự tần số giảm dần**

|  |
| --- |
| Fernando Espinoza, a 29-year-old American teacher and former US Navy submariner, disappeared in Libya on November 9, five weeks after arriving in the country to start a new job at an international school in Tripoli.  He'd ventured south of the city for a weekend trip to a desert oasis, but on his return was picked up for questioning. And the frequent texts he sent to his mom ceased. |

VD với đầu vào

|  |
| --- |
| a b c a a |

Thì in ra

a : 3

b: 1

c: 1

Mảng các từ có thể dùng

* Vector trong C++
* Char words[30][500]
* So sánh các từ dùng hàm strcmp()

**Bài tập 3**. cho đầu vào là 2 file chứa các số nguyên theo định dạng sau

|  |
| --- |
| 5  1  5  7  4  6 |

Dòng đầu là số lượng số trong file

Các dòng tiếp theo lần lượt là giá trị các số

Hãy xây dựng chương trình để gộp 2 file này thành 1 file mới sao cho các phần tử không trùng nhau và giá trị các số được sắp theo thứ tự tăng dần.

=======

Thuật toán sắp xếp trộn

* Trộn 2 dãy đã sắp xếp để được dãy mới cũng theo đúng thứ tự
* Gồm 2 bước là bước chia và bước trộn
  + Bước chia: chia đôi dãy nếu dãy >=2 phần tử
  + Bước trộn: Trộn 2 dãy đã có thứ tự
* Thời gian gian cỡ O(nlogn)
* Có thể áp dụng trên **danh sách liên kết** hoặc mảng
* Nếu áp dụng trên mảng phải dùng thêm 1 mảng phụ khi trộn để tránh phải dịch phần tử ==> không hiệu quả so với các thuật toán sắp xếp khác có trên mảng
* Trên mảng có thể dùng sắp xếp nhanh hoặc sắp xếp vun đống thay thế

|  |
| --- |
| void merge(int A[], int L, int M, int R) {  // tron 2 day da sap A[L..M] va A[M+1..R]  int i = L; int j = M+1;  int \*TA = (int\*)calloc(sizeof(int),R+1);  for(int k = L; k <= R; k++){  if(i > M){ TA[k] = A[j]; j++;}  else if(j > R){TA[k] = A[i]; i++;}  else{  if(A[i] < A[j]){  TA[k] = A[i]; i++;  }  else {  TA[k] = A[j]; j++;  }  }  }  for(int k = L; k <= R; k++) A[k] = TA[k];  free(TA);  }    void mergeSort(int A[], int L, int R) {  if(L < R){  int M = (L+R)/2;  mergeSort(A,L,M);  mergeSort(A,M+1,R);  merge(A,L,M,R);  }  } |

**Dùng với danh sách liên kết**

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  #include <string.h>  typedef struct node  {  char hoten[30];  float diem;  struct node \*pNext;  } NODE;    NODE\* insert(NODE \*pHead, NODE\* ptr)  {  if(pHead==NULL) pHead=ptr;  else // tìm vi tri chen vao giua/cuoi danh sach  {  if(pHead->diem >= ptr->diem) //phan tu moi them dung dau danh sach cu  {  ptr->pNext=pHead;  pHead=ptr; // cap nhat lai dau  }  else  {  NODE \*preQ=pHead; // con tro toi phan tu ngay truoc vi tri chen  NODE \*q=pHead->pNext; // la con tro toi vi tri chen  while(q!=NULL && q->diem < ptr->diem)  {  preQ=q;  q=q->pNext;  }  ptr->pNext=q; // chen ptr vao sau preQ  preQ->pNext=ptr;  }  }  return pHead;  }    NODE \*createNode(char\* hoten, float diem)  {  NODE \*newNode= (NODE\*)malloc(sizeof(NODE));  strcpy(newNode->hoten,hoten);  newNode->diem = diem;  newNode->pNext = NULL;  return newNode;  }      void clearBuffer()  {  int c;  while ((c = getchar()) != '\n' && c != EOF) { }  }  // tao danh sach nhap tu ban phim    NODE\* createList()  {  int size;  printf("So luong phan tu: ");  scanf("%d",&size);  char hoten[30];  float diem;  NODE \*head=NULL, \*newNode;  for(int i=0; i<size;i++)  {  printf("Hoten: ");  clearBuffer();  //gets(hoten);  if (fgets(hoten, sizeof hoten, stdin)) {  hoten[strcspn(hoten, "\n")] = '\0';  }  printf("Diem: ");  scanf("%f",&diem);  if(head==NULL) head = createNode(hoten,diem);  else {  newNode = createNode(hoten,diem);  newNode->pNext = head;  head = newNode;  }  }  return head;  }    // in ra danh sach  void printList(NODE \*head)  {  printf("Noi dung danh sach hien tai:\n");  while(NULL!=head)  {  printf("Ho ten: %s\t", head->hoten);  printf("Diem: %.2f\n", head->diem);  head = head->pNext;  }  printf("\n");  }    void split(NODE \*&first, NODE \*&second)  {  NODE \*start=first;  NODE \*end=first->pNext;  while(end->pNext!=NULL && end->pNext->pNext!=NULL)  {  start = start->pNext;  end = end->pNext->pNext;  }  second = start->pNext;  start->pNext=NULL;  }    void merge(NODE \*first, NODE \*second, NODE \*&third)  {  NODE \*ptr;  third=(NODE\*)malloc(sizeof(NODE));  third->pNext=NULL;  ptr=third;    NODE \*p=first;  NODE \*q=second;  while(p!=NULL && q!=NULL)  {  if(p->diem <= q->diem)  {  ptr->pNext=p;  p=p->pNext;  ptr=ptr->pNext;  ptr->pNext=NULL;  }  else  {  ptr->pNext=q;  q=q->pNext;  ptr=ptr->pNext;  ptr->pNext=NULL;  }  }  if(p==NULL) ptr->pNext=q;  else ptr->pNext=p;  ptr=third;  third=third->pNext;  free(ptr);  }  void MergeSort(NODE \*&pHead)  {  if(pHead!=NULL && pHead->pNext!=NULL)  {  NODE \*second=NULL;  split(pHead,second);  MergeSort(pHead);  MergeSort(second);  NODE \*third=NULL;  merge(pHead,second,third);  pHead=third;  }  }    int main()  {  NODE\* head=NULL;  head = createList();  printf("Danh sach ban dau:\n");  printList(head);  //head = insertionSort(head);  MergeSort(head);  printf("Danh sach sau khi sap xep:\n");  printList(head);  return 0;  } |

**Bài tập 4**. Cho file chứa thông tin của sinh viên theo dạng

|  |
| --- |
| Nguyen Van A  20191234  091332313  Tran Van Cuong  20191434  091332313  Le Anh Dung  20181234  091332313  # |

Các thông tin gồm

* Họ và tên
* SHSV
* SDT

Kết thúc file là dấu #

Hãy viết chương trình đọc file trên và in ra file sắp xếp tùy theo lựa chọn

* Sắp xếp theo họ và tên
* Sắp xếp theo SHSV
* Sắp xếp theo chỉ TÊN (có phần tách tên khỏi họ và tên)

Dùng danh sách liên kết và thuật toán sắp xếp trộn để làm