Nội dung

* Các thuật toán tìm kiếm cơ bản
  + Tìm kiếm tuần tự
  + Tìm kiếm nhị phân

Thuật toán tìm kiếm tuần tự:

* So sánh tuần tự giá trị khóa cần tìm với giá trị khóa của các phần tử trong danh sách
* Thời gian cỡ O(n)
* Chỉ phù hợp với danh sách ít phần tử
* Có thể thực hiện trên cả mảng lẫn danh sách liên kết
* Danh sách ban đầu KHÔNG cần sắp theo thứ tự

Ví dụ. Cho 1 dãy số nguyên gồm n phần tử (không trùng lặp) hãy tìm xem có tồn tại 2 phần tử a và b sao cho a+b=k hay không.

Thử nghiệm với n =100, 10000, và n = 1000000

Làm thế nào để sinh ra các số nguyên không trùng lặp?

Sinh số, sau đó check xem số đó đã tồn tại hay chưa,

* Nếu chưa tồn tại --> kết nạp vào danh sách
* Nếu đã tồn tại --> bỏ qua

Tìm kiếm mỗi khi sinh thêm 1 phần tử mới

Tim xem trong day co tồn tại a= k-b hay không (a se co thể là số A[0] tới A[n])

--> dùng 2 vòng lặp lồng nhau --> O(n2)

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  #include <time.h>  const int MAX\_VAL = 200000;  // sinh ra mang size phan tu KHONG trung nhau  // dung tim kiem tuan tu de check trung ==> chi phu hop voi size nho  int \* generateArr(int size)  {  int x, check;  srand(time(NULL));  int \*A = (int\*)calloc(sizeof(int),size);  // sinh ngau nhien va check xem da co phan tu hay chua  A[0]=rand()%MAX\_VAL;  int numEle = 1;  while(numEle<size)  {  x = rand()%MAX\_VAL;  // check bang tim kiem tuan tu  check =0;  for(int i=0; i<numEle; i++)  if(A[i]==x) {  check=1;  break;  }  if(check==0) // neu khong bi trung --> them vao  {  A[numEle] = x;  numEle++;  }  }  return A;  }  // tim xem trong day co ton tai 2 sô a+b=b  // thuạt toan dung tim kiem tuan tu  int checkExist(const int \*A, int size, int k, int \*a, int \*b)  {  int check =0;  for(int i=2; i<size; i++)  for(int j=0;j<i;j++)  if(A[i]+A[j]==k) {  \*a = A[i];  \*b = A[j];  return 1;  }  return 0;  }  void printArr(const int \*A, int size)  {  printf("Mang hien tai:\n");  for(int i=0;i<size;i++) printf("%d, ",A[i]);  printf("\n");  }    int main()  {  int \*A = NULL;  int size=100000, k=9999,a,b;  if(size>=MAX\_VAL)  {  printf("Check lai tham so!");  return -1;  }  printf("Sinh ra cac so khong lap!\n");  A = generateArr(size);    //printArr(A,size);  printf("Tim kiem so thoa man!\n");  int check =checkExist(A,size,k,&a, &b);  if(check==1) printf("Ton tai 2 so tong la %d, do la %d va %d\n",k,a,b);  else printf("KHONG ton tai 2 so tong la %d!\n",k);    return 0;  } |

Bài 1. đếm và in ra tất cả các cặp số a,b thỏa mãn a+b = k với k được nhập vào từ bàn phím và k bắt buộc phải thỏa mãn k<2\*MAX\_VAL.

================================================

Thuật toán tìm kiếm nhị phân:

* Danh sách ban đầu cần có thứ tự
* Dùng mảng --> dịch phần tử khi thêm
* Tìm kiếm bằng cách so sánh trực tiếp với phần tử ở giữa danh sách
* Tùy vào mối quan hệ giữa khóa và phần tử ở giữa để đi tiếp theo nửa trái hoặc nửa phải
* Thời gian O(logn)
* Chỉ thể hiện sự khác biệt rõ ràng so với tìm kiếm tuần tự khi số lượng phần tử trên danh sách lớn (n>10000)

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  #include <time.h>  const int MAX\_VAL = 200000;  // sinh ra mang size phan tu KHONG trung nhau  // dung tim kiem tuan tu de check trung ==> chi phu hop voi size nho  int \* generateArr(int size)  {  int x, check;  srand(time(NULL));  int \*A = (int\*)calloc(sizeof(int),size);  // sinh ngau nhien va check xem da co phan tu hay chua  A[0]=rand()%MAX\_VAL;  int numEle = 1;  while(numEle<size)  {  x = rand()%MAX\_VAL;  // check bang tim kiem tuan tu  check =0;  for(int i=0; i<numEle; i++)  if(A[i]==x) {  check=1;  break;  }  if(check==0) // neu khong bi trung --> them vao  {  A[numEle] = x;  numEle++;  }  }  return A;  }    // tim kiem nhio phan tren mang A  // vi tri bat dau la s va vi tri ket thuc la e  // khoa can tim la k    int binsearch(const int \*A, int s, int e, int key)  {  int mid;  while(s<=e)  {  mid = (s+e)/2;  if(A[mid]==key) return mid;  if(A[mid]>key) e = mid-1;  else // A[mid]<key  s = mid+1;  }  return -1;  }    // chen phan tu x vao day da co thu tu  void insertValue(int \*A, int currSize, int x)  {  // tim vi tri chen  int pos = 0;  while(A[pos]<x && pos<currSize) pos++;    // phai dich phan tu  for(int i=currSize-1; i>=pos; i--)  A[i+1] = A[i];    A[pos] = x;    }  // sinh ra mang size phan tu KHONG trung nhau  // dung tim kiem tuan tu de check trung ==> chi phu hop voi size nho  int \* generateArr\_binsearch(int size)  {  int x;  srand(time(NULL));  int \*A = (int\*)calloc(sizeof(int),size);  // sinh ngau nhien va check xem da co phan tu hay chua  A[0]=rand()%MAX\_VAL;  int numEle = 1;  while(numEle<size)  {  x = rand()%MAX\_VAL;  // check bang tim kiem nhi phan  if(binsearch(A,0,numEle-1,x)<0){  // them x vao day sao cho van dung thu tu  insertValue(A,numEle,x);  numEle++;  }  }  return A;  }  // tim xem trong day co ton tai 2 sô a+b=b  // thuạt toan dung tim kiem tuan tu  int checkExist(const int \*A, int size, int k, int \*a, int \*b)  {  int check =0;  for(int i=2; i<size; i++)  for(int j=0;j<i;j++)  if(A[i]+A[j]==k) {  \*a = A[i];  \*b = A[j];  return 1;  }  return 0;  }    // tim xem trong day co ton tai 2 sô a+b=b  // thuạt toan dung tim kiem nhi phan  int checkExist\_binSearch(const int \*A, int size, int k, int \*a, int \*b)  {  int check =0;  for(int i=0; i<size; i++)  if(binsearch(A,0,size-1,k-A[i])>0)  {  \*a = A[i];  \*b = k-A[i];  return 1;  }  return 0;  }  void printArr(const int \*A, int size)  {  printf("Mang hien tai:\n");  for(int i=0;i<size;i++) printf("%d, ",A[i]);  printf("\n");  }    int main()  {  int \*A = NULL;  int size=100000, k=100000,a,b;  if(size>=MAX\_VAL)  {  printf("Check lai tham so!");  return -1;  }    printf("Sinh ra cac so khong lap!\n");  A = generateArr\_binsearch(size);  //A = generateArr(size);  printf("done!\n");  //printArr(A,size);      printf("Tim kiem so thoa man!\n");  int check =checkExist(A,size,k,&a, &b);  if(check==1) printf("Ton tai 2 so tong la %d, do la %d va %d\n",k,a,b);  else printf("KHONG ton tai 2 so tong la %d!\n",k);    printf("========================================\n");  printf("Tim kiem so thoa man dung binsearch!\n");  check =checkExist\_binSearch(A,size,k,&a, &b);  if(check==1) printf("Ton tai 2 so tong la %d, do la %d va %d\n",k,a,b);  else printf("KHONG ton tai 2 so tong la %d!\n",k);  return 0;  } |

**Bài tập 2**. Giả sử bạn cần quản lý thông tin liên lạc của các thành viên trong lớp. Thông tin của mỗi sinh viên gồm

* Họ tên: xâu ký tự độ dài tối đa 30
* Số hiệu sinh viên: xâu ký tự độ dài 8 (KHÔNG trùng nhau)
* Số điện thoại: xâu ký tự độ dài 15

Thông tin này sẽ được lưu trữ trong file văn bảng theo format dạng

|  |
| --- |
| Nguyen Van A  20201234  0912345678  Nguyen Thi B  20201234  0912345678  $ |

Dấu $ báo hiệu kết thúc file

Hãy xây dựng chương trình

* Đọc vào nội dung danh sách sinh viên
* Tổ chức tìm kiếm thông tin sinh viên dựa trên số hiệu sinh viên (tìm kiếm nhị phân)
* Nếu trong trường hợp tên trùng nhau thì có thể tổ chức tìm kiếm nhị phân để tìm và trả về tất cả các sinh viên có tên cần tìm nhập từ bàn phím hay không?
* Xây dựng thêm chức năng thêm thông tin của 1 sinh viên mới
* Ghi danh sách sinh viên hiện tại ra file

Chú ý. Danh sách sinh viên cần được lưu trữ bằng mảng để tìm kiếm nhị phân (sắp thứ tự theo SHSV)

**Bài tập 3**. cho 1 file số nguyên gồm n số với các giá trị số nằm trong khoảng 1-M. Hãy xây dựng chương trình đọc vào file số này, tìm và in ra màn hình các số bị thiếu (trong khoảng 1-M) trong file một cách nhanh nhất

VD.  
File đầu vào với M = 15, n =12

|  |
| --- |
| 15 12 2 3 5 6 8 12 4 5 7 9 14 12 |

Dòng đầu sẽ là giá trị M, n và dòng tiếp theo sẽ là giá trị n số

Cần in ra các số bị thiếu

VD. 1 10 11 13 15