Nội dung

* Thuật toán tìm kiếm tuần tự và tìm kiếm nhị phân

**Bài 1**. Tính thời gian thực hiện m thao tác tìm kiếm liên tiếp bằng tìm kiếm tuần tự và tìm kiếm nhị phân

Cho danh sách ban đầu gồm n phần tử (n=1000, 100000, và 1tr)

Thực hiện tìm kiếm liên tiếp m lần (m=10, 100) với khóa k tìm kiếm được sinh ngẫu nhiên

Tính và in ra thời gian tìm kiếm bằng 2 cách

* Tuần tự
* Nhị phân (yêu cầu danh sách phải được sắp thứ tự)

Minh họa với tìm kiếm trên mảng

Vấn đề cần xử lý

* Sinh ngẫu nhiên dùng hàm random sẽ chỉ sinh được các gía trị tử 0-RAND\_MAX (cỡ 32K) --> làm thế nào sinh được 1tr số?
  + Dùng nhiều lần random liên tiếp và cộng kết quả lại (VD. Cộng 20 lần random lại) --> lâu
  + Dung 2 lần sinh liên tiếp: lần 1 \* 1000 + lần 2

|  |
| --- |
| int getNextRandNumber()  {  int a = rand();  int b = rand();  return a\*1000 + b;  } |

* Sinh các phần tử KHÔNG được trùng nhau?
  + Với mỗi phần tử đã sinh, kiểm tra trong mảng xem có trùng với các phần tử trước hay không --> nếu trùng --> sinh lại --> thời gian sinh sẽ cỡ O(n^2)

|  |
| --- |
| // ham sinh ngau nhien cac phan tu cua mang  // cac phan tu cua mang khong trung nhau  int \*generateRandomArray(int size)  {  // khai bao mang dong  int \*arr = (int\*) malloc(size\*sizeof(int));  int n=0, nextNum, check, i;  while(n<size)  {  nextNum = getNextRandNumber();  // kiem tra xem da bi trung? O(n^2)  check = 0;  for(i=0;i<n;i++)  if(arr[i]==nextNum)  {  check =1;  break;  }  if(check==0) // so khong bi trung  {  arr[n]=nextNum;  n++;  }  }  return arr;  } |

* Tinh thoi gian thuc hien

|  |
| --- |
| struct timeval {  long tv\_sec; /\* seconds \*/  long tv\_usec; /\* microseconds \*/  };  Nằm trong thư viện #include <sys/time.h>  Để tính thời gian thực hiện ta cần 2 mốc thời gian  Thời gian bắt đầu và thời gian kết thúc --> lấy thời gian kết thúc trừ thời gian bắt đầu  void checkTimeSpend()  {  struct timeval tval\_before, tval\_after, tval\_result;  **gettimeofday(&tval\_before, NULL);**  // Some code you want to time, for example:  delay(1); --> code cần tính thời gian nào đó  **gettimeofday(&tval\_after, NULL);**  tval\_result.tv\_sec = tval\_after.tv\_sec - tval\_before.tv\_sec;  tval\_result.tv\_usec = tval\_after.tv\_usec - tval\_before.tv\_usec;  printf("Time elapsed: %ld.%06ld\n", (long int)tval\_result.tv\_sec, (long int)tval\_result.tv\_usec);  } |

CODE

|  |
| --- |
| #include<stdio.h>  #include<time.h>  #include<stdlib.h>  #include <stdint.h>  #include <sys/time.h>    // sinh ngau nhien so co gia tri toi co 1tr  int getNextRandNumber()  {  int a = rand();  int b = rand();  return a\*1000 + b;  }    struct timeval timeSub(struct timeval s, struct timeval e)  {  struct timeval tval\_result;  tval\_result.tv\_sec = e.tv\_sec - s.tv\_sec;  tval\_result.tv\_usec = e.tv\_usec - s.tv\_usec;  if(tval\_result.tv\_usec<0) {  tval\_result.tv\_usec = 1000000+tval\_result.tv\_usec;  tval\_result.tv\_sec--;  }  return tval\_result;  }  // ham sinh ngau nhien cac phan tu cua mang  // cac phan tu cua mang khong trung nhau  int \*generateRandomArray(int size)  {  struct timeval tval\_before, tval\_after, tval\_result;  gettimeofday(&tval\_before, NULL);  // khai bao mang dong  int \*arr = (int\*) malloc(size\*sizeof(int));  int n=0, nextNum, check, i;  while(n<size)  {  nextNum = getNextRandNumber();  // kiem tra xem da bi trung? O(n^2)  check = 0;  for(i=0;i<n;i++)  if(arr[i]==nextNum)  {  check =1;  break;  }  if(check==0) // so khong bi trung  {  arr[n]=nextNum;  n++;  }  }  gettimeofday(&tval\_after, NULL);  tval\_result = timeSub(tval\_before, tval\_after);  printf("thoi gian sinh mang : %ld.%06ld\n", (long int)tval\_result.tv\_sec, (long int)tval\_result.tv\_usec);  return arr;  }  // ham sap xep mang  void sortArray(int \*arr, int size)  {  }    // in ra mang hien tai  void printArr(int \*arr, int size)  {  printf("Danh sach hien tai: \n");  for(int i=0; i<size; i++) {  printf("%9d, ", arr[i]);  if((i+1)%10==0) printf("\n");  }  printf("\n");  }  void delay(int number\_of\_seconds)  {  // Converting time into milli\_seconds  int milli\_seconds = 1000 \* number\_of\_seconds;    // Storing start time  clock\_t start\_time = clock();    // looping till required time is not achieved  while (clock() < start\_time + milli\_seconds);  }    // ham minh hoa qua trinh tim kiem tuan tu  void sequentialSearch(int \*arr, int size)  {  int searchKey = getNextRandNumber();  for(int i=0; i<size; i++)  if(arr[i]==searchKey)  {  printf("Found key %d at %d\n",searchKey, i);  return;  }  printf("Could not find key %d\n", searchKey);  }    void checkTimeSpend(int \*arr, int size, int repeatTimes)  {  struct timeval tval\_before, tval\_after, tval\_result;  gettimeofday(&tval\_before, NULL);  // Some code you want to time, for example:  for(int i=0; i<repeatTimes; i++)  sequentialSearch(arr,size);  //delay(1);  gettimeofday(&tval\_after, NULL);  tval\_result = timeSub(tval\_before, tval\_after);  printf("Time elapsed: %ld.%06ld\n", (long int)tval\_result.tv\_sec, (long int)tval\_result.tv\_usec);  }  int main()  {  int \*arr, size = 50000, repeatTimes = 100;  arr =generateRandomArray(size);  //printArr(arr,size);  checkTimeSpend(arr, size, repeatTimes);  return 0;  } |

**Bài tập 1**:

* Thực hiện sắp xếp mảng dùng thuật toán bất kỳ - có in ra thời gian sắp xếp
* Thực hiện m lần tìm kiếm nhị phân và in ra tổng thời gian - để so với thời gian tìm kiếm tuần tự