ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN



BÁO CÁO ĐỒ ÁN GIỮA KÌ

Môn: Cấu trúc dữ liệu và giải thuật

Họ và tên: Âu Dương Gia Tuấn

MSSV: 1712867

MỤC LỤC

1. Giới thiệu chung

Về việc các *thuật toán sắp xếp* (*Sorting Algorithm*), ta có rất nhiều loại thuật toán khác nhau, ví dụ như: selection sort, insertion sort, interchange sort, binary interchange sort, radix sort, merge sort, heap sort, flash sort, counting sort, shell sort, bubble sort, shaker sort, quick sort, bucket sort, cocktail sort,….

Mỗi thuật toán trên đều có ưu và nhược điểm khác nhau, tùy thuộc vào từng bộ dữ liệu mà ta chọn mộ thuật toán thích hợp nhất để cài đặt.

Nhưng trong bài báo cáo này ta chỉ tìm hiểu về 10 thuật toán khá phổ biến hiện nay, đó là: selection sort, interchange sort, insertion sort, binary insertion sort, shaker sort, quick sort, heap sort, shell sort, merge sort, radix sort.

1. Tìm hiểu về các thuật toán

# Selection sort

## Ý tưởng

Ý tưởng của thuật toán Selection sort chia phần mảng ban đầu thành hai phần: phần mảng đã được sắp xếp và phần mảng chưa được sắp xếp.

Sau đó tiến hành sắp xếp mảng bằng cách liên tục tìm ra *phần tử nhỏ nhất (ví dụ sắp cho xếp tăng dần)* từ phần mảng chưa được sắp xếp và hoán đổi vị trí đó với vị trí đầu. Tiến trình này sẽ dừng lại khi toàn bộ phần tử của mảng chưa được sắp xếp đều được chuyển sang phần mảng đã được sắp xếp.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 10 | 4 | 7 | 3 | 15 | 1 |

Ví dụ:

**Mảng ban đầu:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 10 | 4 | 7 | 3 | 15 | 1 |

Vị trí đầu tiên chính là số 10 và sau khi duyệt mảng thì ta có số 1 là giá trị nhỏ nhất:

***i*=0**

Ta tiến hành hoán đổi số 10 và số 1:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 4 | 7 | 3 | 15 | 10 |

Lúc này mảng đã được chia thành hai phần: phần đã sắp xếp và phần chưa sắp xếp

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 4 | 7 | 3 | 15 | 10 |

Tiến trình trên sẽ áp dụng cho phần còn lại của mảng.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 4 | 7 | 3 | 15 | 10 |

***i*=1**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 3 | 7 | 4 | 15 | 10 |

***i*=2**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 3 | 4 | 7 | 15 | 10 |

***i*=3**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 3 | 4 | 7 | 15 | 10 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 3 | 4 | 7 | 10 | 15 |

***i*=4**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 3 | 4 | 7 | 10 | 15 |

**Mảng đã sắp xếp:**

## Thuật toán

|  |
| --- |
| void hoanVi(int &a, int &b) |
| {  int temp;  temp = a;  a = b;  b = temp;  } |

|  |
| --- |
| void selectionSort(int\* arr, int soLuongPhanTu) |
| {  int min;  for (int i = 0; i < soLuongPhanTu - 1; i++)  {  min = i;  for (int j = i + 1; j < soLuongPhanTu; j++)  {  if (arr[min] > arr[j])  {  min = j;  }  }  if (i != min)  {  hoanVi(arr[i], arr[min]);  }  }  } |

## Đánh giá

Độ phức tạp của thuật toàn là O(

|  |  |
| --- | --- |
| **Ưu điểm** | **Nhược điểm** |
| Thuật toán dễ cài đặt.  Sắp xếp nhanh với mảng ít phần tử. | Tốn nhiều bộ nhớ.  Đối với mảng nhiều phần tử, thuật toán sắp xếp lâu. |

# Interchange sort



## Ý tưởng

Ý tưởng của thuật toán Interchange sort là bắt cặp tất cả phần tử trong mảng và đổi chổ cho nhau nếu hai phần tử đó thỏa điều kiện sắp xếp.

Thuật toán này sẽ dùng hai vòng lặp. Vòng lặp ngoài sẽ chạy từ *vị trí đầu tiên* đến vị trí thứ *n-1* của mảng. Vòng lặp trong sẽ chạy từ *vị trí* *ta đang xét + 1* đến vị trí *n* của mảng. Mỗi lần ta sẽ so sánh hai giá trị trong vòng lặp, nếu chúng thỏa điều kiện, sẽ tiến hành hoán đổi vị trí.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 10 | 4 | 7 | 3 | 15 | 1 |

**Ví dụ:**

**Mảng ban đầu:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 10 | 4 | 7 | 3 | 15 | 1 |

Ta sẽ tiến hành sắp xếp mảng trên tăng dần.

***i*=0**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 4 | 10 | 7 | 3 | 15 | 1 |

Vì số 10 và 4 thỏa điều kiện nên ta sẽ hoán đổi vị trí hai số này.

Ta sẽ tiến hành quá trinh này cho đến khi chạy hết vòng lặp.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 4 | 10 | 7 | 3 | 15 | 1 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 4 | 10 | 7 | 3 | 15 | 1 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 3 | 10 | 7 | 4 | 15 | 1 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 3 | 10 | 7 | 4 | 15 | 1 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 3 | 10 | 7 | 4 | 15 | 1 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 10 | 7 | 4 | 15 | 3 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 10 | 7 | 4 | 15 | 3 |

***i*=1**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 7 | 10 | 4 | 15 | 3 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 10 | 7 | 4 | 15 | 3 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 4 | 7 | 10 | 15 | 3 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 4 | 7 | 10 | 15 | 3 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 4 | 7 | 10 | 15 | 3 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 3 | 7 | 10 | 15 | 4 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 3 | 7 | 10 | 15 | 4 |

***i*=2**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 3 | 7 | 10 | 15 | 4 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 3 | 7 | 10 | 15 | 4 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 3 | 4 | 10 | 15 | 7 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 3 | 4 | 10 | 15 | 7 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 3 | 4 | 10 | 15 | 7 |

***i*=3**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 3 | 4 | 7 | 15 | 10 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 3 | 4 | 7 | 15 | 10 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 3 | 4 | 7 | 10 | 15 |

***i*=4**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 3 | 4 | 7 | 10 | 15 |

**Mảng đã sắp xếp:**

## Thuật toán

|  |
| --- |
| void interchangeSort(int\* arr, int soLuongPhanTu) |
| {  for (int i = 0; i < soLuongPhanTu - 1; i++)  {  for (int j = i + 1; j < soLuongPhanTu; j++)  {  if (arr[i] > arr[j])  {  hoanVi(arr[i], arr[j]);  }  }  }  } |

## Đánh giá

Độ phức tạp của thuật toàn là O(.

|  |  |
| --- | --- |
| **Ưu điểm** | **Nhược điểm** |
| Thuật toán dễ cài đặt.  Sắp xếp nhanh với mảng ít phần tử. | Tốn nhiều bộ nhớ.  Đối với mảng nhiều phần tử, thuật toán sắp xếp lâu. |

# Insertion sort

## Ý tưởng

Ý tưởng của thuật toán Insertion sort giống như việc chơi bài, khi trên tay bạn đang cầm một dãy bài đã theo thứ tự và người chơi phát cho bạn thêm một quân bài, bạn sẽ so sánh quân bài mới đó với từng quân bài trên tay bạn cho đến khi bạn tìm được vị trí thích hợp để chèn vào.

Với thuật toán Insertion sort, ta sẽ bắt đầu vòng lặp từ vị trí thứ 1 đến hết mảng, trong vòng lặp ta sẽ tìm ra phần tử cần chèn rồi di chuyển phần tử đó đến vị trí thích hợp rồi chèn vào vị trí đó.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 10 | 4 | 7 | 3 | 15 | 1 |

**Ví dụ:**

**Mảng ban đầu:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 10 | 4 | 7 | 3 | 15 | 1 |

***i*=1**

Ta sẽ bắt đầu vòng lặp từ vị trí số 4, ta thấy số 4 không theo thứ tự với vị trí đứng trước nó nên ta sẽ di chuyển số 4 và chèn vào trước số 10

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 4 | 10 | 7 | 3 | 15 | 1 |

Cứ như vậy thực hiện hết vòng lặp

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 4 | 10 | 7 | 3 | 15 | 1 |

***i*=2**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 4 | 7 | 10 | 3 | 15 | 1 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 4 | 7 | 10 | 3 | 15 | 1 |

**i=3**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 3 | 4 | 7 | 10 | 15 | 1 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 3 | 4 | 7 | 10 | 15 | 1 |

***i*=4**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 3 | 4 | 7 | 10 | 15 | 1 |

***i*=5**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 3 | 4 | 7 | 10 | 15 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 3 | 4 | 7 | 10 | 15 |

**Mảng sau sắp xếp:**

## Thuật toán

|  |
| --- |
| void insertionSort(int\* arr, int soLuongPhanTu) |
| {  int x, j;  for (int i = 1; i < soLuongPhanTu; i++)  {  x = arr[i];  j = i - 1;  while ((j >= 0) && (arr[j] > x))  {  arr[j + 1] = arr[j];  j--;  }  arr[j + 1] = x;  }  } |

## Đánh giá

Độ phức tạp của thuật toán là O(n)

|  |  |
| --- | --- |
| **Ưu điểm** | **Nhược điểm** |
| Thuật toán dễ cài đặt.  Sắp xếp nhanh với mảng ít phần tử hoặc mảng đã được sắp xếp một phần. | Đối với mảng nhiều phần tử, thuật toán sắp xếp khá lâu. |

# Binary Insertion sort

## Ý tưởng

Ý tưởng của thuật toán Binary Insertion sort giống như thuật toán Insertion sort nhưng trong Binary Insertion sort dùng thuật toán tìm kiếm Binary để tìm kiếm vị trí cần chèn.

## Thuật toán

|  |
| --- |
| int timKiemNhiPhan(int\* arr, int x, int left, int right) |
| {  if (right <= left)  {  return (x > arr[left]) ? (left + 1) : left;  }  int mid = (left + right) / 2;  if (arr[mid] == x)  {  return mid + 1;  }  else if (arr[mid] > x)  {  return timKiemNhiPhan(arr, x, left, mid - 1);  }  else  {  return timKiemNhiPhan(arr, x, mid + 1, right);  }  } |

|  |
| --- |
| void binaryInsertionSort(int\* arr, int soLuongPhanTu) |
| {  int x, j, y;  for (int i = 1; i < soLuongPhanTu; i++)  {  x = arr[i];  j = i - 1;  y = timKiemNhiPhan(arr, x, 0, j);  while (j >= y)  {  arr[j + 1] = arr[j];  j--;  }  arr[j + 1] = x;  }  } |

## Đánh giá

Độ phức tạp của thuật toán: O(log(n))

|  |  |
| --- | --- |
| **Ưu điểm** | **Nhược điểm** |
| Sắp xếp nhanh với mảng ít phần tử hoặc mảng đã được sắp xếp một phần. | Cài đặt thuật toán khá khó.  Đối với mảng nhiều phần tử, thuật toán sắp xếp khá lâu. |

# Shaker sort

## Ý tưởng

Ý tưởng thuật toán Shaker sort là xét ba vòng lặp,vòng lặp thứ nhất ta xét trong khi left<=right, vòng lặp thứ hai bắt đầu từ cuối dãy, vòng lặp thứ ba bắt đầu từ đầu dãy. Trong vòng lặp thứ hai, ta xét hai phần tử đứng liền nhau nếu chúng thỏa điều kiện, thì hoán đổi vị trí, kết thúc vòng lập thứ hai thì phần tử nhỏ nhất sẽ đứng đầu mảng. Trong vòng lặp thứ hai, ta cũng xét hai phần tử đứng liền nhau nếu chúng thỏa điều kiện, thì hoán đổi vị trí, kết thúc vòng lập thứ ba thì phần tử lớn nhất sẽ đứng cuối mảng.

Ví dụ:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 10 | 4 | 7 | 15 | 3 | 1 |

Mảng ban đầu:

Left=0, right 5

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 10 | 4 | 7 | 15 | 3 | 1 |

While(left<=right)

Vòng lặp thứ hai:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 10 | 4 | 7 | 3 | 1 | 3 |

*i*=5

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 10 | 4 | 7 | 15 | 1 | 3 |

*i*=4

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 10 | 4 | 7 | 1 | 15 | 3 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 10 | 4 | 7 | 1 | 15 | 3 |

*i*=3

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 10 | 4 | 1 | 7 | 15 | 3 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 10 | 4 | 1 | 7 | 15 | 3 |

*i*=2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 10 | 1 | 4 | 7 | 15 | 3 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 10 | 1 | 4 | 7 | 15 | 3 |

*i*=1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 10 | 4 | 7 | 15 | 3 |

Kết thúc vòng lặp thứ hai ta có giá trị nhỏ nhất đứng đầu mảng:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 10 | 4 | 7 | 15 | 3 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 10 | 4 | 7 | 15 | 3 |

Vòng lặp thứ ba:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 10 | 4 | 7 | 15 | 3 |

*i*=1

*i*=2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 10 | 4 | 7 | 15 | 3 |

*i*=3

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 10 | 4 | 7 | 15 | 3 |

*i*=4

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 10 | 4 | 7 | 15 | 3 |

*i*=5

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 10 | 4 | 7 | 3 | 15 |

Kết thúc vòng lặp thứ ba ta có gia trị lớn nhất đứng cuối mảng:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 10 | 4 | 7 | 3 | 15 |

Tiếp tục thực hiện quá trinh trên cho đến khi kết thúc vòng lặp while(left<=right), ta sẽ được mãng sắp xếp hoàn chỉnh

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 3 | 4 | 7 | 10 | 15 |

## Thuật toán

|  |
| --- |
| void shakerSort(int\* arr, int soLuongPhanTu) |
| {  int left = 1, right = soLuongPhanTu - 1, i, k;  do  {  for (i = right; i >= left; i--)  {  if (arr[i - 1] > arr[i])  {  hoanVi(arr[i], arr[i - 1]);  }  k = i;  }  left = k + 1;  for (i = left; i <= right; i++)  {  if (arr[i - 1] > arr[i])  {  hoanVi(arr[i], arr[i - 1]);  }  k = i;  }  right = k - 1;  } while (left <= right);  } |

## Đánh giá

Độ phức tạp của thuật toàn là O(

|  |  |
| --- | --- |
| **Ưu điểm** | **Nhược điểm** |
| Thuật toán dễ dàng cài đặt  Sắp xếp nhanh với mảng ít phần tử. | Tốn nhiều bộ nhớ.  Đối với mảng nhiều phần tử, thuật toán sắp xếp rất lâu. |

# Shell sort

## Ý tưởng