CÁC THUẬT TOÁN SẮP XẾP

MỤC TIÊU

Hoàn tất bài thực hành này, sinh viên có thể:

- Hiểu được các thuật toán sắp xếp: Selection Sort, Heap Sort, Quick Sort, Merge Sort.
- Áp dụng các thuật toán sắp xếp để giải quyết các bài toán sắp xếp đơn giản.
- Áp dụng các thuật toán sắp xếp để giải quyết các bài toán sắp xếp trên danh sách các cấu trúc theo từng khóa.
- So sánh, đánh giá thời gian chạy của thuật toán với số lượng phần tử lớn.

Thời gian thực hành: **từ 120 phút đến 400 phút**

TÓM TẮT

Sắp xếp là quá trình xử lý một danh sách các phần tử (hoặc các mẫu tin) để đặt chúng theo một thứ tự thỏa mãn một tiêu chuẩn nào đó dựa trên nội dung thông tin lưu giữ tại mỗi phần tử.

Mức độ hiệu quả của từng giải thuật phụ thuộc vào tính chất của cấu trúc dữ liệu cụ thể mà nó tác động đến.

Có nhiều giải thuật sắp xếp: **Selection sort**, Insertion sort, Interchange sort, Bubble sort, Shaker sort, Binary Insertion sort, Shell sort, **Heap sort**, **Quick sort**, **Merge sort**, Radix sort…

Selection sort

- Chọn phần tử nhỏ nhất trong N phần tử ban đầu, đưa phần tử này về vị trí đúng là đầu dãy hiện hành.
- Xem dãy hiện hành chỉ còn N-1 phần tử của dãy ban đầu, bắt đầu từ vị trí thứ 2; lặp lại quá trình trên cho dãy hiện hành... đến khi dãy hiện hành chỉ còn 1 phần tử.

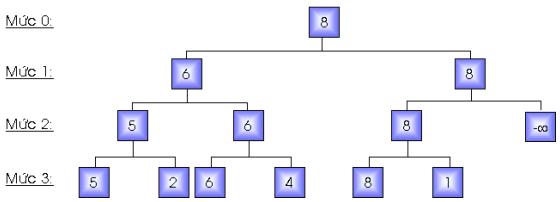
Heap sort

Heap là một dãy các phần tử $a_{left}, a_{left+1}, \dots, a_{right}$ sao cho: $a_i \geq a_{2i}$ và $a_i \geq a_{2i+1}, \forall i \in [left, right]$.

(a_i, a_{2i}), (a_i, a_{2i+1}): các cặp phần tử liên đới.

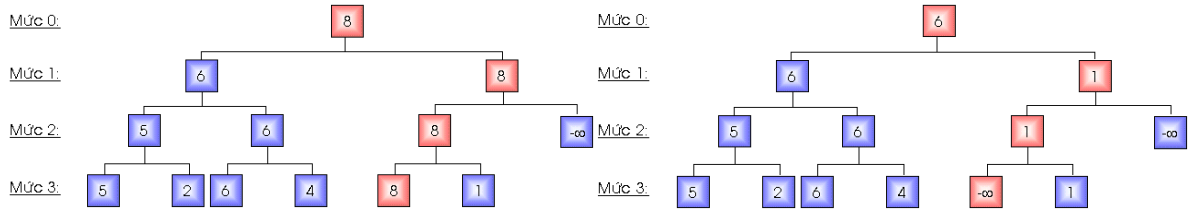
Heap được định nghĩa như trên được dùng trong trường hợp sắp xếp tăng dần, khi sắp xếp giảm dần phải đổi chiều các quan hệ.

Ví dụ 1: Dãy số 5 2 6 4 8 1 được bố trí theo quan hệ so sánh và tạo thành cấu trúc như sau:



Phần tử ở mức i chính là phần tử lớn trong cặp phần tử tương ứng ở mức i+1

Ví dụ 2: Loại bỏ 8 ra khỏi cây.



Tiến hành nhiều lần việc loại bỏ phần tử gốc của cây cho đến khi tất cả các phần tử của cây đều là -∞, khi đó xếp các phần tử theo thứ tự loại bỏ trên cây sẽ có dãy đã sắp xếp.

Quick sort

Phân chia dãy thành các đoạn con như sau:

$a_k \leqslant x$	$a_k = x$	$a_k \geqslant x$

- Đoạn thứ 2 đã có thứ tự.
- Nếu các đoạn 1 và 3 chỉ có 1 phần tử thì chúng cũng đã có thứ tự, khi đó dãy con ban đầu đã được sắp.
- Ngược lại, nếu các đoạn 1 và 3 có nhiều hơn 1 phần tử thì dãy con ban đầu chỉ có thứ tự khi các đoạn 1, 3 được sắp.
- Để sắp xếp các đoạn 1 và 3, ta lần lượt tiến hành việc phân hoạch từng dãy con theo cùng phương pháp phân hoạch dãy ban đầu vừa trình bày …

Với x là một phần tử tùy ý trong dãy và thường được chọn là vị trí chính giữa dãy ban đầu.

Merge sort

- Phân hoạch dãy ban đầu thành các dãy con liên tiếp mà mỗi dãy con đều đã có thứ tự..
- Làm giảm số dãy con bằng cách trộn từng cặp dãy con của hai dãy phụ thành một dãy con của dãy ban đầu.

NỘI DUNG THỰC HÀNH

Cơ bản

Sinh viên đọc kỹ phát biểu bài tập và thực hiện theo hướng dẫn:

Sử dụng các thuật toán Selection Sort, Heap Sort, Quick Sort, Merge Sort để sắp xếp một dãy các số nguyên theo thứ tự tăng dần.

Người dùng sẽ lần lượt nhập chiều dài n và các phần tử của dãy các nguyên A từ bàn phím. Toàn bộ dãy A được lưu trữ trong một mảng số nguyên.

Lần lượt sử dụng các thuật toán Selection Sort, Heap Sort, Quick Sort, Merge Sort để sắp xếp dãy A. Chương trình sẽ in các kết quả sắp xếp theo từng thuật toán ra màn hình.

Phân tích

Selection sort

- *Phân tích*
- Dùng vòng lặp để tìm phần tử nhỏ nhất trong dãy hiện hành.
- Đảo phần tử đó ra đầu mảng
- *Chương trình mẫu (CacThuatToanSapXep)*

```
#include <stdio.h>

void Swap(int &a, int &b)
{
	int c = a;
	a = b;
	b = c;
}

void SelectionSort(int a[],int N ){ //Ghi chú: tại sao không sử dụng kí hiệu & trong hàm này?
	int min; //chỉ số phần tử nhỏ nhất trong dãy hiện hành

	for (int  i=0; i<N-1 ; i++){ //Ghi chu: vòng lặp này dùng để làm gì?
		min = i;

		for(int j = i+1; j < N ; j++){ //Ghi chu: vòng lặp này dùng để làm gì?
		if (a[j] < a[min]){
			min = j; //Ghi chu: thao tác này dùng để làm gì?
		}
	  }

		 if (min != i){
		Swap(a[min], a[i]); //Ghi chu: thao tác này dùng để làm gì?
	  }
	}
}

void main()
{
	int x[10] = {12, 2, 8, 5, 1, 6, 4, 15}; // khởi tạo các giá trị trong mảng
	int n = 8; // số phần tử của mảng

	SelectionSort(x, n);

	for (int  i=0; i<n ; i++){
		printf("%d ", x[i]);
	}
}
```

- *Yêu cầu*

1. Biên dịch đoạn chương trình nêu trên.
2. Tại sao trong hàm SelectionSort, vòng lặp thứ nhất có điều kiện là i < N-1?
3. Trả lời các dòng lệnh có yêu cầu ghi chú.
4. Sửa lại chương trình để nhập dãy số nguyên từ file input.txt có nội dung như sau:

5 1 2 3 8 6 23 10

Sau đó dùng thuật toán Selection Sort sắp xếp dãy số nguyên trên tăng dần.

5. Sửa hàm SelectionSort trên để sắp xếp dãy số nguyên ở câu 4 giảm dần.

Heap Sort

- *Phân tích*

- Hiệu chỉnh dãy số ban đầu về dạng heap được định nghĩa trên mảng (hay list).
- Áp dụng thuật toán Heap Sort trên cấu trúc này.

- *Chương trình mẫu*

```
void Shift (int a[], int left, int right){
        int  x, curr, joint;
        curr = left; joint =2*curr+1; // a_joint :  Phần tử liên đới
        x = a[curr];

        while (joint <= right){
            if (joint < right){ // Ghi chú: điều kiện này có ý nghĩa gì?
                if (a[joint] < a[joint+1]){
                    joint = joint+1;
                }
            }

            if (a[joint]<x){
                break;  // Thỏa quan hệ liên đới
            }

            a[curr] = a[joint];
            curr = joint;   // Xét khả năng hiệu chỉnh lan truyền
            joint = 2*curr+1;
        }

        a[curr] = x;
}
```

- *Yêu cầu*

1. Trả lời các dòng lệnh có yêu cầu ghi chú.

2. Cho biết chức năng của đoạn chương trình trên.

3. Viết hàm void CreateHeap(int a[], int N); để chuyển đổi dãy $a_0, a_1, \ldots, a_{N-1}$ thành heap.

Gợi ý: Sử dụng hàm **Shift** bên trên với left hiện hành là phần tử ở giữa dãy (**(N-1)/2**). Lặp lại quá trình trên với left giảm dần về đầu dãy.

4. Viết hàm void HeapSort(int a[], int N); để sắp xếp một dãy số nguyên tăng dần.

Gợi ý: *Giai đoạn 1*: Hiệu chỉnh dãy ban đầu thành heap

Giai đoạn 2: Sắp xếp dãy số dựa trên heap.

- Xét dãy hiện hành là dãy nhập
- Hoán vị phần tử lớn nhất (a_0) về vị trí cuối.
- Xét dãy hiện hành loại đã trừ phần tử cuối.
- Hiệu chỉnh lại dãy hiện hành thành heap
- Lặp lại quá trình trên tới hết dãy ban đầu.

5. Bổ sung các hàm trên vào chương trình mẫu (CacThuatToanSapXep) đồng thời thay đổi hàm main và file input để sắp xếp dãy số nguyên sau tăng dần:

44 55 12 42 94 18 6 67

6. Viết lại thuật toán Heap Sort để sắp xếp dãy số ở câu 3 giảm dần.

Quick Sort

- *Phân tích*

- Chọn phần tử làm mốc.
- Tiến hành phân hoạch dãy ban đầu thành 3 phần $a_k<x$ (1), $a_k=x$ (2) và $a_k>x$ (3) theo thứ tự.
- Lặp lại thao tác trên trên 2 đoạn (1) và (3)

- *Chương trình mẫu*

```
void QuickSort(int a[], int left, int right){
      int i, j, x;

      if (left >= right){
          return;
      }

      x = a[(left+right)/2]; // chọn phần tử giữa làm giá trị mốc
      i = left; j = right;

      while(i < j) {
          while(a[i] < x){
              i++;
          }

          while(a[j] > x){
              j--;
          }

          if(i <= j) {
              Swap(a[i], a[j]);
              i++ ;
              j--;
          }
      }

      QuickSort(a, left, j);
      QuickSort(a, i, right);
}
```

- *Yêu cầu*

1. Bổ sung các hàm trên vào chương trình mẫu (CacThuatToanSapXep) đồng thời thay đổi hàm main và file input để sắp xếp dãy số nguyên sau tăng dần:

42 23 74 11 65 58 94 36 99 87

2. Sửa lại chương trình để đếm số phép gán và số phép so sánh sự dụng trong hàm QuickSort.

Merge Sort

- *Phân tích*

- Phân phối đều luân phiên các dãy con độ dài k từ mảng a vào hai mảng b và c.
- Trộn mảng b và mảng c vào mảng a.
- Lặp lại quá trình trên với k tăng gấp đôi đến khi k lớn hơn hay bằng chiều dài của dãy.

- *Chương trình mẫu*

```
int b[MAX], c[MAX], nb, nc; // Ghi chú: 2 mảng này dùng để làm gì?

void Distribute(int  a[], int N,  int &nb, int &nc, int k){
        int i, pa, pb, pc; //Ghi chú: các biến này có ý nghĩa gì?
        pa = pb = pc = 0;
        while (pa < N){
            for (i=0; (pa<N) && (i<k); i++, pa++, pb++){ //Ghi chú: vòng lặp này có ý nghĩa gì?
                    b[pb] = a[pa];
            }
            for (i=0; (pa<N) && (i<k); i++, pa++, pc++){ //Ghi chú: vòng lặp này có ý nghĩa gì?
                    c[pc] = a[pa];
            }
        }
        nb = pb;  nc = pc;
}
```

```
void Merge(int a[], int nb, int nc, int k){
        int pa, pb, pc;
        pa = pb = pc = 0;

        while ((pb < nb) && (pc < nc)){
            MergeSubarr(a, nb, nc, pa, pb, pc, k);
        }

        while (pb < nb){
            a[pa ++] = b[pb ++]; //Ghi chú: câu lệnh này có ý nghĩa gì?
        }

        while (pc < nc){
            a[pa ++] = c[pc ++]; //Ghi chú: câu lệnh này có ý nghĩa gì?
        }
}
```

Trong đó hàm MergeSubarr được cài đặt như sau:

```
void MergeSubarr(int a[], int nb, int nc, int &pa, int &pb, int &pc, int k){
        int rb, rc;
        rb = min(nb, pb+k);
       rc = min(nc, pb+k);

        while ((pb < rb) && (pc < rc)){
            if (b[pb] < c[pc])
                    a[pa ++] = b[pb ++];
            else   a[pa ++] = c[pc ++];
       }

        while (pb < rb){
            a[pa ++] = b[pb ++];
       }

        while (pc < rc){
            a[pa ++] = c[pc ++];
       }
}
```

Yêu cầu

1. Trả lời các dòng lệnh có yêu cầu ghi chú.
2. Cho biết chức năng của từng hàm trên.
3. Bổ sung các hàm cần thiết vào chương trình mẫu (CacThuatToanSapXep) và viết hàm void MergeSort(int a[], int N); để sắp xếp dãy số nguyên sau tăng dần.

5 2 9 3 7 2 4 11

Gợi ý: Xem lại 2 thao tác đã nêu bên trên.

4. Sửa lại chương trình để sắp xếp dãy số trên giảm dần.

Áp dụng – Nâng cao

Cho dãy số nguyên A như sau:

12 2 15 -3 8 5 1 -8 6 0 4 15

1. Sắp xếp dãy trên tăng dần.
2. Suy ra số lớn thứ 3 trong dãy.
3. Suy ra số lượng phần tử lớn nhất trong dãy.
4. Sắp xếp dãy trên theo thứ tự giá trị tuyệt đối tăng dần.
5. Sắp xếp dãy trên theo quy luật sau:
   - các số dương (nếu có) ở đầu mảng và có thứ tự giảm dần,
   - các số âm (nếu có) ở cuối mảng và có thứ tự tăng dần.
6. Sắp xếp dãy trên theo quy luật:

- các số chẵn (nếu có) ở đầu mảng và có thứ tự tăng dần,
- các số lẻ (nếu có) ở cuối mảng và có thứ tự giảm dần.

Cho một danh sách gồm các sinh viên sau:

STT	MSSV	Họ và tên	Năm sinh
1	1005	Trần Minh Thành	1991
2	1001	Trần Thị Bích	1988
3	1003	Trần Minh Thành	1990
4	1000	Võ Quang Vinh	1990
5	1008	Nguyễn Văn An	1990

7. Tạo một cấu trúc dữ liệu để xử lý danh sách trên.

8. Sắp xếp danh sách tăng dần theo mã số tăng dần.

9. Sắp xếp danh sách tăng dần theo tên (thứ tự bảng chữ cái) và năm sinh (nếu trùng tên thì sắp theo năm sinh tăng dần).

BÀI TẬP THÊM

1. Viết chương trình so sánh các thuật toán Selection Sort, Heap Sort, Quick Sort, Merge Sort về các mặt sau:

- Thời gian chạy.
- Số phép gán.
- Số phép so sánh.

Gợi ý: Dùng mẫu chương trình sau để tính thời gian chạy một đoạn lệnh

```
#include <time.h>
...
clock_t start, finish;
start = clock();

//Đoạn chương trình cần tính thời gian thực thi

finish = clock();
clock_t duration = finish - start; //Thời gian thực thi
...
```

2. Trong thuật toán QuickSort, nếu lấy x là phần tử dầu dãy, hãy viết chương trình và so sánh thời gian chạy thuật toán với khi lấy x là phần tử chính giữa dãy.

3. Sắp xếp dãy trên theo quy luật:

- các số chẵn (nếu có) có thứ tự tăng dần,
- các số lẻ (nếu có) có thứ tự giảm dần
- tính chất chẵn/lẻ tại mỗi vị trí trong dãy A không thay đổi sau khi sắp xếp (tức là trước khi sắp xếp, tại vị trí i của dãy A là số chẵn/lẻ thì tại vị trí i của mảng sau khi sắp xếp cũng là số chẵn/lẻ)

Ví dụ: A = (1, 1, 2, 3, 4, 5, 6, 7)

Kết quả kq = (1, 1, 6, 3, 4, 5, 2, 7).

4. Viết lại các thuật toán Selection Sort, Heap Sort, Quick Sort, Merge Sort.với cấu trúc dữ liệu dạng danh sách liên kết đơn.

5.Tìm hiểu và cài đặt thuật toán *Insertion Sort.*

6. Tìm hiểu và cài đặt thuật toán *Binary Insertion Sort.*

7. Tìm hiểu và cài đặt thuật toán *Interchange Sort.*

8. Tìm hiểu và cài đặt thuật toán *Bubble Sort.*

9. Tìm hiểu và cài đặt thuật toán *Shaker Sort.*

10. Tìm hiểu và cài đặt thuật toán *Shell Sort.*

11. Tìm hiểu và cài đặt thuật toán *Radix Sort* lần lượt trên 2 loại cấu trúc dữ liệu dạng mảng và dạng danh sách liên kết. So sánh và rút ra nhận xét.