1. **Selection Sort (Sắp xếp chọn):**
2. Ý tưởng: Ta thấy rằng khi một dãy ai, …, an đã được sắp xếp thì phần tử ai luôn là phần tử nhỏ nhất trong dãy ai, …, an. Nên ta có được ý tưởng để sắp xếp một dãy theo phương pháp chọn: tìm vị trí của phần tử nhỏ nhất trong n phần tử ban đầu và đưa giá trị của phần tử đó về đầu dãy. Tiếp tục với n - 1 phần tử còn lại. Và đến khi còn lại 1 phần tử thì phần dừng và phần tử đó chắc chắn là số lớn nhất. Cuối cùng, dãy đã được sắp xếp theo thứ tự tăng dần.
3. Thuật toán: Cho mảng gồm n phần tử a1, a2, …, an.

* Bước 1: Chọn i = 1.
* Bước 2: Tìm phần tử amin nhỏ nhất trong dãy ai, …, an.
* Bước 3: Hoán vị hai phần tử amin và an.
* Bước 4:
  + Nếu i ≤ n – 1. Lặp lại bước 2.
  + Ngược lại thì dừng chương trình.

1. Đánh giá thuật toán:

* Độ phức tạp về thời gian: O(n2).
* Độ phức tạp về không gian: O(1).

1. **Insertion Sort (Sắp xếp chèn):**
2. Ý tưởng: Đối với một dãy đã được sắp xếp, khi ta chèn một phần tử vào dãy đó ở một vị trí thích hợp thì dãy vẫn giữ nguyên được thứ tự ban đầu.

* Cho một dãy gồm n phần tử a1, a2, …, an. Chia dãy ban đầu thành 2 dãy con a1, a2, …, ai (gọi là dãy B) và ai + 1, ai + 2, …, an (gọi là dãy C). Giả sử dãy a1, a2, …, ai đã có thứ tự (tăng dần). Chọn một phần tử từ dãy C, để tiện ta chọn ai + 1 (tương đương C0) và chèn vào dãy B sao cho nó vẫn giữ nguyên thứ tự (tăng dần).
* Tìm vị trí k trong đoạn [1, i] thoả ak - 1 ≤ ai + 1 < ak, có 2 trường hợp xảy ra:
  + Nếu tồn tại k: dời tất cả phần tử trong đoạn [k, i] về sau (sang phải) một vị trí. Và sau đó cập nhật lại ak bằng với ai + 1.
* Ngược lại: các phần tử trong đoạn từ [1, i + 1] đã có thứ tự.
* Tiếp tục thực hiện cho đến khi chèn hết phần tử từ dãy C sang dãy B. Kết quả là dãy B sẽ chứa tất cả các phần tử ban đầu theo thứ tự tăng dần.

1. Thuật toán: Cho mảng gồm n phần tử a1, a2, …, an.

* Bước 1: Chọn i = 2.
* Bước 2: Tìm vị trí k trong đoạn [1, i - 1] thoả ak > ai. Nếu k tồn tại thì dời các phần tử trong đoạn [k, i – 1] về sau một đơn vị (dịch sang phải một đơn vị) và chèn ai vào vị trí k (gán ak = ai).
* Bước 3: Tăng i lên thêm 1.
  + Nếu i ≤ n. Lặp lại bước 2.
  + Ngược lại thì dừng chương trình.

1. Đánh giá thuật toán:

* Độ phức tạp về thời gian:
  + Trường hợp tốt nhất: O(n).
  + Trường hợp xấu nhất: O(n2).
  + Trường hợp trung bình: O(n2).
* Độ phức tạp về không gian: O(1).

1. **Bubble Sort (Sắp xếp nổi bọt):**
2. Ý tưởng: Bất đầu duyệt từ cuối dãy và hoán vị 2 phần tử kề nhau để đưa được phần tử nhỏ hơn về đầu dãy. Và tiếp tục thực hiện như vậy sau khi đã bỏ ra phần tử đã được đưa về đầu dãy. Thực hiện cho đến khi dãy chỉ còn một phần tử, lúc này dãy đã được sắp xếp theo thứ tự.
3. Thuật toán: Cho mảng gồm n phần tử a1, a2, …, an.

* Bước 1: Chọn i = 1.
* Bước 2:
* Nếu i nhỏ hơn hoặc bằng n - 1 🡺 Tiếp tục bước kế tiếp.
* Ngược lại 🡺 Chuyển sang bước 5.
* Bước 3: Cho j = n.
* Bước 4:
* Nếu j lớn hơn i.
* So sánh cặp phần tử aj - 1 và aj 🡺 Nếu aj - 1 > aj (nghịch thế) 🡺 hoán vị giá trị của hai phần tử aj - 1 và aj .
* Giảm j xuống 1 giá trị và quay lại bước 4.
* Ngược lại 🡺 Tăng giá trị i lên thêm 1 và quay lại bước 2.
* Bước 5: Dừng chương trình.

1. Đánh giá thuật toán:

* Độ phức tạp về thời gian: O(n2).
* Độ phức tạp về không gian: O(1).

1. **Heap Sort (Sắp xếp vun đống):**
2. Ý tưởng:
3. Thuật toán:
4. Algorithm’s review:

* Độ phức tạp về thời gian:
* Độ phức tạp về không gian:

1. **Merge Sort (Sắp xếp trộn):**
2. Ý tưởng: Dựa theo kỹ thuật chia để trị (divide and conquer). Lần lượt chia đôi dãy ban đầu thành 2 dãy con (kích thước dãy cha giảm một nửa). Tiếp tục việc chia đôi dãy con như vậy cho đến khi được n dãy con (một dãy con chứa một phần tử). Trộn theo thứ tự 2 dãy con được chia ra từ dãy con trước đó. Tiếp tục trộn như vậy cho đến khi nhận được một dãy có kích thước bằng với dãy ban đầu. Và dãy này cũng là dãy đã được sắp xếp theo thứ tự (tăng dần).

* Thuật toán trộn 2 mảng: mảng a gồm n phần tử a1, a2, …, an và mảng b gồm m phần tử b1, b2, …, bm.
* Bước 1: Khởi tạo mảng c với k = n + m phần tử và ic = ia = ib = 1.
* Bước 2:
  + Nếu ia ≤ n và ib ≤ m:
    - Nếu aia > bib (cặp phần tử nghịch thế) => Thêm phần tử aia vào mảng c và tăng ia lên thêm 1.
    - Ngược lại thêm phần tử bib vào mảng c và tăng ib lên thêm 1.
    - Tăng ic lên thêm 1.
  + Ngược lại => Chuyển qua bước 3.
* Bước 3:
  + Nếu ia ≤ n 🡺 Thêm tất cả phần tử còn lại của a vào c.
  + Ngược lại nếu ib ≤ m 🡺 Thêm tất cả phần tử còn lại của b vào c.
  + Ngược lại 🡺 Dừng thuật toán.

1. Thuật toán: Cho mảng gồm n phần tử a0, a1, ..., an.

* Bước 1: Chọn L = 1, R = n.
* Bước 2:
  + Nếu L < R 🡺 Chuyển sang bước 3.
  + Ngược lại 🡺 Chuyển sang bước 6.
* Bước 3: Cho mid = (L + R) / 2.
* Bước 4:
  + Nếu mid > 1:
    - Lặp lại bước 1 với R = mid và giữ nguyên L (aL, …, amid).
    - Lặp lại bước 1 với L = mid + 1 và giữ nguyên R (amid + 1, …, aR).
  + Ngược lại 🡺 Chuyển sang bước 5.
* Bước 5: Tiến hành trộn 2 mảng con bằng thuật toán trộn 2 mảng (đã trình bày ở mục 1). Quay lại bước 2.
* Bước 6: Dừng chương trình.

1. Đánh giá thuật toán:

* Độ phức tạp về thời gian: O(nlog(n)).
* Độ phức tạp về không gian: O(n).

1. **Quick Sort (Sắp xếp nhanh):**
2. Ý tưởng: Áp dụng thuật toán phân hoạch dãy a1, a2, …, an thành 2 phần:

* Dãy con 1: gồm các phần tử nhỏ hơn x.
* Dãy con 2: gồm các phần tử lớn hơn x.

Với x là một phần tử bất kỳ trong dãy ban đầu. Sau khi phân hoạch dãy ban đầu ta có được 3 phần:

* Dãy 1: ak < x với k = 1…i.
* Dãy 2: ak = x với k = (i + 1)…(j – 1).
* Dãy 3: ak > x với k = j…n.

Trong đó dãy 2 đã có thứ tự, nếu dãy 1 hoặc dãy 3 chỉ chứa 1 phần tử thì dãy đó cũng có thứ tự. Ngược lại, nếu dãy 1 hoặc dãy 3 chứa nhiều hơn 1 phần tử thì tiếp tục thực hiện việc phân hoạch với các dãy đó như đã trình bày.

* Thuật toán phân hoạch: có dãy gồm n phần tử a1, a2, …, an.
* Bước 1: Chọn i = L = 1, j = R = n và x = ak với L ≤ k ≤ R.
* Bước 2: Thực hiện việc chỉnh lại vị trí cho 2 phần tử ai, aj đang bị đặt sai vị trí:
  + Trong khi ai < x thì tăng i lên thêm 1.
  + Trong khi aj > x thì giảm j xuống 1.
  + Nếu i ≤ j (có 2 phần tử đang bị đặt sai vị trí) thì hoán vị 2 phần tử ai, aj và tăng i lên thêm 1, giảm j xuống 1.
* Bước 3:
  + Nếu i ≤ j (chưa duyệt hết dãy số) thì lặp lại bước 2.
  + Ngược lại thì dừng thuật toán.

1. Thuật toán: Cho dãy a gồm n phần tử a1, a2, …, an.

* Bước 1: Chọn L = 1 và R = n.
* Bước 2: Áp dụng thuật toán phân hoạch dãy aL, …, aR thành 3 phần:
  + Dãy 1: ak < x với k = L…j.
  + Dãy 2: ak = x với k = (j + 1)…(i – 1).
  + Dãy 3: ak > x với k = i…R.
* Bước 3:
  + Nếu L < j (dãy 1 vẫn còn chứa nhiều hơn 1 phần tử) thì lặp lại bước 2 với R = j và giữ nguyên L.
  + Nếu i < R (dãy 3 vẫn còn chứa nhiều hơn 1 phần tử) thì lặp lại bước 2 với L = I và giữ nguyên R.
* Bước 4: Dừng chương trình.

1. Đánh giá thuật toán:

* Độ phức tạp về thời gian:
  + Trường hợp tốt nhất: O(nlog(n)).
  + Trường hợp tệ nhất: O(n2).
  + Trường hợp trung bình: O(nlog(n)).
* Độ phức tạp về không gian:

1. **Radix Sort (Sắp xếp theo cơ số):**
2. Ý tưởng:
3. Thuật toán:
4. Đánh giá thuật toán:

* Độ phức tạp về thời gian:
* Độ phức tạp về không gian:

TÀI LIỆU THAM KHẢO

1. Đỗ Xuân Lôi: “Cấu trúc dữ liệu và giải thuật”. Nhà xuất bản Đại học Quốc gia Hà Nội – 2010.
2. Trần Hạnh Nhi – Dương Anh Đức: “NHẬP MÔN CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT”. Trường Đại học Khoa học Tự nhiên TP.HCM – 2003.