1. Selection sort
   * Ý tưởng
     + Cho mảng có n phần tử được chia ra 2 phần: phần chưa được sắp xếp và được sắp xếp.
     + Phần được sắp xếp có thể bắt đầu ở đầu hoặc cuối mảng.
     + Bao gồm 2 quá trình: tìm kiếm và sắp xếp
     + Quá trình tìm kiếm: trong phần chưa sắp xếp tìm giá trị lớn nhất (hoặc nhỏ nhất)
     + Quá trình sắp xếp: khi tìm được giá trị thì đưa phần tử đó đến phần được sắp xếp.
     + Sắp xếp thứ tự tăng dần hoặc giảm dần phụ thuộc vào vị trí đặt phần tử vào phần được sắp xếp và việc tìm phần tử lớn nhất hay nhỏ nhất.
     + Độ phức tạp: quá trình tìm kiếm là O(2^n) vì có 2 vòng for, trong khi quá trình sắp xếp chỉ có O(n). Chi phí chạy của Selection sort hoàn toàn cố định. Trong code có thêm cải tiến nhỏ: đầu tiên duyệt xem thử các phần tử có dạng tăng dần hay không, tuy tăng thêm chi phí nhưng lượng tăng vô cùng nhỏ.
   * A picture containing text, cabinet, screenshot

     Description automatically generatedCách làm việc
2. Heap sort
   * Ý tưởng
     + Heap sort là cải tiến của Selection sort, cũng có 2 quá trình: tìm kiếm và sắp xếp với điểm khác biệt là sử dụng cấu trúc heap trong quá trình tìm kiếm.
     + Cấu trúc heap có 2 kiểu sắp xếp là max-heap và min-heap với max-heap thì phần tử đầu tiên của mảng luôn luôn lớn nhất, với min-heap thì nhỏ nhất.
     + Cho mảng n phần tử được chia thành 2 phần: phần chưa sắp xếp và phần được sắp xếp đặt ở cuối mảng.
     + Quá trình tìm kiếm: tạo cấu trúc heap ở phần chưa sắp xếp, sau khi sắp xếp thì phần tử đầu tiên là phần tử cần tìm.
     + Quá trình sắp xếp: hoán đổi phần tử đầu tiên với phần tử cuối cùng trong phần chưa được sắp xếp, lúc này kích thước của phần chưa được sắp xếp sẽ giảm đi 1 và phần được sắp xếp tăng thêm 1.
     + Lặp lại 2 quá trình trên cho đến khi kích thước của phần chưa được sắp xếp bằng 0.
     + Mảng đã được sắp xếp có thứ tự tăng dần khi dùng max-heap, giảm dần khi dùng min-heap.
     + VD: Cho 1 mảng:

4 8 5 0 2 7 11 9 3 10

Tạo cấu trúc heap ta có được:

11 10 7 9 8 4 5 0 3 2

Hoán đổi phần tử 11 với phần tử 2:

2 10 7 9 8 4 5 0 3 | 11

( bên trái dấu “|” là phần chưa sắp xếp, bên phải là phần được sắp xếp)

Tạo lại cấu trúc heap ở phần chưa sắp xếp:

10 8 9 7 4 5 0 3 2 | 11

Hoán đổi phần tử 10 với phần tử 2:

2 8 9 7 4 5 0 3 | 10 11

Tạo lại cấu trúc heap ở phần chưa sắp xếp:

9 7 8 4 5 0 3 2 | 10 11

Tiếp tục như vậy, kết quả cuối cùng sẽ là:

0 2 3 4 5 7 8 9 10 11

* + - Dựa vào ví dụ trên, khi mới tạo lại cấu trúc thì có sự thay đổi lớn vị trí của các phần tử, nhưng từ các lần sau ta có thể thấy chỉ vó vài phần tử thay đổi vị trí, các phần tử ở sau gần như giữ nguyên. Đó là điểm đặc biệt của cấu trúc heap, nó giúp giảm các tác vụ xử lí bằng cách tận dụng cấu trúc đã được sắp xếp từ trước.
    - Độ phức tạp: O( n\*log2(n) ), không có best case, worst case hay average case.
  + Cách làm việc
    - Tạo cấu trúc heap:  
      -Cho mảng có n phần tử: nếu n = 5 thì sẽ chạy từ 0 đến 4 vì mảng trong code bắt đầu từ 0. Đặt k là vị trí phần tử của mảng trong code .Chú ý điều này để tránh sự nhầm lẫn.

-Cấu trúc max-heap sẽ có các phần tử lớn hơn hoặc bằng phần tử con ở vị trí 2k + 1 và 2k + 2 (2k và 2k + 1 nếu k bắt đầu từ 1). Tương tự với min-heap sẽ có các phần tử bé hơn hoặc bằng các phần tử con.

-Bắt đầu từ phần tử có vị trí k = n/2 – 1 (nếu k là số thập phân thì làm tròn xuống) vì phần tử tại vị trí k có phần từ con nằm ở cuối mảng. VD:

+Cho mảng n = 7, k = 2 -> k = 2 có phần tử con là k = 5 và k = 6.

+Cho mảng n = 8, k = 3 -> k = 3 có phần từ con là k = 7 (không có k = 8)

-Từ k cho là phần tử a bắt đầu xét, chọn phần tử con lớn nhất rồi so sánh với a. Với max-heap, nếu k bé hơn phần tử con thì hoán đổi vị trí với nhau. Nếu xảy ra hoán đổi và a ở vị trí mới cũng có phần tử con thì xét tiếp, cho đến khi a không còn phần tử con hoặc a lớn hơn hoặc bằng phần tử con (chỉ xét trong phần chưa sắp xếp). Khi xét xong, giảm k đi 1 và xét tại phần tử ở k đó giống như phần tử a. Lặp lại cho đến khi k bằng -1.

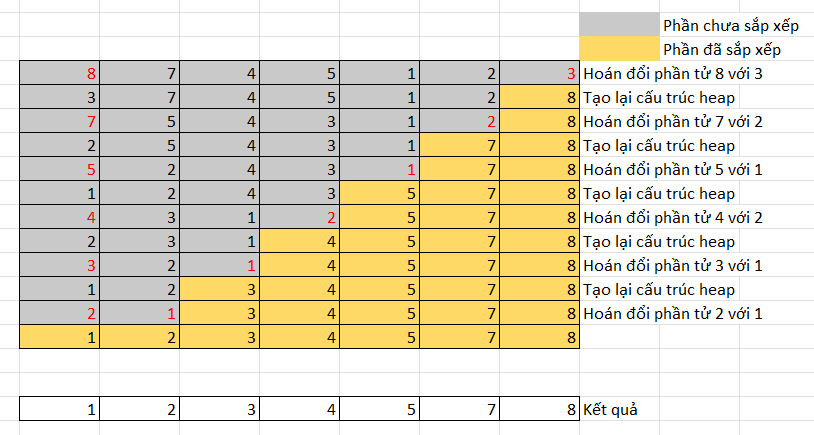
* + - Tạo lại cấu trúc heap:  
      -Dùng sau khi hoán đổi vị trí đầu tiên của phần chưa sắp xếp. Với đặc trưng của cấu trúc heap, khi hoán đổi phần tử đầu tiên, các phần tử còn lại vẫn giữ cấu trúc heap trong phần chưa sắp xếp. Vì vậy ta chỉ cần xét phần tử đầu tiên, cách làm giống với việc xét trong tạo cấu trúc heap với k = 0.
    - Sắp xếp:  
      -Sau khi tạo cấu trúc heap, lấy phần tử đầu tiên hoán đổi phần tử cuối cùng trong phần chưa sắp xếp (tức là phía trước phần đã sắp xếp).

-Sau khi hoán đổi, số phần tử trong phần chưa sắp xếp giảm đi 1 và trong phần được sắp xếp tăng thêm 1. Tạo lại cấu trúc heap và hoán đổi vị trí. Lặp lại cho tới khi số phần tử trong phần chưa sắp xếp bằng 0;

* + - Calendar

      Description automatically generatedHình ví dụ minh hoạ:

Quá trình tạo cấu trúc heap

 Quá trình sắp xếp