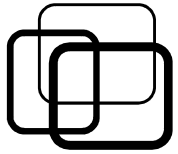
Sắp xếp và Quy hoạch động

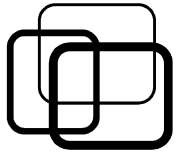
GV. Nguyễn Minh Huy

Nội dung



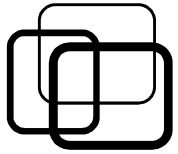
- Thuật toán sắp xếp.
- Quy hoạch động.

Nội dung



- **Thuật toán sắp xếp.**
- Quy hoạch động.

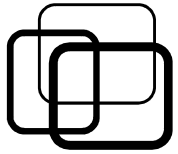
Thuật toán sắp xếp



- Khái niệm sắp xếp:
  - Bài toán:
    - Cho mảng N phần tử.
    - Bố trí các phần tử theo thứ tự.
    - ➔ N! cách bố trí!
  - Các thuật toán:

Thuật toán	Độ phức tạp			Không gian bộ nhớ
	Tốt	Xấu	Trung bình	
Interchange Sort	N^2	N^2	N^2	1
Selection Sort	N^2	N^2	N^2	1
Merge Sort	N logN	N logN	N logN	N
Quick Sort	N logN	N^2	N logN	logN

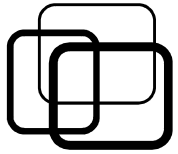
Thuật toán sắp xếp



- Sắp xếp đổi chỗ (Interchange Sort):
  - Ý tưởng:
    - Mảng có thứ tự không có nghịch thế!
    - Duyệt tất cả cặp phần tử.
    - Đổi chỗ nếu phát hiện nghịch thế.
  - Cài đặt:

```
interchange_sort( mảng A, kích thước N )
{
        for ( int i = 0; i < N – 1; i++ )
            for ( int j = i + 1; j < N; j++ )
                if ( a[ j ] < a[ i ] )   // Phát hiện nghịch thế.
                    swap( a[ i ], a[ j ] );
}
```

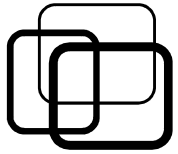
Thuật toán sắp xếp



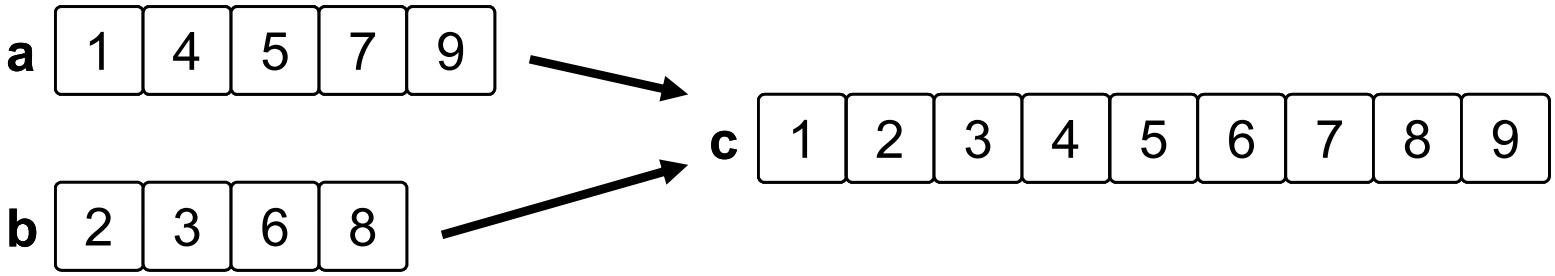
- Sắp xếp chọn (Selection Sort):
  - Ý tưởng:
    - Tìm phần tử nhỏ nhất đưa lên đầu.
    - Lặp lại với phần còn lại của mảng.
  - Cài đặt:

```
selection_sort( mảng A, kích thước N )
{
        for ( int i = 0; i < N – 1; i++ )
        {
            int min = i;
            for (int j = i + 1; j < N; j++ )
                if ( a[ j ] < a[ min ] )
                    min = j;
            swap( a[ i ], a[ min ] );
        }
}
```

Thuật toán sắp xếp



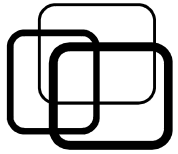
- Sắp xếp trộn (Merge Sort):
  - Bài toán trộn 2 mảng có thứ tự:



    - Quyết đinh chọn phần tử ở mảng kết quả:

```
// i, j, k lần lược là vị trí đang xét ở mảng a, b, c.
if ( a[ i ] < b[ j ] )
        c[ k ] = a[ i++ ];
else
        c[ k ] = b[ j++ ];
➔ c[ k ] = (a[ i ] < b[ j ]) ? a[ i++ ] : b[ j++ ];
```

Thuật toán sắp xếp

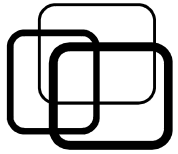


- Sắp xếp trộn (Merge Sort):
  - Dùng kỹ thuật chia để trị:

merge_sort(mảng A, kích thước N)
{
    if (**N == 1**)
        Kết thúc;
    else
    {
        Chia đôi A thành A_1 và A_2;
        merge_sort(A_1, kích thước A_1);
        merge_sort(A_2, kích thước A_2);

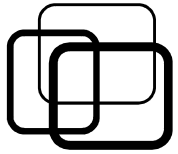
        merge_array(A_1, kích thức A_1, A_2, kích thước A_2, A);
    }
}

Nội dung



- Thuật toán sắp xếp.
- **Quy hoạch động.**

Quy hoạch động



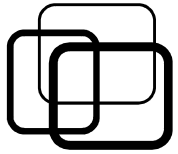
- Khái niệm quy hoạch động:
  - Lời giải đệ quy:
    - Trường hợp cơ bản: giải tường minh.
    - Suy biến bài toán về trường hợp đơn giản hơn.
  - Nếu nhiều trường hợp suy biến trùng lặp?
    - Xây dựng bảng tra lời giải cho các trường hợp.
    - Quy hoạch động = Quy hoạch bảng tra lời giải.

Bài toán Fibonacci

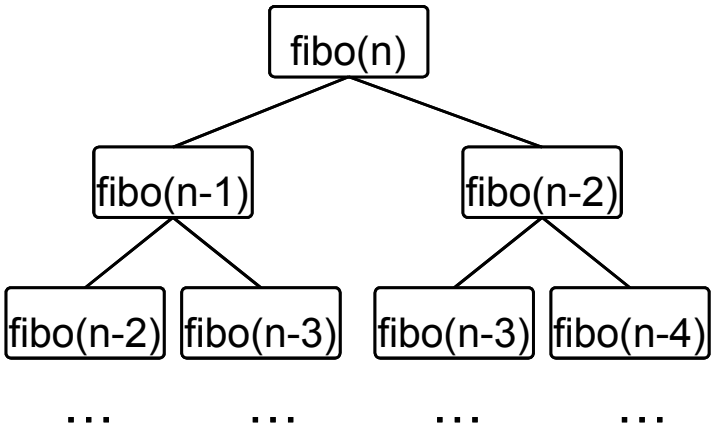
- F(0) = 1, F(1) = 1
- F(n) = F(n – 1) + F(n – 2)

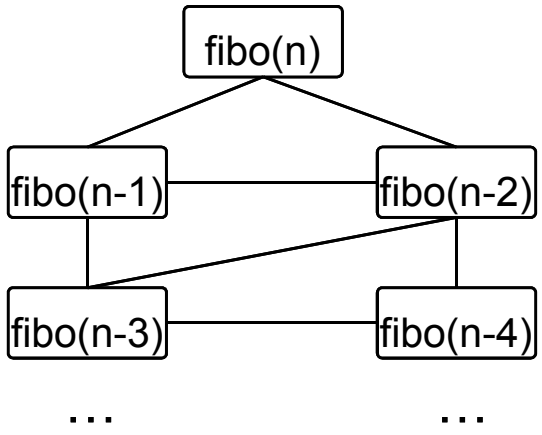
n = 0	1
n = 1	1
n = 2	2
n = 3	3

Quy hoạch động

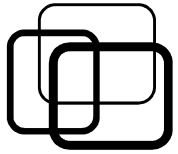


- Khái niệm quy hoạch động:
  - Khi nào dùng quy hoạch động?
    - Bài toán đệ quy.
      ➔ Có thể suy biến về trường hợp cơ bản.
    - Các trường hợp suy biến trùng lắp nhau.
      ➔ Có thể dùng bảng tra để lưu lời giải.
    - Không quá nhiều trường hợp suy biến.
      ➔ Có thể lưu trữ bảng tra.





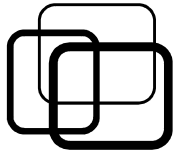
Quy hoạch động



- Phân loại quy hoạch động:
  - Quy hoạch từ trên xuống (top-down):
    - Quy hoạch bảng tra cho trường hợp tổng quát trước.
    - Gọi đệ quy đến trường hợp đơn giản hơn.
    - Gặp trường hợp đã giải thì truy xuất bảng tra.

```
solve_top_down( trường hợp N, bảng tra lời giải T )
{
        if ( đã giải trường hợp N )
                return T[ N ];

        // Giải trường hợp N và đưa vào bảng tra.
        if ( N cơ bản )
                T[ N ] = lời giải cơ bản;
        else
                T[ N ] = solve_top_down( N – 1, T );
}
```

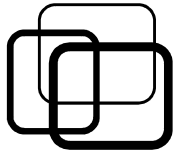
Quy hoạch động



- Phân loại quy hoạch động:
  - Quy hoạch từ dưới lên (bottom-up):
    - Quy hoạch bảng tra cho trường hợp cơ bản trước.
    - Dùng vòng lặp tính trường hợp tổng quát dựa vào bảng tra.
    - Cách thức khử đệ quy.

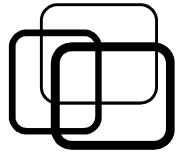
```
solve_bottom_up( trường hợp N )
{
        Khai báo bảng tra T;
        Khởi tạo T với các trường hợp cơ bản;

        for (int i = trường hợp tổng quát đầu tiên; i <= N; i++ )
                T[ i ] = tính dựa vào T[ i – 1 ];

        return T[ N ];
}
```

Tóm tắt



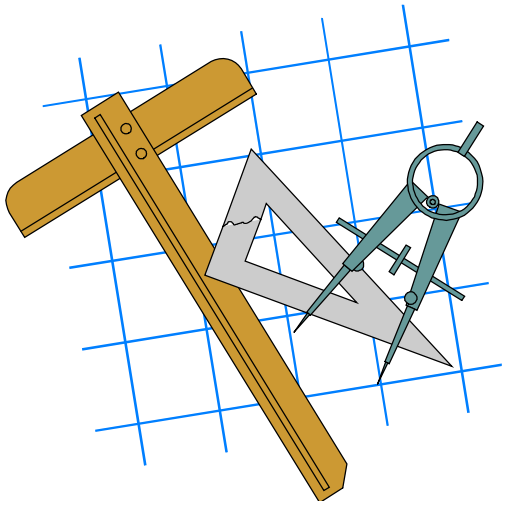
- Thuật toán sắp xếp:
  - Interchange Sort.
  - Selection Sort.
  - Merge Sort.
- Quy hoạch động:
  - Quy hoạch bảng tra lời giải.
  - Phân loại:
    - Top-down: tổng quát trước, dùng đệ quy.
    - Bottom-up: cơ bản trước, dùng vòng lặp.

Bài tập

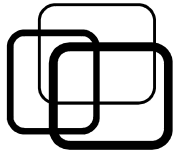


- **Bài tập 10.1:**

  Viết chương trình C cài đặt sắp xếp trộn trên danh sách liên kết đơn.



Bài tập



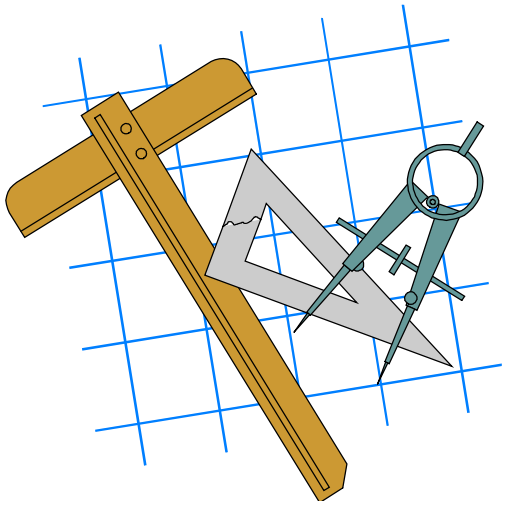
- ## Bài tập 10.2:

  Viết chương trình C dùng quy hoạch động (cả 2 cách top-down và bottom-up) để tìm dãy con tăng không liên tục dài nhất trong dãy N phần tử.

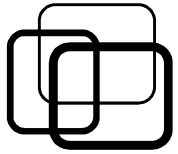
  Ví dụ:

  Cho dãy A_N = { 1 5 9 2 4 7 8 }

  Dãy con tăng không liên tục dài nhất S_N = { 1 2 4 7 8 }



Bài tập



Bài tập 10.3:

Viết chương trình C dùng quy hoạch động (cả 2 cách top-down và bottom-up) để giải bài toán sau:

- Cho N loại tiền mệnh giá { T0, T1, …, TN }
- Tìm cách đổi số tiền M ra các tờ tiền có mệnh giá được cho.
- Các tờ tiền được dùng để đổi là ít nhất.

Ví dụ 1:

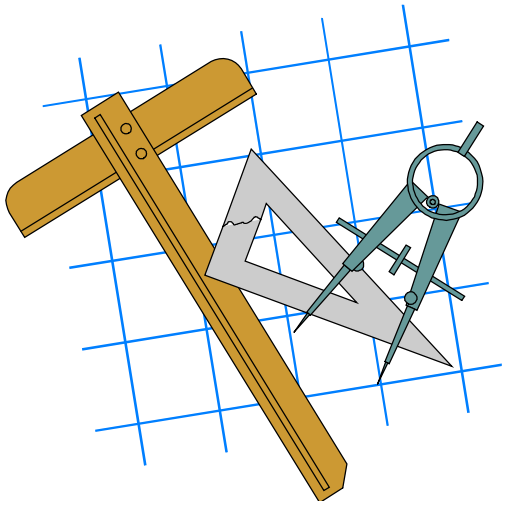
Cho các loại tiền mệnh giá { 5 10 20 40 }

Số tiền cần đổi 90 = 40 + 40 + 10.

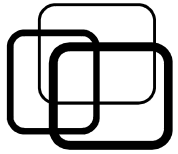
Ví dụ 2:

Cho các loại tiền mệnh giá { 10 50 60 90 }

Số tiền cần đổi 110 = 50 + 60.



Bài tập



- ## Bài tập 10.4:

  Viết chương trình C dùng quy hoạch động (cả 2 cách top-down và bottom-up) để giải bài toán sau:

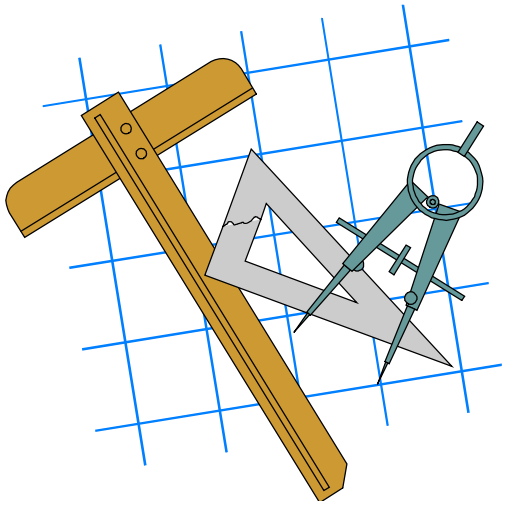
  - Tìm cách biểu diễn số nguyên N bằng tổng các số nguyên tố.
  - Các số nguyên tố dùng để biểu diễn là ít nhất.

  Ví dụ:

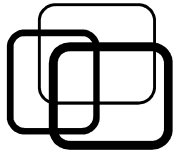
  6 = 3 + 3 (ít hơn 2 + 2 + 2)

  8 = 3 + 5

  38 = 31 + 7



Bài tập



Bài tập 10.5:

Viết chương trình C dùng quy hoạch động (cả 2 cách top-down và bottom-up) để giải bài toán sau:

Một lâu đài cổ có M tầng lầu, mỗi tầng lầu có N phòng. Tại mỗi phòng của lâu đài đều có 3 cái thang thông với 3 phòng của tầng lầu phía trên gồm: phòng ngay bên trên, phòng trên bên trái và phòng trên bên phải. Trong mỗi phòng có một rương tiền với giá trị nhất định.

Một người săn kho báu xuất phát từ tầng trệt của lâu đài. Anh ta vào một phòng bất kỳ rồi từ đó đi lên các phòng ở tầng trên. Mỗi tầng anh ta chỉ ghé qua một phòng.

Hãy tìm lộ trình phòng ở mỗi tầng mà người săn kho báu ghé qua để thu được số tiền nhiều nhất.