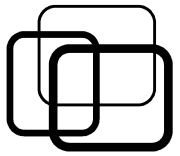
Advanced Topics

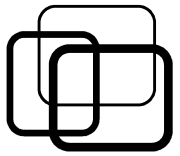
Inst. Nguyễn Minh Huy

Contents



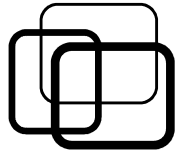
- Sort algorithms.
- Dynamic programming.

Contents



- **Sort algorithms.**
- Dynamic programming.

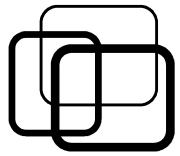
Sort algorithms



- Sorting concepts:
  - Sorting problem:
    - Given array of N integers.
    - Organize elements in ascending order.
    - ➔ N! ways to organize!
  - Sort algorithms:

Algorithm	Complexity			Storage Space
	Best	Worst	Average	
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

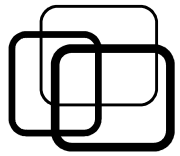
Sort algorithms



- Interchange sort:
  - Idea:
    - Ordered array does not have conflict pair!
    - Iterate every possible pair.
    - Swap if it is conflict pair..
  - Implementation:

```
interchange_sort( array A, size N )
{
        for ( int i = 0; i < N – 1; i++ )
            for ( int j = i + 1; j < N; j++ )
                if ( a[ j ] < a[ i ] )   // Conflict pair.
                    swap( a[ i ], a[ j ] );
}
```

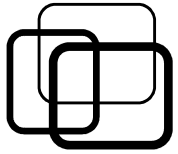
Sort algorithms



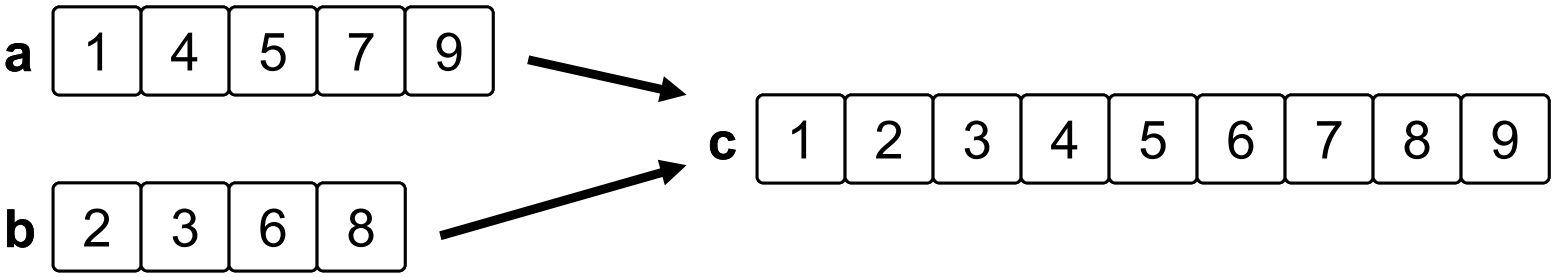
- Selection sort:
  - Idea:
    - Find smallest element then move to first.
    - Do the same with the rest of the array.
  - Implementation:

```
selection_sort( array A, size 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 ] );
        }
}
```

Sort algorithms



- Merge sort:
  - Merge two ordered arrays into one:

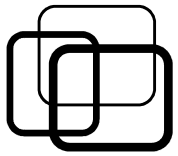


    - Select element from a or b to put into c?

```
// i, ia, ib current positions in c, a, b.
if ( a[ ia ] < b[ ib ] )
        c[ i ] = a[ ia++ ];
else
        c[ i ] = b[ ib++ ];
```

➔ c[i] = (a[ia] < b[ib]) ? a[ia++] : b[ib++];

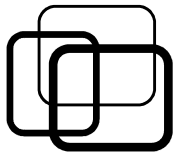
Sort algorithms



- Merge sort:
  - Divide-and-conquer technique:

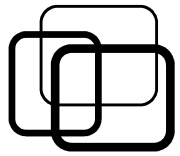
```
merge_sort( array A, size N )
{
        if ( N == 1 )
            Stop;
        else
        {
            Split A into A1 and A2;
            merge_sort(A1, size N1 );
            merge_sort(A2, size N2 );

            merge_array(A1, size N1, A2,size N2, A );
        }
}
```

Contents



- Sort algorithms.
- **Dynamic programming.**

Dynamic programming



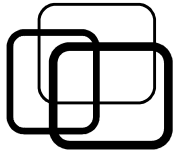
- Basic concepts:
  - Recursive solution:
    - Base case: solve explicitly.
    - Recursive case: reduce problem to simpler cases.
  - Redundancy of recursive solutions?
    - Store solutions to look up later.
    - Dynamic programming = Organize look up table.

Fibonacci problem

- 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

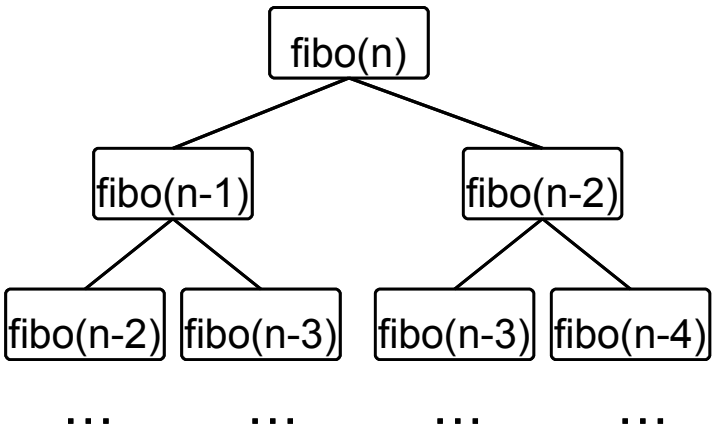
Dynamic programming

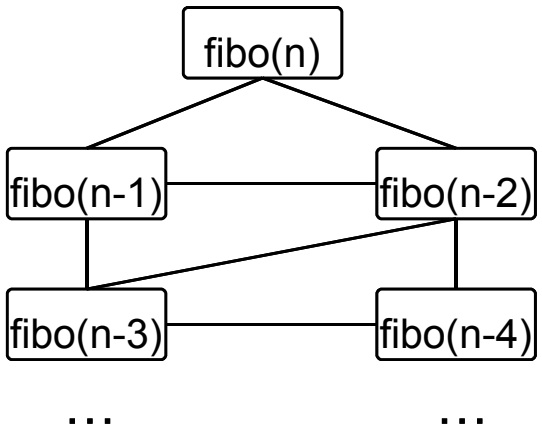


- Basic concepts:
  - When to use dynamic programming?
    - Recursive problem.

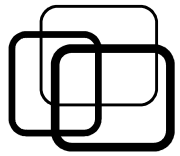
      ➔ Can be reduced to simpler cases.
    - Redundancy of recursive solutions.

      ➔ Organize look up table.
    - Enough memory to store look up table.





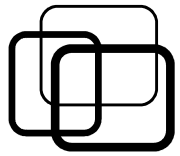
Dynamic programming



- Organizing approaches:
  - Top-down:
    - Organize look up table from general to simple cases.
    - Use recursion to solve general case from simpler ones.
    - Look up a case before solving.

```
solve_top_down( case N, look up table T )
{
        if ( N is solved )
                return T[ N ];

        // Solve N with recursion and put into look up table.
        if ( N is base case )
                T[ N ] = solve explicitly;
        else
                T[ N ] = solve_top_down( N – 1, T );
}
```

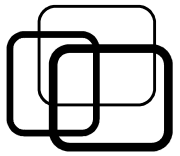
Dynamic programming



- Organizing approaches:
  - Bottom-up:
    - Organize look up table from simple to general cases.
    - Initialize base cases in look up table.
    - Use loop to solve general case from simpler ones.

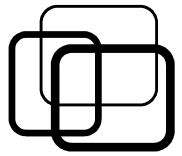
```
solve_bottom_up( case N )
{
        Declare look up table T;
        Initialize T with some base cases;

        for (int i = base case; i <= N; i++ )
                T[ i ] = solve from T[ i – 1 ];

        return T[ N ];
}
```

Summary



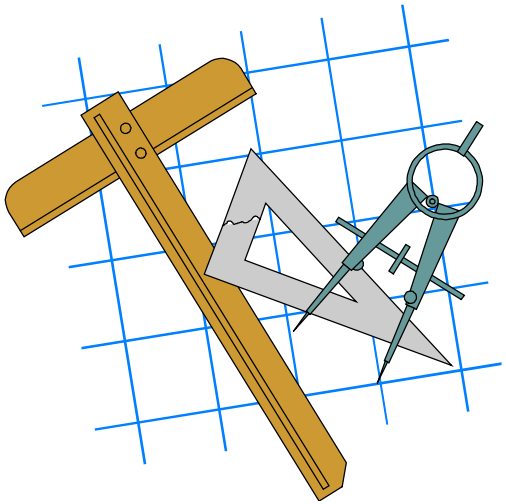
- Sort algorithms:
  - Interchange Sort.
  - Selection Sort.
  - Merge Sort.
- Dynamic programming:
  - Organize look up table.
  - Classifications:
    - Top-down: use recursion.
    - Bottom-up: use loop.

Practice

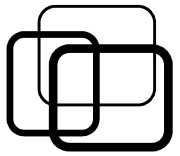


- **Practice 10.1:**

  Write C program to implement Merge Sort for Singly linked list.



Practice



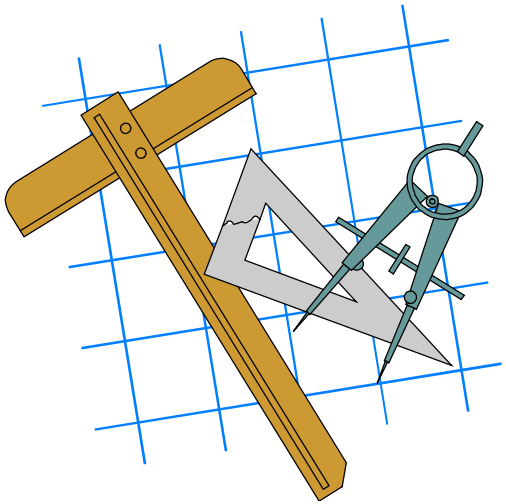
- ## Practice 10.2:

  Write C program to find **longest increasing subsequence** in a given sequence of N elements (use both top-down and bottom-up dynamic programming).

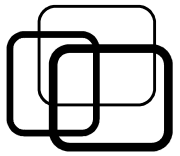
  Example:

  Given A_N = { 1 5 9 2 4 7 8 }

  Longest increasing subsequence S_N = { 1 2 4 7 8 }



Practice



- ## Practice 10.3:

  Write C program to solve the following problem by using both top-down and bottom-up dynamic programming:

  - Given N types of money notes { T0, T1, …, TN }
  - Given M amount of money.
  - Find a way to exchange M with smallest number of money notes.

  Example 1:

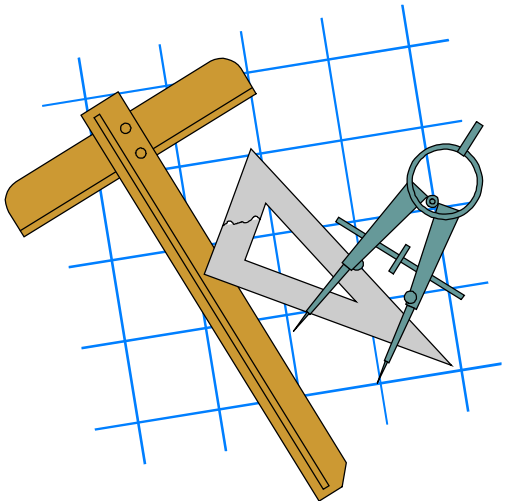
  Given money notes { 5 10 20 40 }

  Given amount of money 90 = 40 + 40 + 10.

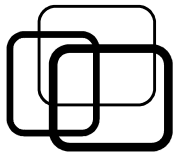
  Example 2:

  Given money notes { 10 50 60 90 }

  Given amount of money 110 = 50 + 60.



Practice



- Practice 10.4:

  Write C program to solve the following problem by using both top-down and bottom-up dynamic programming:

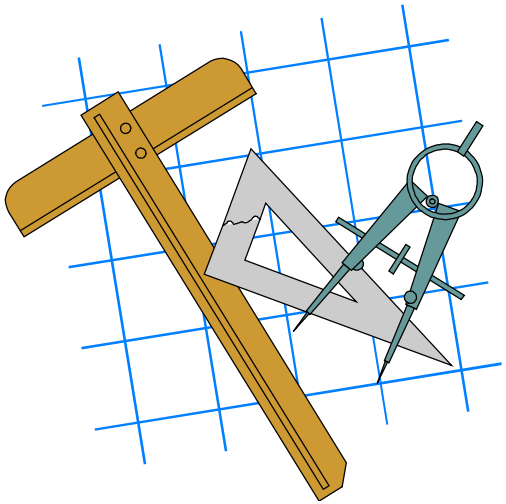
  - Given a positive integer N.
  - Find a way to represent N with sum of the least primes.

  Example:

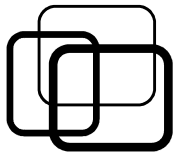
  6 = 3 + 3 (less than 2 + 2 + 2)

  8 = 3 + 5

  38 = 31 + 7



Practice



- Practice 10.5:

  Write C program to solve the following problem by using both top-down and bottom-up dynamic programming:

  An old castle has M floors, and N rooms at each floor. At each room, there are 3 stairways lead to 3 adjacent rooms at the upper floor: upper straight, upper left, upper right. There are an amount of gold in each room.

  A gold seeker departs from the ground floor of the castle. She choose to enter an arbitrary room and follow the stairways to go to the upper floors. She can only enter one room at each floor to collect gold.

  Find a route for her to enter which room at each floor to collect the largest amount of gold.