MAT2440, Classwork30, Spring2025

ID:________________________________ Name:________________________________

1. Use the bubble sort to put 2, 5, 4, 3 into increasing order.

Initilation $n=4 \Rightarrow i=1$ to 3 . $j=1$ to $4-i$

$i=1$

$j=1$	$j=2$	$j=3$	
2	2	2	2
5	5	4	4
4	4	5	3
3	3	3	$\boxed{5}$

$i=2$

$j=1$	$j=2$	
2	2	2
4	4	3
3	3	$\boxed{4}$
5	5	$\boxed{5}$

$i=3$

$j=1$	
2	2
3	3
4	4
5	5

previous sorted

2. Try one round of the insertion sort to put "4" from $\boxed{2, 3, 5, 8,}$ 4, 7, 1 into right location.

4th Round: insert $a_5 = 4$ to the right position. $i=5$.

$i=5$, $j=1$

$a_i > a_j$ and $i > j$ (T or F)		$j = j+1$
$4 > 2$ $(a_5 > a_1)$	$\wedge$ $5 > 1$ (T)	$j=2$
$j=2$: $4 > 3$	$\wedge$ $5 > 2$ (T)	$j=3$
$j=3$: $\boxed{4 > 5}$	$\wedge$ $5 > 3$ (F)	STOP

from while-loop we get $j=3$ when $i=5$

$m := a_5 \Rightarrow m = 4 \leftarrow$ temp. storage

$k = 0$ to 1 ($\leftarrow i-j-1 = 5-3-1$) need to move a_3, a_4 to a_4 a_5 first

when $k=0$ $a_5 = a_4$ ($a_5 = 8$) move "8" from a_4 to a_5

when $k=1$ $a_4 = a_3$ ($a_4 = 5$) move "5" from a_3 to a_4

$a_j = m$ ($j=3$, $a_3 = 4$)

3. Introduction of **Optimization Problems** and **Greedy Algorithms**.

__Optimization__ Problem: Find a solution to the given problem that either minimizes or maximizes the value of some parameters.

__Greedy__ Algorithm: Instead of considering all sequences of steps, this approach selects the best choice at each step. Once we know that a greedy algorithm finds a feasible solution, we need to determine whether it has found an __optimal__ solution.

To do this, we either prove that the solution is **optimal** or show that there is a __counterexample__ where the algorithm yields a nonoptimal solution.

4. Introduction of Cashier's Algorithm.

Cashier's algorithm: One example of greedy algorithms that makes change using coins.

Problem: making n cents change with quarters, dimes, nickels, and pennies, and using **the least total** number of coins.

Algorithm: At each step we choose the coin of the largest denomination possible to Add to the pile of change without exceeding n cents.

Here we use an example to explain the algorithm:

$$67\ cents = ?\ quarters + ?\ dimes + ?\ nickels + ?\ pennies$$

Step1: $\frac{67}{25} = 2\frac{17}{25}$ means 2 quarter(s) + 17 cents

Step2: $\frac{17}{10} = 1\frac{7}{10}$ means 1 dime(s) + 7 cents

Step3: $\frac{7}{5} = 1\frac{2}{5}$ means 1 nickel(s) + 2 cents

Step4: $\frac{2}{1} = 2$ means 2 pennies

Thus, we have 67 cents = 2 quarters + 1 dime + 1 nickel + 2 pennies.

Pseudocode:

procedure $change(c_1, c_2, \cdots, c_r$: values of denominations of coins, $c_1, > c_2 > \cdots > c_r$, n: a positive integer)
$r :=$ the length of $\{c_i\}$
for $i :=$ 1 **to** r
    $d_i :=$ 0 $\{d_i$ counts the coins of denomination c_i used$\}$
    while $(n \geq c_i)$
        $d_i := d_i + 1$ { add a coin of denomination } c_i
        $n := n - c_i$
return $(d_1, d_2, \cdots, d_r)$ $\{d_i$ counts the coins of denomination c_i used$\}$